# Simulation of 23x23 Bit multiplication algorithm in vedic mathematics using verilog code

Ugra Mohan Kumar*, Devendra Singh and Pankaj Punia
*Department of Electronics & Communication*
*Uttaranchal University, Dehradun, India*
*ugramohan2007@gmail.com

## Abstract:

Multiplication is the most fundamental and commonly used operations in a CPU. These operations furthermore form the origin for other complex operations. With ever increasing requirement for faster clock frequency it becomes essential to have faster arithmetic unit. In this paper a new structure of Mathematics – Vedic Mathematics is used to execute operations. In this paper mainly two types of algorithms one is vedic and other one is vedic wallace algorithms are implemented for Multiplication in Verilog and performance is evaluated in Xilinx ISE Design Suite 13.2 platform then compared with different parameters like delay time and area (number of LUT) for several bits algorithms.

**Key words -**Vedic mathematics, multiplication, delay time, Verilog.

## Introduction

Multiplication is an important essential function in arithmetic operations. Multiplication-based different operations considered as Multiply and Accumulate (MAC) and internal product are among some of the commonly used Computation- Intensive Arithmetic Functions (CIAF) presently implemented and designed in many Digital Signal Processing (DSP) appliances considered as convolution of two or more than two information, Fast Fourier Transform (FFT) of different sequences, filtering of signals or information and in microprocessors its used in arithmetical and logical unit (ALU) (Chidgupkar and Karad, 2004). Since multiplying is the most important factor for the implementation time for most of the DSP algorithms or techniques, so there is a need of most efficient and high speed multiplier. Currently, multiplication time is still the most important factor in determining the instruction cycle time and the delay time of a DSP chip.

The requirement for high speed processing has been growing as a result of increasing work for computer and signal processing applications. Higher throughput arithmetical and logical operations are important to accomplish the required performance in various real-time signal and image processing applications (Thapliyal and Arabnia, 2004). The main key of arithmetical and logical operations in these applications is multiplication techniques and the development and designing of fast and efficient multiplier circuit has been a subject of interest over last few years. Sinking the execution time and power consumption of required circuits are very necessary requirements for various applications such as in digital signal processing and in digital image processing (Thapliyal and Arabnia, 2004; Abu-Shama *et al.*,1996). This work presents different multiplier techniques and architectures. Multiplier based on Vedic or ancient Mathematics is one of the fast and efficient with low propagation delay and low power consumption multiplier.

## Vedic Mathematics

Vedic Mathematics introduces the magnificent applications to Arithmetical calculation and verification, theory of numbers, complex multiplications, fundamental algebraic operations, complex factorizations, simple quadratic and advanced order equations, concurrent quadratic equations, partial fractions, in differential calculus and integral calculus, squaring of complex number, cubing, square root of complex number, cube root, 2-Dimensional and 3-Dimensional coordinate geometry and brilliant Vedic Numerical code (Tiwari *et al.,* 2008).

### (a) Vedic Mathematics Sutras and Up-sutras

Entire mechanics of Vedic mathematics is based on 16 sutras – formulas and 13 up-sutras meaning – corollaries.

*1. Ekadhikena Purvena*
*2. Nikhilam Navatascharamam Dashatah*
*3. Urdhva-tiryagbhyam*
*4. Paravartya Yojayet*
*5. Shunyam Samyasamucchaye*
*6. Anurupye Sunyamanyat*
*7. Sankalana vyavakalanabhyam*
*8. Puranaprranabhyam*
*9. Calana – Kalanabhyam*
*10. Yavadunam*
*11. Vyastisamashtih*
*12. Sheshanynkena Charmena*
*13. Sopantyadvayamantyam*
*14. Ekanyunena Purvena*
*15. Ginitasamucchayah*
*16. Gunaksamucchayah*

