# Study, Implementation and Comparison of Different Multipliers based on Array, KCM and Vedic Mathematics Using EDA Tools

**Mohammed Hasmat Ali***, **Anil Kumar Sahani****

* M.Tech, Electrical Engineering, National Institute of Technology Patna, Patna, India
** Assoc. Professor & Head, Electrical Engineering, National Institute of Technology Patna, Patna, India

***Abstract-*** As multiplication dominates the execution time of the most Digital Signal Processing algorithms, so there is a need of high speed multiplier. This paper presents the detailed study of different multipliers based on Array Multiplier, Constant coefficient multiplication (KCM) and multiplication based on vedic mathematics. All these multipliers are coded in Verilog HDL (Hardware Description Language) and simulated in ModelSimXEIII6.4b and synthesized in EDA tool Xilinx_ISE12. All multipliers are then compared based on LUTs (Look up table) and path delays. Results show that Vedic Urdhva Tiryakbhyam sutra is the fastest Multiplier with least path delay.

***Index Terms***- Array Multiplier, Constant Coefficient Multiplier (KCM), Vedic Mathematics, Urdhva Tiryakbhyam sutra.

## I. Introduction

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate(MAC) and inner product are among some of the frequently used Computation Intensive Arithmetic Functions(CIAF) currently implemented in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform(FFT), filtering and in ALU of microprocessors. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. Multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip.

The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications [2]. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications [2, 3]. This paper presents the study of different multipliers. Multiplier based on Vedic Mathematics is the fastest multiplier.

Digital multipliers are the most commonly used components in any digital circuit design. They are fast, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available. Particular multiplier architecture is chosen based on the application.

In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of the algorithm. The speed of multiplication operation is of great importance in DSP as well as in general processor. In the past, multiplication was implemented generally with a sequence of addition, subtraction and shift operations. There are many algorithms proposed in literature to perform multiplication, each offering different advantages and having trade off in terms of speed, circuit complexity, area and power consumption.

Different multipliers are discussed in below sections.

## II. ARRAY MULTIPLIER

An array multiplier is a digital combinational circuit that is used for the multiplication of two binary numbers by employing an array of full adders and half adders. This array is used for the nearly simultaneous addition of the various product terms involved. To form the various product terms, an array of AND gates is used before the Adder array.

To clarify more on the concept, let us consider a 2X2 bit multiplication with A and B being the multiplicand and the multiplier respectively. Assuming A = a(1)a(0) and B= b(1)b(0), the various bits of the final product term P can be written as:-

P(0)= a(0)b(0)
P(1)=a(1)b(0) + b(1)a(0)
P(2) = a(1)b(1) + C1 where C1 is the carry generated during the addition for the P(1) term.
P(3)=C2 where C2 is the carry generated during the addition for the P(2) term.

For the above multiplication, an array of four AND gates is required to form the various product terms like a(0)b(0) etc. and then an Adder array is required to calculate the sums involving the various product terms and carry combinations mentioned in the above equations in order to get the final Product bits.

The Hardware requirement for an m x n bit array multiplier is given as:-
(m x n) AND gates,
(m-1).n Adders in which n HA(Half Adders) and
(m-2).n FA(full adders).

Example 1 describes the multiplication process using array multiplier. Instead of Ripple Carry Adder (RCA), here Carry Save Adder (CSA) is used for adding each group of partial product terms, because RCA is the slowest adder among all other types of adders available. In case of multiplier with CSA, partial product addition is carried out in Carry save form and RCA is used only in final addition.

Example 1: (1 0 0 1 x 1 0 1 0) = 1011010

```
1 0 0 1
     x1 0 1 0
0 0 0 0
   1 0 0 1   ← Left Shift by 1 bit
  0 0 0 0    ← Left Shift by 2 bit
 1 0 0 1     ← Left Shift by 3 bit
 1 0 1 1 0 1 0
```

Here from the above example it is inferred that partial products are generated sequentially, which reduces the speed of the multiplier. However the structure of the multiplier is regular.

Consider 4x4 multiplications, say A= $A_3$ $A_2$ $A_1$ $A_0$ and B= $B_3$ $B_2$ $B_1$ $B_0$. The output line for this multiplication is $P_7$ $P_6$ $P_5P_4P_3P_2$ $P_1$ $P_0$. Using the fundamental of Array Multiplication, taking partial product addition is carried out in Carry save form; we can have the following structure for multiplication as shown in Fig. 1.

