Design & Implementation of Urdhva-Triyakbhyam based fast 16*16 Vedic Binary Multiplier

[1]Abhay Satmohankar, [2]Kalyani Doifode, [3]Akanksha Ramteke, [4]Ashok Bind
[1]Asst. Professor, [2]Scholar , [3] Scholar, [4] Scholar
[1]Department of Electronics and Telecommunication, [1]GNIET, Nagpur, India,
[2]Department of Electronics and Telecommunication, [2]GNIET, Nagpur, India,
[3]Department of Electronics and Telecommunication, [3]GNIET, Nagpur, India,
[4]Department of Electronics and Telecommunication, [4]GNIET, Nagpur, India,

Abstract : 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 microprocessors m its arithmetic and logic unit. Since multiplication dominates the execution tune of most DSP algorithms, so there is a need of high speed multiplier. Currently, 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 . 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. Multiplier based on Vedic Mathematics is one of the fast and low power multiplier. For arithmetic multiplication various Vedic multiplication techniques like Urdhva Tiryakbhyam, Nikhilam and Anurupye has been thoroughly analysed. Also Karatsuba algorithm for multiplication has been discussed. It has been found that Urdhva Tiryakbhyam Sutra is most efficient Sutra (Algorithm), giving minimum delay for multiplication of all types of numbers. Using Urdhva Tiryakbhyam, a 16x16 bit Multiplier has been designed and using this Multiplier, a Multiply Accumulate unit has been designed. Then, an Arithmetic module has been designed which employs these Vedic multiplier and MAC units for its operation. Logic verification of these modules has been done by using Modelsim 6.5.

***Index Terms* – MAC, MODELSIM, VEDIC**

I. INTRODUCTION

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 gives explanation of several 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 it has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristics, Vedic maths has already crossed the boundaries of India and has become an interesting topic of research abroad. Vedic maths deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful [2, 3].

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. These Sutras along with their brief meanings are enlisted below alphabetically.

1) (Anurupye) Shunyamanyat – If one is in ratio, the other is zero.
2) Chalana-Kalanabyham – Differences and Similarities.
3) Ekadhikina Purvena – By one more than the previous One.
4) Ekanyunena Purvena – By one less than the previous one.
5) Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.
6) Gunitasamuchyah – The product of the sum is equal to the sum of the product.
7) Nikhilam Navatashcaramam Dashatah – All from 9 and last from 10.
8) Paraavartya Yojayet – Transpose and adjust.
9) Puranapuranabyham – By the completion or noncompletion.
10) Sankalana- vyavakalanabhyam – By addition and by subtraction.
11) Shesanyankena Charamena – The remainders by the last digit.
12) Shunyam Saamyasamuccaye – When the sum is the same that sum is zero.
13) Sopaantyadvayamantyam – The ultimate and twice the penultimate.
14) Urdhva-tiryagbhyam – Vertically and crosswise.
15) Vyashtisamanstih – Part and Whole.
16) Yaavadunam – Whatever the extent of its deficiency.

These methods and ideas can be directly applied to trigonometry, plain and spherical geometry, conics, calculus (both differential and integral), and applied mathematics of various kinds. As mentioned earlier, all these Sutras were reconstructed from ancient Vedic texts early in the last century. Many Sub-sutras were also discovered at the same time, which are not discussed here. The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a 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. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing [1,4].

The multiplier architecture can be generally classified into three categories. First is the serial multiplier which emphasizes on hardware and minimum amount of chip area. Second is parallel multiplier (array and tree) which carries out high speed mathematical operations. But the drawback is the relatively larger chip area consumption. Third is serial- parallel multiplier which serves as a good trade-off between the times consuming serial multiplier and the area consuming parallel multipliers.

VEDIC MULTIPLICATION

The proposed Vedic multiplier is based on the Vedic multiplication formulae (Sutras). 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. Vedic multiplication based on some algorithms, is discussed below:

Urdhva Tiryakbhyam sutra

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 and then, concurrent addition of these partial products can be done. Thus parallelism in generation of partial products and their summation is obtained using Urdhava Tiryakbhyam. 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 [3,5].