**Up-sutras**

*1. Anurupyena*
*2. Shishyate Sheshsamjnah*
*3. Adyamadye Nantyamantyena*
*4. Kevalaih Saptakam Gunyat*
*5. Vestanam*
*6. Yavadunam Tavadunam*
*7. Yavadunam Tavadunikutya Varganka ch Yojayet*
*8. Antyayordhshakepi*
*9. Antyatoreva*
*10. Samucchayagunitah*
*11. Lopanasthapanabhyam*
*12. Vilokanam*
*13.Gunitasamucchyah Samucchayagunitah*

### (b) Urdhva-tiryagbhyam:

The Nikhilam and Anurupyena are for special cases, whereas Urdhva-tiryagbhyam is general formula applicable to all. Its algebraic principle is based on multiplication of polynomials. Consider we want to multiply two 4th degree polynomials.

$$\begin{array}{c} Ax^4 + Bx^3 + Cx^2 + Dx + E \\ \underline{Zx^4 + Yx^3 + Xx^2 + Wx + V} \\ AZ\,x^8 + (AY+BZ)\,x^7 + (AX+BY+CZ)\,x^6 + (AW+BX+CY+DZ)\,x^5 + \\ (AV+BW+CX+DY+EZ)\,x^4 + (BV+CW+DX+EY)\,x^3 + (CV+DW+XE)\,x^2 + (DV+EW)\,x + \\ EV \end{array}$$

**Figure 1:** Multiplication of two fourth degree polynomials

Highest degree coefficient can be obtained by multiplication of two highest degree coefficients of individual polynomial namely A and Z. A next degree coefficient is obtained by addition of cross multiplication of coefficients of 4th degree and 3rd degree of other polynomials. It means A which is 4th degree coefficient of polynomial-1 is multiplied by 3rd degree coefficient of polynomial-2 is added to 4th degree coefficient of polynomial-2 multiplied by 3rd degree coefficient of polynomial-1 to get (AY+BZ).

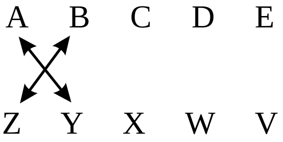


**Figure 2:** Vertically Crosswise First Cross Product

Similar logic of cross multiplication and addition can be extended till all 5 coefficients of both polynomials are used as follows. Every iteration gives a coefficient of product.

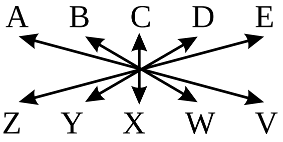


**Figure 3:** Vertically Crosswise Intermediate Cross Product

In this iteration, coefficient of degree 4 of product is obtained. For next iteration we drop A and Z which are the highest degree polynomial coefficients. The resulting operation gives coefficient of the degree 3 of multiplication of polynomials. As follows

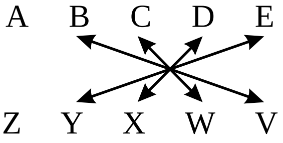


**Figure 4:** Vertically Crosswise Intermediate Cross Product

Continuing with this process last coefficient is obtained by multiplication of 0th degree terms of both polynomials as E*V. This process can be done is both ways as it is symmetric. In summary the process can be stated as, process of addition of product of coefficients of two polynomials in crosswise manner with increase and then decrease in number of coefficients from left to right with crosswise meaning product of coefficients for one polynomial going rightwards while for other leftwards.