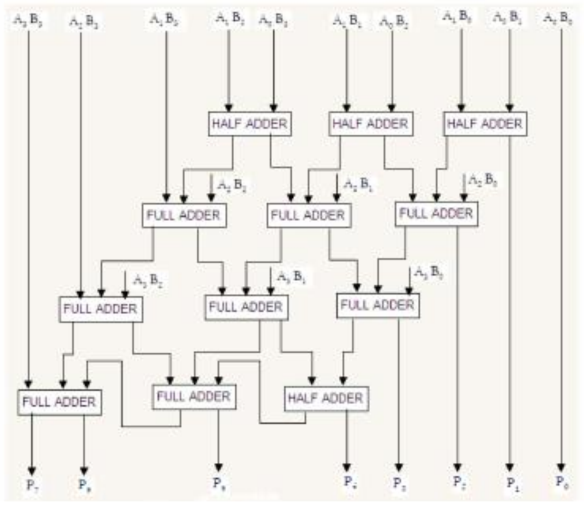


**Fig. 1. A 4-Bit Array multiplier.**

## III. CONSTANT COEFFICINET MULTIPLIER

This method is based on ROM approach. In conventional KCM, one input is fixed but here, both the inputs for the multiplier can be variables. In this method, a ROM is used for storing the squares of numbers as compared to KCM where the multiples are stored.

The sample ROM content is given in Table 1.

**Table 1**

| Address | Content(Square) |
|---|---|
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |
| … | … |

**Fig.2: A sample ROM content of KCM.**

**Method:**

To find (a x b), first we have to find whether the difference between 'a' and 'b' is odd or even. Based on the difference, the product is calculated using equations (1) and (2).

**i. In case of Even Difference**

Result of Multiplication= $[\text{Average}]^2$- $[\text{Deviation}]^2$ ….... (1)

**ii. In case of Odd Difference**

Result of Multiplication = [Average x (Average + 1)] - [Deviation x (Deviation+ 1)] …….(2)

Where, Average = [(a+b)/2] and Deviation = [Average - smallest( a, b)]

Example 2 (Even difference) and Example 3 (Odd difference) depict the multiplication process. Thus the two variable multiplication is performedby averaging, squaring and subtraction. To find the average[(a+b )/2], which involves division by 2 is performed by right shifting the sum by one bit. If the squares of the numbers are stored in a ROM, the result can be instantaneously calculated.

However, in case of Odd difference, the process is different as the average is a floating point number. In order to handle floating point arithmetic, Ekadikena Purvena - the Vedic Sutra which is used to find the square of numbers end with 5 is applied. Example 4 illustrates this. In this case, instead of squaring the average and deviation, [Average x (Average + 1)] - [Deviation x (Deviation+ 1)] is used. However, instead of performing the multiplications, the same ROM is used and using equation (3) the result of multiplication is obtained.

$n(n+l) = (n^2+n)$ ... (3)

Here $n^2$ is obtained from the ROM and is added with the address which is equal to n(n+l). The sample ROM contents are given in Table 1.

Thus, division and multiplication operations are effectively converted to subtraction and addition operations using Vedic Maths. Square of both Average and Deviation is read out simultaneously by using a two port memory to reduce memory access time.

**Example 2: 16 x 14 = 224**

1) Find the difference between (16-14) = 2 , Even Number
2) For Even Difference, Product = $[\text{Average}]^2$- $[\text{Deviation}]^2$
i. Average = [(a+b)/2] = [(16+14)/2] = [30/2] = 15
ii. Smallest(a,b) = smallest(l6,14) =14
iii. Deviation = Average - Smallest (a,b) =15 -14=1
3) Product = $15^2$- $1^2$= 225 - 1 = 224

**Example 3: 15 x 12 = 180**

1) Find the difference between (15-12)=3, Odd Number
2) For Odd Number Difference find the Average and Deviation.
i.Average = [(a+b)/2] = [(12+15)/2] = 13.5
ii.Deviation = [Average - smallest(a, b)] =
[12.5 - smallest(l3,12)] = [13.5 - 12] = 1.5
3)Product = (l3x14) - (lx2) = 182 - 2 =180

**Example 5: $25^2$= 625**

1) To find the square of 25, first find the square of 5 which is 25 and put 2 in the tens place and 5 inthe ones place of the answer respectively.