*Multiplication of two decimal numbers- 43*68*

To illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (43*68). The digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one digit of result and a carry digit. 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, unit’s place digit acts as the result bit while the higher digits act as carry for the next step. Initially the carry is taken to be zero. The working of this algorithm has been illustrated in Fig 1.1

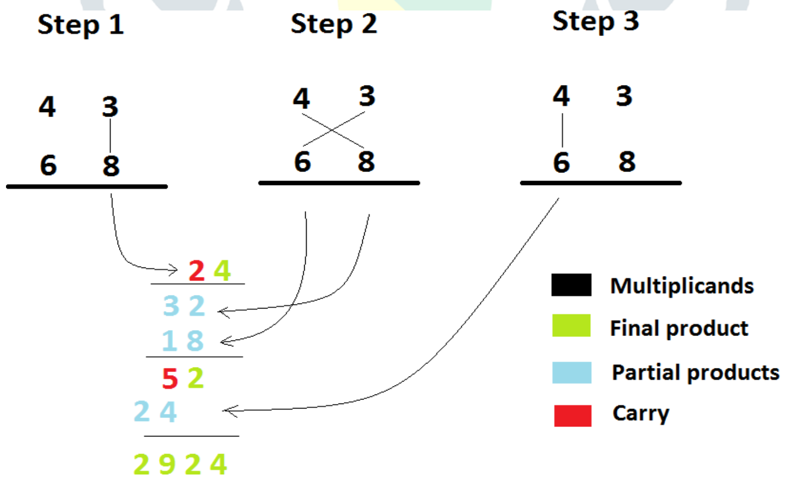


Fig 1.1 Multiplication of 2 digit decimal numbers using Urdhva Tiryakbhyam Sutra

Now we will see how this algorithm can be used for binary numbers. For example (1101 * 1010) as shown in Fig 1.2

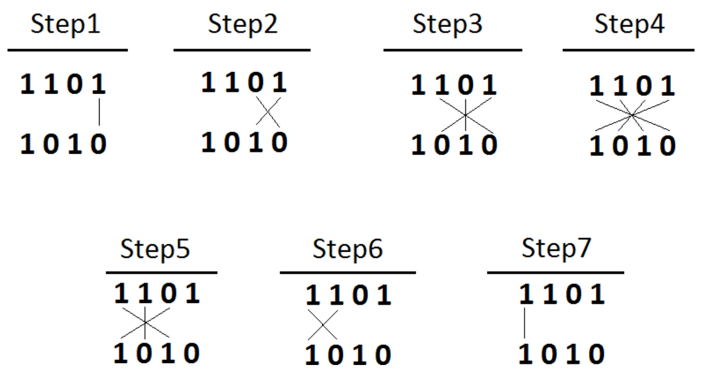


Fig 1.2 Using Urdhva Tiryakbham for Binary numbers

Firstly, least significant bits are multiplied which gives the least significant bit of the product (vertical). Then, the LSB of the multiplicand is multiplied with the next higher bit of the multiplier and added with the product of LSB of multiplier and next higher bit of the multiplicand (crosswise). The sum gives second bit of the product and the carry is added in 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. Next, all the four bits are processed with crosswise multiplication and addition to give the sum and carry. The sum 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. The same operation continues until the multiplication of the two MSBs to give the MSB of the product. For example, if in some intermediate step, we get 110, then 0 will act as result and 11 as the carry. It should be clearly noted that carry may be a multi-bit number.

II. RESEARCH METHODOLOGY

The 16x16 Multiplier is made by using 4 , 8x8 multiplier blocks. Here , the multiplicands are of bit size (n=16) where as the result is of 32 bit size. The input is broken into smaller chunks size of n/2 = 8, for both inputs, that is a and b. These newly formed chunks of 8 bits are given as input to 8x8 multiplier block, where again these new chunks are broken into even smaller chunks of size n/4 = 4 and fed to 4x4 multiply block, just as in case of 8x8 Multiply block. Again, the new chunks are divided in half, to get chunks of size 2, which are fed to 2x2 multiply block. The result produced, from output of 8x8 bit multiply block which is of 16 bits, are sent for addition to an addition tree , as shown in the Fig 2.1