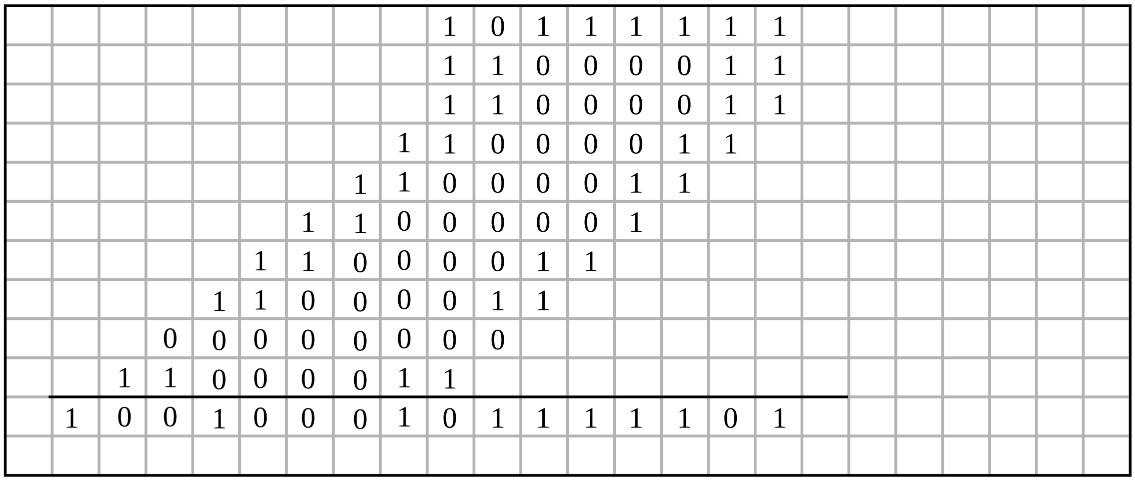
Any decimal number can be thought as a polynomial with unknown or x equal to 10. Being said that, formula stated above can be utilized to calculate product of two decimal numbers. Each digit of decimal number is though as coefficient of power of 10. Only restriction in this case is each cross product should be only one digit, if not it is added to the next power of 10.

## Proposed Algorithm

In binary system only 0 and 1 are used hence multiplication in Urdhva-tiryagbhyam or vertically-crosswise formula is replaced by AND logic. Each AND will be a bit wide and these bits are added together to generate cross-product. Rules for vertically-crosswise multiplication remains same as starting from MSBs – Most Significant Bit, of both multiplicands considered for first cross product. Then increasing one bit in each further calculation with cross product taken for bits of multiplicands till all bits are used (Mehta, *et al.*, 2009). Further dropping bits from MSB process of cross-product is continued till only LSB is used for cross-product. In binary number system the maximum width of cross-product depends on width of multiplicands. For example, in 8 bit multiplication maximum cross-product width will be $\log_2 8 + 1 = 4$. In 16 bit it will be 5 and in 23 bits it will be 5 again.

### (a) Comparison of Vedic and Conventional Multiplier

In this section comparison between Vedic multiplier by Urdhva-tiryagbhyam and conventional multiplier is made (Saha *et al.*, 2011). Conventional multiplier of width N x N will generate N number of partial products with each product containing N bits with 0 to 7 zeros added at the end. Vedic method generates (2N - 1) cross products of different widths which when combined forms (log2N + 1) partial products for N bit multiplier (Ramalatha *et al.*, 2009). In case of number of partial products there is significant decrease in number for Vedic Mathematics. But partial products generated in case of conventional multipliers are just by AND one multiplier by digits of another multiplier, whereas in case of Vedic, partial products are obtained after cross products are generated which requires some logic. Hence in Vedic mathematics delay for partial products is equal to adder delay. Critical path would consist of adders adding maximum number of bits in cross product. In all cases it will be the cross product in which all bits of multipliers are considered. Different techniques are used to combine these partial products efficiently to reduce the total time required for multiplication. One such technique Wallace tree addition is discussed in next section (Akhter, 2007).



| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | |
| | | | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | |
| | | | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | |
| | | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | |
| | | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | |
| | | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | |
| | | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | |
| | | | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | |
| | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | |

**Figure 5:** Conventional Cross Product Addition

### (b) Combining Partial Products

Combining partial products is the most critical part in any multiplier which decides the performance of it. Different types of adders like Carry Save adders (CSA) or Carry Look-ahead Adders (CLA) are used frequently. To further improve the performance more parallelism is achieved by combining three or more partial products at a time until two are left and then to add these two partial products to get the final answer. This technique is called as Wallace tree adder.