2) To find the number in the hundreds place, multiply 2 by its immediate next number, 3, which is equal to (2x3) = 6

3) Answer $25^2$= 625

## IV. VEDIC MATHEMATICS

Vedic mathematics - a gift given to this world by the ancient sages of India. A system which is far simpler and more enjoyable than modern mathematics. The simplicity of Vedic Mathematics means that calculations can be carried out mentally though the methods can also be written down. There are many advantages in using a flexible, mental system. Pupils can invent their own methods, they are not limited to one method. This leads to more creative, interested and intelligent pupils. Vedic Mathematics refers to the technique of Calculations based on a set of 16 Sutras, or aphorisms, as algorithms and their upa-sutras or corollaries derived from these Sutras. Any mathematical problems (algebra, arithmetic, geometry or trigonometry) can be solved mentally with these sutras. Vedic Mathematics is more coherent than modern mathematics.

Vedic Mathematics offers a fresh and highly efficient approach to mathematics covering a wide range - starts with elementary multiplication and concludes with a relatively advanced topic, the solution of non-linear partial differential equations. But the Vedic scheme is not simply a collection of rapid methods; it is a system, a unified approach. Vedic Mathematics extensively exploits the properties of numbers in every practical application.

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It covers explanation of several modern mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus.

His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That's why Vedic Mathematics has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristic, Vedic Mathematics has already crossed the boundaries of India and has become a leading topic of research abroad. Vedic Mathematics deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful [4, 9].

The word Vedic is derived from the word "veda" which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc.

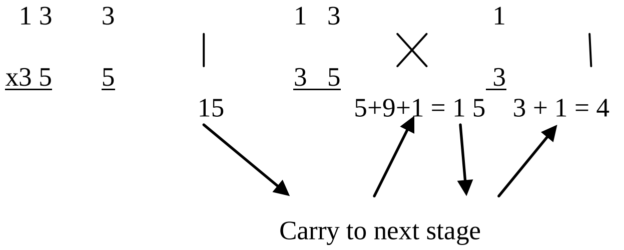
### Vedic Urdhva Tiryabhyam Multiplier

The multiplier is based on an algorithm Urdhva Tiryakbhyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra 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 parallelism in generation of partial products and their summation is obtained using Urdhva Triyakbhyam explained in next chapter. 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. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency. The net advantage is that it reduces the need of microprocessors to operate at increasingly high clock frequencies. While a higher clock frequency generally results in increased processing power, its disadvantage is that it also increases power dissipation which results in higher device operating temperatures.

By adopting the Vedic multiplier, microprocessors 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 has the advantage that as the number of bits increases, gate delay and area increases very slowly as compared to other multipliers. Therefore it is time, space and power efficient. It is demonstrated that this architecture is quite efficient in terms of silicon area/speed [10, 4].

**Multiplication of two digit decimal numbers an example- 13 x 35 = 455**



Answer = 13x35 = 455

The above discussions can now be extended to multiplication of binary number system with the preliminary knowledge that the multiplication of two bits $\mathbf{a_0}$ and $\mathbf{b_0}$ is just an AND operation and can be implemented using simple AND gate. To illustrate this multiplication scheme in binary number system, consider the multiplication of two binary numbers $\mathbf{a_3a_2a_la_0}$ and $\mathbf{b_3b_2b_lb_0}$. As the result of this multiplication would be more than 4 bits, the product is expressed as $\mathbf{r_7r_6r_5r_4r_3r_2r_1r_0}$. Least significant bit $r_0$ is obtained by multiplying the least significant bits of the multiplicand and the multiplier as shown in the Fig.2. The digits on 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 ($r_n$) and a carry ($C_n$). This carry is added in the next step and thus 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 the other entire bits act as carry. For example, if in some intermediate step, we get **110**, then **0** will act as result bit and **11** as the carry (referred to as $C_n$ in this text). It should be clearly noted that $C_n$ may be a multi-bit number.

Thus the following expressions (1) to (7) are derived:

$r_0 = a_0 b_0$ …(1)
$c_1 r_1 = a_1 b_0 + a_0 b_1$ …(2)
$c_2 r_2 = c_1 + a_2 b_0 + a_1 b_1 + a_0 b_2$ …(3)
$c_3 r_3 = c_2 + a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3$ …(4)
$c_4 r_4 = c_3 + a_3 b_1 + a_2 b_2 + a_1 b_3$ …(5)
$c_5 r_5 = c_4 + a_3 b_2 + a_2 b_3$ …(6)
$c_6 r_6 = c_5 + a_3 b_3$ …(7)

Here $c_6\mathbf{r_6r_5r_4r_3r_2r_1r_0}$is the final product. Partial products are calculated in parallel and hence the delay involved is just the time it takes for the signal to propagate through the gates.