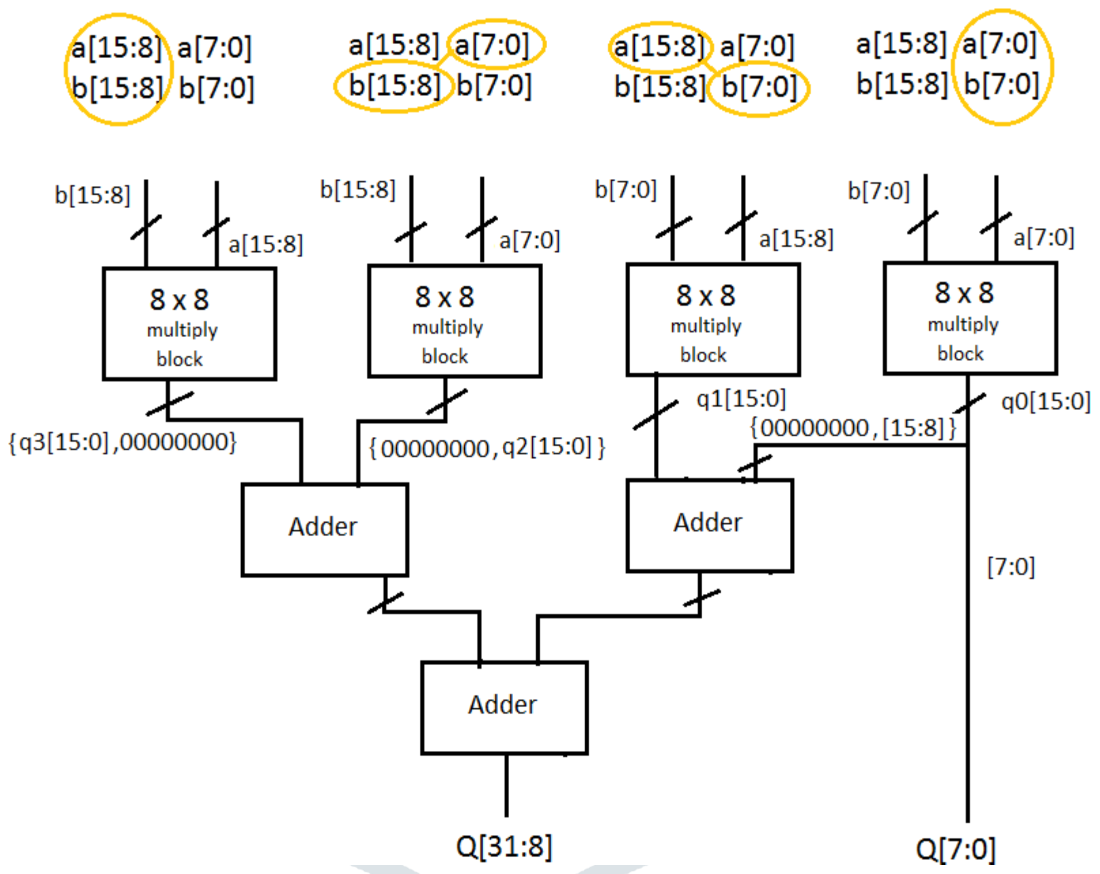


Fig2.1 Block diagram of 16x16 Multiply block

Here, as shown in Fig 16, the lower 8 bits of q0 directly pass on to the result, while the higher bits are fed for addition into the addition tree. The addition of partial products is shown in Fig.

16 BIT MAC UNIT USING 16X16 VEDIC MULTIPLIER

The 16 bit MAC unit here, uses a 16x16 vedic multiplier in its data path. As the 16x16 vedic multiplier is faster than other multipliers for example booth multiplier, even the MAC unit made using vedic multiplier is faster than the conventional one.

In the MAC unit, the data inputs, A and B are 16 bits wide and they are stored in two data registers, that is Data a_reg and Data b _reg, both being 16 bits wide, Then the inputs are fed into a vedic multiplier, which stores the result "Mult" in a 32 bit wide register named, Multiply_reg. The contents of Multiply_reg are continuously fed in to a conventional adder and the result is stored in a 64 bit wide register "Dataout_reg".

Here we notice that the MAC unit makes use of two clocks, one for the operation of MAC unit and other one, namely clk2 for the Multiplier. The frequency of clk2 should be 4 times the frequency of MAC unit for proper operation. A clock divider by 4 circuit may be used, in future here, which takes clk2 as the parent clock and produces 'clk' as the daughter clock, which is 4 times slower than the parent clock. But with 50% duty cycle. The faster clock 'clk2' is used for the multiplier while slower clock 'clk' is used for the MAC unit. The data coming as input to MAC unit may vary with clock 'clk'. The signal "clr" when applied, makes the contents of all the data registers that is Data a _reg, Data b_reg, Multipl _reg and Dataout_reg to be forced to 0. The "clken" signal is used to enable the MAC operation.