### (c) Vedic – Wallace

If Wallace tree structure with full and half adder is used then the class of multiplier is named as Vedic – Wallace multiplier. This type of multiplier essentially defers from conventional multiplier in the way the different bits are combined to form the final answer. 8, 16, and 23 bit multiplier are designed in Verilog with Vedic – Wallace structure with each of them customized to that particular width of multiplication to achieve high performance. These multipliers are designed as combinational blocks similar to equivalent Verilog design (Kumar, *et al.*, 2010). Each of these designed are synthesized with Xilinx in two ways 1. To achieve the highest possible clock frequency with which they work 2. Relaxed clock frequency to check the area generated.

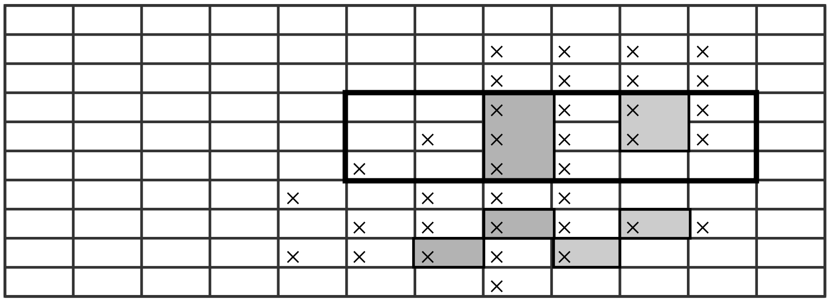


**Figure 6:** Wallace tree reduction

### (d) Analysis

Wallace tree addition makes use of full adder extensively for reducing partial products. Figure 8 shows a gate level diagram of a full adder. It can be observed that to generation of sum has to pass through two XOR gates and is inherently slower than generation of carry.

### Critical Path Analysis of a 4 bit multiplier

In this section comparison between critical path in 4 bit conventional and Vedic multiplier is done. In a 4 bit multiplier 4 partial products will be generated, in the diagram below named as P0 to P3.

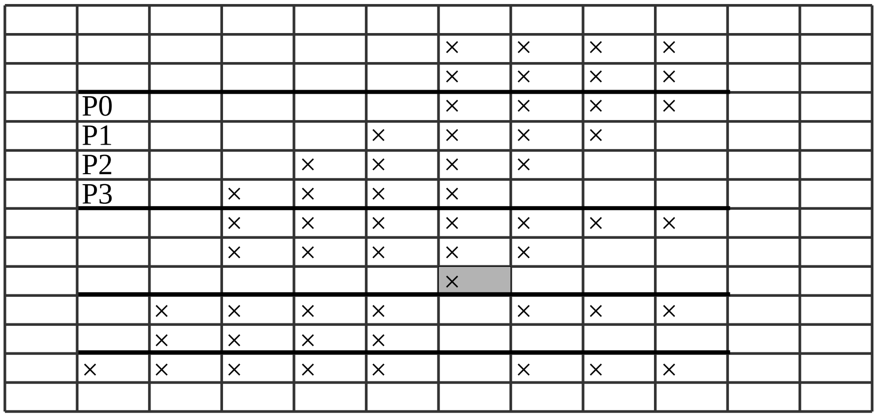


**Figure 7:** 4x4 Multiplication using Wallace tree Addition

In above multiplier a 3:2 reduction is used. After first level of Wallace tree 4 partial products are reduced to 3 partial products. As there are 4 bits to combine at power 23, in first level only 3 can be combined leaving fourth for next level. It is shown in blue background. This extra bit is combined in next level of reduction. After this level only 2 partial products are left and hence added together to get final answer. The block level diagram of the process is shown below with critical path marked in red. Delay in critical path in terms of half and full adder delays are as follows.

Delay = FAS + FAS + HAC + FAC + FAC + FAS

= 3 FAS + 2 FAC + HAC

Where FAS = Full Adder Carry, FAC = Full Adder Carry, HAC = Half Adder Carry,

HAS = Half Adder Sum, FA = Full Adder and HA = Half Adder.