The main advantage of the Vedic Multiplication algorithm (Urdhva Tiryakbhyam Sutra) stems from the fact that it can be easily implemented in FPGA due to its simplicity and regularity [3]. The digital hardware realization of a 4-bit multiplier using this Sutra is shown in next chapter. This hardware design is very similar to that of the array multiplier where an array of adders is required to arrive at the final product. Here in Urdhva, all the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders.

## V. IMPLEMENTATION OF VEDIC URDHVA TIRYAKBHYAM MULTILPIERS

### *A. 2x2 Bit Vedic Urdhva Tiryakbham Multiplier*

Consider 2 Bit number ‘a1a0’ and ‘b1 b0’ to be calculated. Using Urdhva sutra, a line diagram may be drawn as:

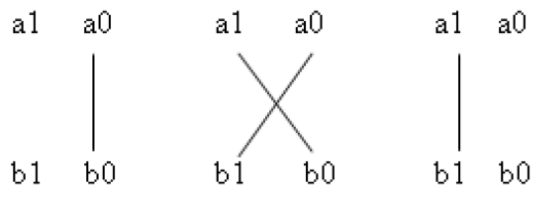


**Fig 4. Line Diagram for 2x2 Bit binary multiplication usin Urdhva Tiryakbhyam Sutra.**

The 2X2 Vedic multiplier module is implemented using four input AND gates & two half-adders which is displayed in its block diagram in Fig. 5.

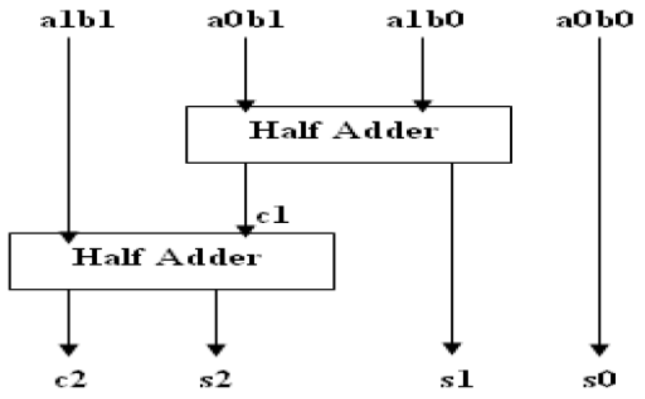


**Fig 5. 2x2 Vedic Urdhva Tiryakbhyam multiplier.**

### *B. 4x4 Bit Vedic Urdhva Tiryakbham Multiplier*

The 4x4 bit Vedic multiplier module is implemented using four 2x2 bit Vedic multiplier modules as discussed in Fig. 5. Let’s analyze 4x4 multiplications, say A= A3 A2 A1 A0 and B= B3 B2 B1 B0. The output line for the multiplication result is S7S6S5S4S3S2S1S0 .Let’s divide A and B into two parts, say A3 A2 & A1 A0 for A and B3 B2 & B1B0 for B. Using the fundamental of Vedic multiplication, taking two bit at a time and using 2 bit multiplier block, we can have the following structure for multiplication as shown in Fig. 6.

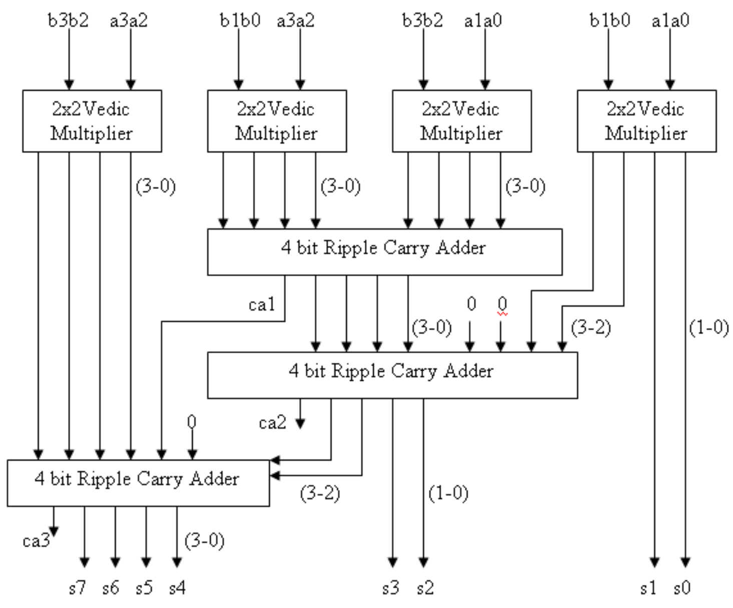


**Fig 8. Hardware implementation of 4x4 Bit Vedic Multiplier**

Also from the above explained method of multiplication using Urdhva Tiryakbhyam Multiplier for 4x4 Bit multiplication can also be implemented using expressions (1) to (7) as:

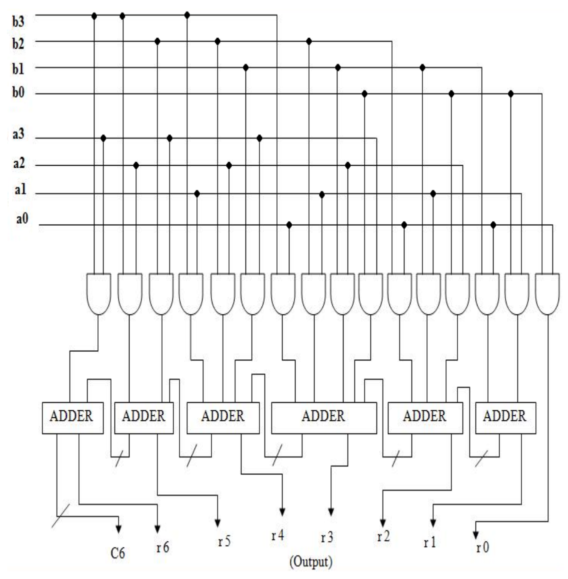


**Fig 7. Hardware implementation of 4x4 Bit Vedic Urdhva Tiryakbhyam Multiplier**

C. 8x8 Bit Vedic Urdhva Tiryakbhyam Multiplier

The 8x8 bit Vedic multiplier module as shown in the block diagram in Fig. 8. It 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 P = S15 S14 S13 S12 S11 S10 S9 S8 S7 S6S5S4 S3 S2 S1 S0. In this figure 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:

P=AxB= (AH-AL)x(BH-BL)
=AHxBH+AHxBL+ALxBH+ALxBL

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. Thus, in the final stage two adders are also required. Here four 4x4 Bit Vedic Multiplier and two carry save adder is used to implement the 8x8 Vedic Multiplier.

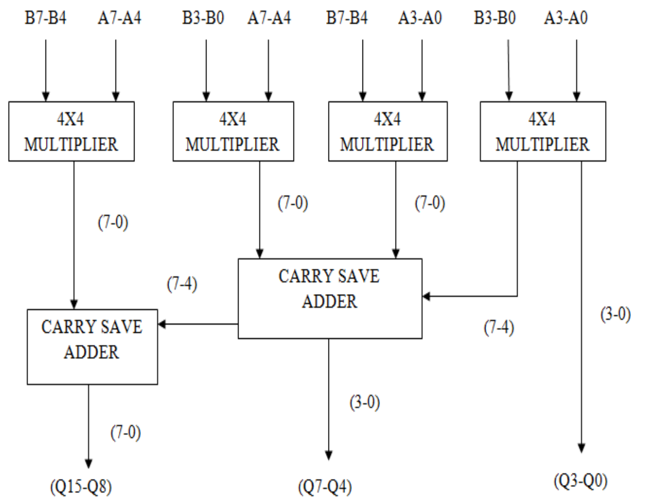


**Fig 9. Implementation of 8x8 Vedic Urdhva Tiryabhyam Multiplier.**

## VI. SIMULATIONS AND RESULTS

Different multipliers and proposed architecture process and the developed architecture for the required functionality were discussed in the previous chapters. Now this chapter deals with the simulation and synthesis results of the proposed architecture process. Here **ModelSimXEIII6.4b** tool is used in order to simulate the design and checks the functionality of the design. Vedic multiplier is designed in **Verilog HDL** (Hardware Description Language). Logic synthesis and simulation was done using EDA (Electronic Design Automation) tool in **XilinxISE12.1i .**

### A. Simulation Result of Array Multiplier

Simulation is tested for 8x8 bit Array multiplier for input as the multiplier 'a'="00110010" (decimal number system '50') and multiplicand 'b'="00010100" (decimal number system '20') and we get 16-bit output= "0000001111101000" (decimal number system 1000).

The screenshot of result is shown in fig. 9.