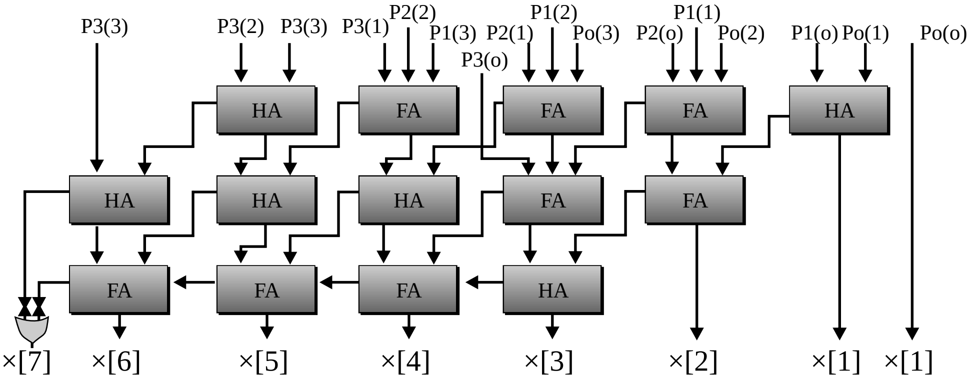


**Figure 8:** Block level structure of a 4x4 multiplier by Wallace tree reduction

In case of Vedic mathematics as bits at same power of two are added in one level. Hence 4 bits which are at power $2^3$ are added together to get a 3 bit sum. The MSB of this sum is at power $2^5$. If we compare the combination of partial product in Vedic and conventional the extra bit is at power 5 in Vedic compared to at power 3 for conventional. This carry forwarding is achieved with extra delay of a Half Adder Sum (HAS). Multiplication with Vedic structure is shown below. Figure 8 shows the critical path in red.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |
| | | | | | | × | × | × | × | | |
| | | | | | | × | × | × | × | | |
| | P0 | | | | | × | × | × | × | | |
| | P1 | | | | × | × | × | × | | | |
| | P2 | | | × | × | × | × | × | | | |
| | P3 | | × | × | × | × | × | × | | | |
| | | | × | × | × | × | × | × | × | | |
| | | | × | × | × | × | × | | | | |
| | | | | × | | | | | | | |
| | | × | × | × | × | × | × | × | × | | |
| | | × | × | × | × | × | | | | | |
| | | × | × | × | × | × | × | × | × | | |
| | | | | | | | | | | | |

**Figure 9:** Multiplication by Vedic Mathematics

Critical Delay

= HAS + HAC + HAS + HAS + HAS + FAC + FAC + FAS

= 2 FAC + FAS + 3HAS + HAC

Critical delay for conventional ??Critical delay for Vedic

3 FAS + 2 FAC + HAC ?? 2 FAC + FAS + 3 HAS + HAC

~~3~~ 2 FAS + ~~2 FAC + HAC ??~~ 2 FAC + ~~FAS + 3 HAS~~ + ~~HAC~~

2 FAS ?? 3 HAS

*4 XOR Delay* > *3 XOR Delay*

After comparing this with critical delay of conventional multiplier and cancelling equal terms we get 2 FAS – Full Adder Sums for conventional against 3 HAS - Half Adder Sums for Vedic. In terms of XOR gate this delay corresponds to 2×2 = 4 XOR gate delay for conventional against 3×1 = 3 XOR gate delay for Vedic multiplier. This analysis shows that 4 bit Vedic multiplier has less carry propagation delay than

Wallace tree multiplier. Also for 4 bit design there is no difference between Vedic-Vedic and Vedic-Wallace structures. Depending upon the number of bits in multiplication Vedic mathematics structure and Wallace tree structure critical path varies and hence there is variable improvement in Vedic-Vedic and Vedic-Wallace structure.

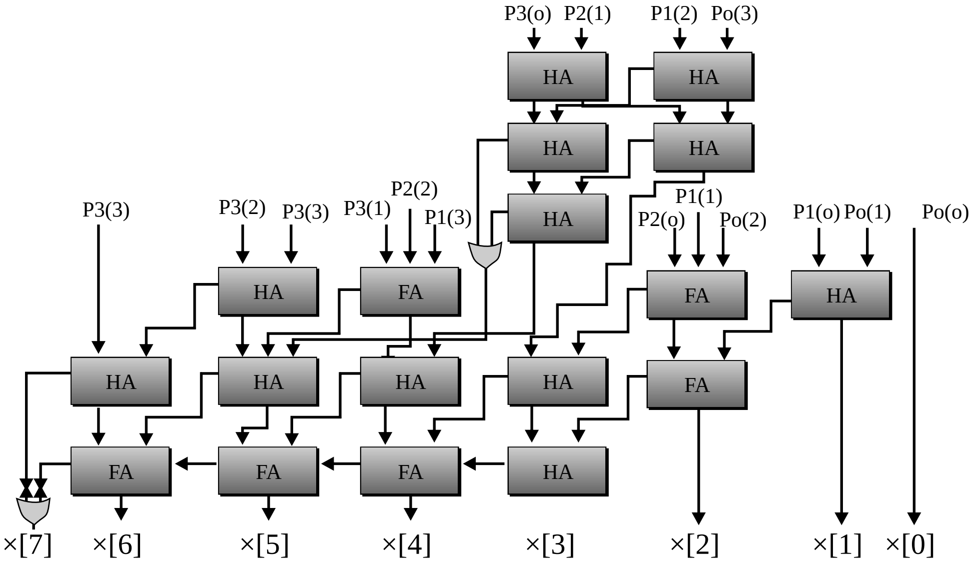
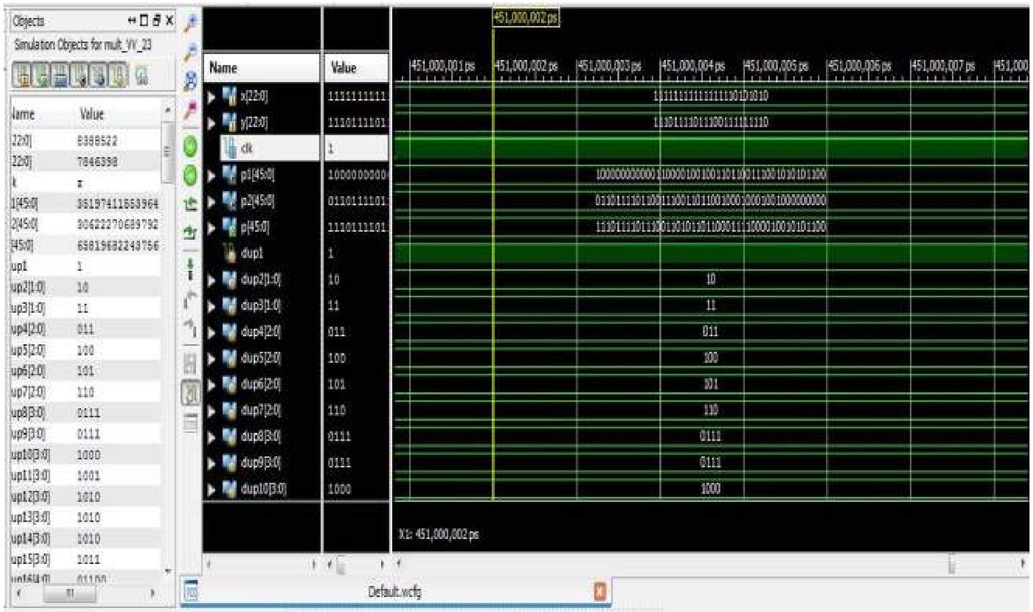