## B. Simulation Result of KCM

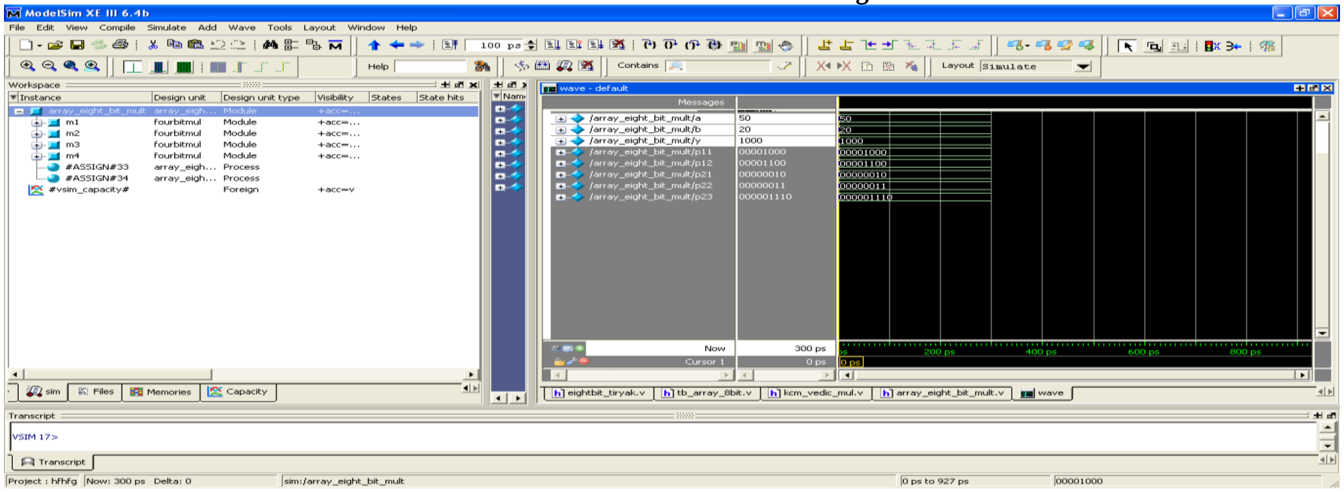
Simulation is tested for 8x8 bit KCM for input as the multiplier ‘a’=“00001010” (decimal number system ‘10’) and multiplicand ‘b’=“00001001” (decimal number system ‘9’) and we get 16-bit output= “0000000001011010” (decimal number system 90).

The screenshot of result is shown in fig. 10.

## C. Simulation Result of Vedic Urdhva Tiryakbhyam Multiplier

Simulation is tested for 8x8 bit bit Vedic Urdhva Tiryakbhyam multiplier input as the multiplier ‘a’=“00001100” (decimal number system ‘12’) and multiplicand ‘b’=“00000100” (decimal number system ‘4’) and we get 16-bit output= “0000000000110000” (decimal number system 48).