**Figure 10:** Block level structure of a 4x4 multiplier by Vedic reduction

## Results

### (a) Simulation result of 23x23 bit vedic multiplier:



| lame | Value |
|---|---|
| 22:0] | 8388522 |
| 22:0] | 7846398 |
| k | z |
| 1[45:0] | 35197411553964 |
| 2[45:0] | 90622270689792 |
| [45:0] | 65819682243756 |
| up1 | 1 |
| up2[1:0] | 10 |
| up3[1:0] | 11 |
| up4[2:0] | 011 |
| up5[2:0] | 100 |
| up6[2:0] | 101 |
| up7[2:0] | 110 |
| up8[3:0] | 0111 |
| up9[3:0] | 0111 |
| up10[3:0] | 1000 |
| up11[3:0] | 1001 |
| up12[3:0] | 1010 |
| up13[3:0] | 1010 |
| up14[3:0] | 1010 |
| up15[3:0] | 1011 |
| up16[4:0] | 01100 |

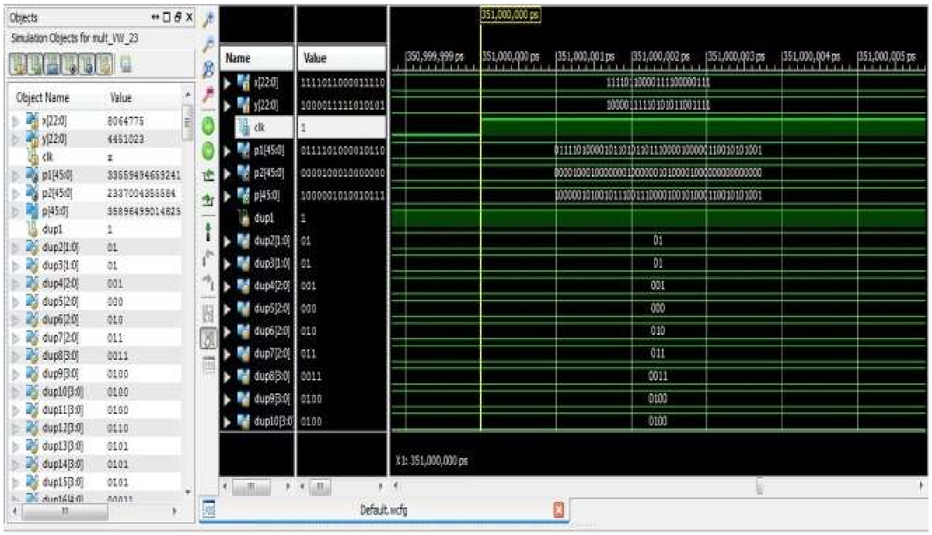
451,000,002 ps

| Name | Value | |
|---|---|---|
| x[22:0] | 1111111111 | 11111111111111111010010 |
| y[22:0] | 1110111101 | 11101111011100111111110 |
| clk | 1 | |
| p1[45:0] | 1000000000 | 1000000000000100001001001101100111001010101100 |
| p2[45:0] | 0110111101 | 0110111101100111001101100100010001001000000000 |
| p[45:0] | 1110111101 | 1110111101110011001011001000111100001001010101100 |
| dup1 | 1 | |
| dup2[1:0] | 10 | 10 |
| dup3[1:0] | 11 | 11 |
| dup4[2:0] | 011 | 011 |
| dup5[2:0] | 100 | 100 |
| dup6[2:0] | 101 | 101 |
| dup7[2:0] | 110 | 110 |
| dup8[3:0] | 0111 | 0111 |
| dup9[3:0] | 0111 | 0111 |
| dup10[3:0] | 1000 | 1000 |

451,000,001 ps 451,000,002 ps 451,000,003 ps 451,000,004 ps 451,000,005 ps 451,000,006 ps 451,000,007 ps 451,000