**The screenshot of result is shown in fig. 11.**



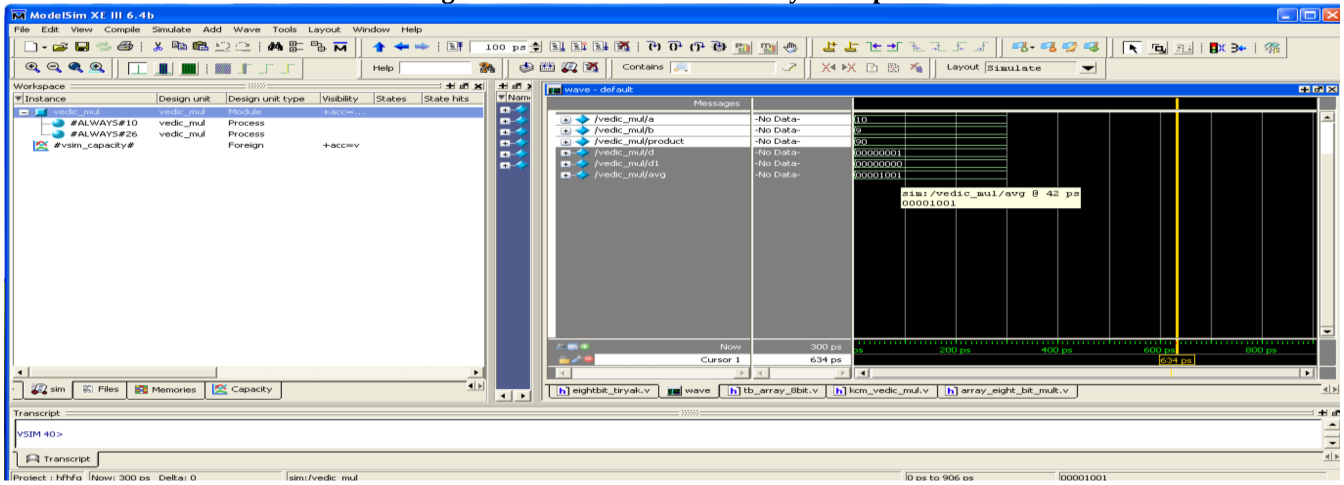
ModelSim XE III 6.4b

| Instance | Design unit | Design unit type | Visibility | States | State hits |
|---|---|---|---|---|---|
| array_eight_bit_mult | array_eigh... | Module | +acc=... | | |
| m1 | fourbitmul | Module | +acc=... | | |
| m2 | fourbitmul | Module | +acc=... | | |
| m3 | fourbitmul | Module | +acc=... | | |
| m4 | fourbitmul | Module | +acc=... | | |
| #ASSIGN#33 | array_eigh... | Process | | | |
| #ASSIGN#34 | array_eigh... | Process | | | |
| #vsim_capacity# | | Foreign | +acc=v | | |

```
wave - default
/array_eight_bit_mult/a     50
/array_eight_bit_mult/b     20
/array_eight_bit_mult/y     1000
/array_eight_bit_mult/p11   00001000
/array_eight_bit_mult/p12   00001100
/array_eight_bit_mult/p21   00000010
/array_eight_bit_mult/p22   00000011
/array_eight_bit_mult/p23   000001110
Now 300 ps
Cursor 1 0 ps
```

```
VSIM 17>
Project : hfhfg  Now: 300 ps  Delta: 0   sim:/array_eight_bit_mult   0 ps to 927 ps   00001000
```

**Fig 10. Simulation result of Array Multiplier**



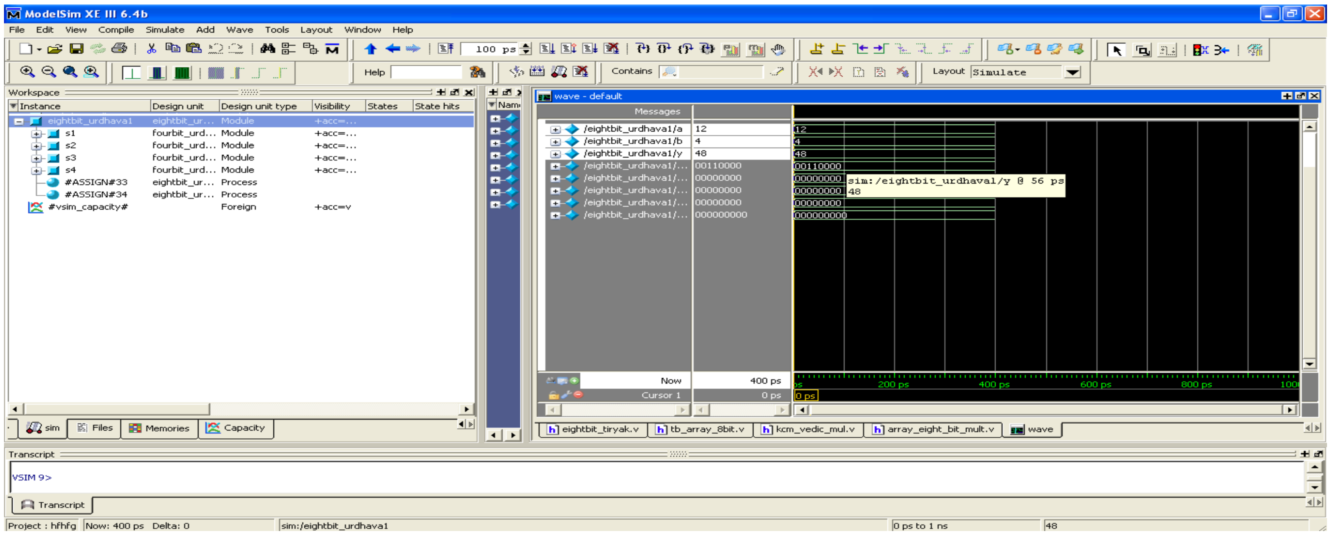
ModelSim XE III 6.4b

| Instance | Design unit | Design unit type | Visibility | States | State hits |
|---|---|---|---|---|---|
| vedic_mul | vedic_mul | Module | +acc=... | | |
| #ALWAYS#10 | vedic_mul | Process | | | |
| #ALWAYS#26 | vedic_mul | Process | | | |
| #vsim_capacity# | | Foreign | +acc=v | | |

```
wave - default
/vedic_mul/a        -No Data-   10
/vedic_mul/b        -No Data-   9
/vedic_mul/product  -No Data-   90
/vedic_mul/d        -No Data-   00000001
/vedic_mul/d1       -No Data-   00000000
/vedic_mul/avg      -No Data-   00001001
sim:/vedic_mul/avg @ 42 ps
00001001
Now 300 ps
Cursor 1 634 ps
```

```
VSIM 40>
Project : hfhfg  Now: 300 ps  Delta: 0   sim:/vedic_mul   0 ps to 906 ps   00001001
```

**Fig 10. Simulation result of KCM**



```
ModelSim XE III 6.4b
Workspace
Instance             Design unit       Design unit type  Visibility
eightbit_urdhava1    eightbit_ur...    Module            +acc=...
  s1                 fourbit_urd...    Module            +acc=...
  s2                 fourbit_urd...    Module            +acc=...
  s3                 fourbit_urd...    Module            +acc=...
  s4                 fourbit_urd...    Module            +acc=...
  #ASSIGN#33         eightbit_ur...    Process
  #ASSIGN#34         eightbit_ur...    Process
  #vsim_capacity#                      Foreign           +acc=v

wave - default
Messages
/eightbit_urdhava1/a      12          12
/eightbit_urdhava1/b      4           4
/eightbit_urdhava1/y      48          48
/eightbit_urdhava1/...    00110000    00110000
/eightbit_urdhava1/...    00000000    00000000   sim:/eightbit_urdhaval/y @ 56 ps
/eightbit_urdhava1/...    00000000    00000000   48
/eightbit_urdhava1/...    00000000    00000000
/eightbit_urdhava1/...    000000000   000000000
Now       400 ps
Cursor 1  0 ps

VSIM 9>

Project : hfhfg   Now: 400 ps   Delta: 0   sim:/eightbit_urdhava1   0 ps to 1 ns   48
```

**Fig 10. Simulation result of Vedic Urdhva Tiryakbhyam Multiplier**

## VII. DISCUSSION AND COMPARISON

All the three multipliers are synthesized in EDA (Electronic Design Automation) tool in XilinxISE12.1i software.
Two parameters LUTs and Path Delay are taken into considerations for the comparison of different multipliers.

**Table 2. Comparison of Multipliers**

| XilinxISE12.1i software | Array Multiplier | KCM | Vedic Urdhva Tiryakbhyam Multiplier |
|---|---|---|---|
| No. of LUTs used | 146 | 79 | 136 |
| Path delay (In Nanoseconds) | 18.792 | 22.864 | 17.995 |

Table 2 shows that Urdhva Tiryakbhyam Multiplier based on Vedic Mathematics uses only 136 LUTs which is very less as comparison to array Multiplier.

KCM uses only 79 LUTs but it uses ROM of 8KB space for an 8x8 Multiplier which is not desirable at all!

Table 2 shows that Urdhva Tiryakbhyam Multiplier takes **17.995** nanoseconds only which is the fastest of all other multipliers.

## VIII. CONCLUSION

The computational path delay for proposed 8x8 bit Vedic Urdhava Tiryakbhyam multiplier is found to be **17.995ns** as shown in Table 2 which is the least time delay of all the multipliers**.** Hence our motivation to reduce delay is finely fulfilled. Therefore, we observed that the Vedic Urdhava multiplier is much more efficient than Array and Constant Coefficient Multiplier (KCM) in terms of execution time (speed). Effective memory implementation and deployment of memory compression algorithms can yield even better results.

## REFERENCES

[1] 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.

[2] L. Sriraman, T. N. Prabakar, "Design and Implementation of Two Variable Multiplier Using KCM and Vedic Mathematics", 1" In!' I Conf. on Recent Advances in Information Technology,IEEE, I RAIT -2012 I

[3] 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.

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

[5] 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.

[6] 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 2011.

[7] 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 2010.

## AUTHORS

**First Author** – Mohammed Hasmat Ali, M.Tech, Electrical Engineering, National Institute of Technology Patna
Patna, India, e-mail: hashmatalig@gmail.com

**Second Author** – Anil Kumar Sahani, Assoc. Professor & Head, Electrical Engineering, National Institute of Technology Patna, Patna, India, e-mail: anilk_58@rediffmail.com