X1: 451,000,002 ps

**Figure 11:** Simulation result of 23X23 bits Vedic Multiplier



| Object Name | Value |
|---|---|
| x[22:0] | 8064775 |
| y[22:0] | 4461023 |
| clk | z |
| p1[45:0] | 35659494653241 |
| p2[45:0] | 2337004355584 |
| p[45:0] | 35896499014825 |
| dup1 | 1 |
| dup2[1:0] | 01 |
| dup3[1:0] | 01 |
| dup4[2:0] | 001 |
| dup5[2:0] | 000 |
| dup6[2:0] | 010 |
| dup7[2:0] | 011 |
| dup8[3:0] | 0011 |
| dup9[3:0] | 0100 |
| dup10[3:0] | 0100 |
| dup11[3:0] | 0100 |
| dup12[3:0] | 0110 |
| dup13[3:0] | 0101 |
| dup14[3:0] | 0101 |
| dup15[3:0] | 0101 |
| dup16[4:0] | 00011 |

| Name | Value |
|---|---|
| x[22:0] | 1111011000011110 |
| y[22:0] | 1000011111010101 |
| clk | 1 |
| p1[45:0] | 0111101000010110 |
| p2[45:0] | 0000100010000000 |
| p[45:0] | 1000001010010111 |
| dup1 | 1 |
| dup2[1:0] | 01 |
| dup3[1:0] | 01 |
| dup4[2:0] | 001 |
| dup5[2:0] | 000 |
| dup6[2:0] | 010 |
| dup7[2:0] | 011 |
| dup8[3:0] | 0011 |
| dup9[3:0] | 0100 |
| dup10[3:0] | 0100 |

**Figure 12:** Simulation result of 23X23 bits Wallace Multiplier

## Conclusion

The designs of 23x23 bits Vedic multiplier have been implemented on Spartan3E (3s500efg320-4) device. The computation delay for 23x23 bits Wallace multiplier is 23.249ns and 23x23 bits Vedic multiplier is obtained 19.113ns respectively. It is therefore seen that the Vedic multipliers are much faster than the conventional multipliers for higher order bits. The algorithms of Vedic mathematics are much more efficient than of conventional mathematics.

## References

Abu-Shama, E., Maaz, M.B., Bayoumi, M. A. 1996. A Fast and Low Power Multiplier Architecture", The Center for Advanced Computer Studies, The University of Southwestern Louisiana Lafayette, LA 70504, *Proceeding of 39th Midwest Symposium on Circuit and Systems,*1996.

Thapliyal, H., Arabnia, H.R. 2004. A Time-Area- Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics. Department of Computer Science, The University of Georgia, 415 *Graduate Studies Research Center Athens,* Georgia 30602-7404.

Tiwari, D., Gankhuyag, G., Chan, Mo Kim., Yong, Beom Cho. 2008. Multiplier design based on ancient Indian Vedic Mathematics. *International SoC Design Conference.*

Kumar, A., Raman, A., Sarin, R.K., Khosla, A. 2010. Small area Reconfigurable FFT Design by Vedic *Mathematics, IEEE.*

Mehta, P., Gawali, D. 2009. Conventional versus Vedic mathematical method for Hardware implementation of a multiplier. *International Conference on Advances in Computing, Control, and Telecommunication Technologies.*

Saha, A., Banerjee, P., Bhattacharyya, A., Dandapat. 2011. High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics. *Proceeding of the 2011 IEEE Students' Technology Symposium* 14-16 January, IIT Kharagpur.

Purushottam, D. Chidgupkar and Mangesh, T. Karad. 2004. The Implementation of Vedic Algorithms in Digital Signal Processing, *Global J. of Engineering. Education,* 8 (2), UICEE Published in Australia.

Ramalatha, M., Thanushkodi, K., Dayalan, K. D., Dharani, P. 2009. A Novel Time and Energy Efficient Cubing Circuit using Vedic Mathematics for Finite Field Arithmetic. *International Conference on Advances in Recent Technologies in Communication and Computing.*

Shamim, A. 2007. VHDL Implementation of Fast NxN Multiplier Based on Vedic Mathematics, *IEEE.*