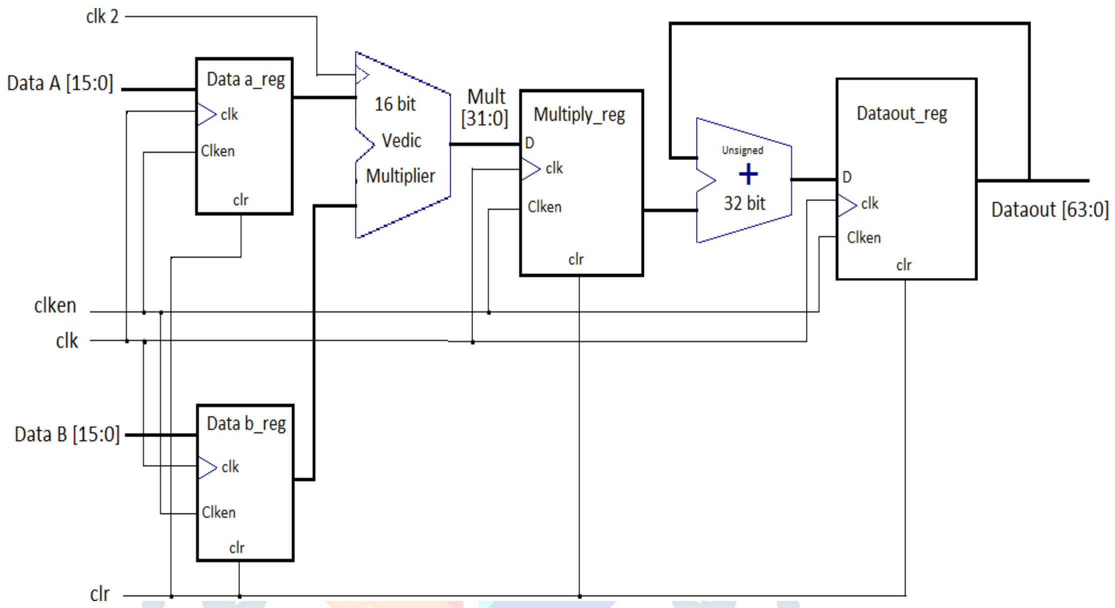


Fig 2.2 Block Diagram of 16 bit MAC unit

ADDER

In general, carry ripple adders can be used when they meet timing constraints because they are compact and easy to build. When faster adders are required, carry increment and carry skip architectures can be used, particularly for 8 to 16 bit lengths. Hybrids combining these techniques are also popular. At word lengths of 32 and especially 64 bits, tree adders are distinctly faster.

Good logic synthesis tools automatically map the "+" operator onto an appropriate adder to meet timing constraints while minimizing the area. For example, the Synopsys Designware libraries contain carry-ripple adders, carry select adders, carry lookahead adders and a variety of prefix adders.[12]

ARITHMETIC MODULE

The Arithmetic module designed in this work, makes use of 4 components, that are, Adder, Subtractor, Multiplier and MAC unit. As a result, the Arithmetic unit can perform fixed point addition, subtraction, multiplication and multiply accumulate operations on 16 bit data.Here the inputs are Data a and Data b¸ which are 16 bits wide. The arithmetic unit uses conventional adder and subtractor, while the multiplier and MAC unit are made using Vedic Mathematics Algorithm. The control signals which guide the Arithmetic unit to perform a particular operation, ie Addition, subtraction, multiplication or MAC operation are s0 and s1, which are provided by the control circuit. Control circuit is beyond the scope of this thesis. The clock used by multiplier,ie "clk2" runs 4 times as fast as the global clock "clk"which is used for controlling MAC operation. The details of MAC operation and Multiplier have already been discussed in previous sections. Now lets have a look at the status of control lines s0 and s1 and the corresponding arithmetic operation being performed.

S1	S0	Operation performed
0	0	Addition
0	1	Subtraction
1	0	Multiplication
1	1	Multiply Accumulate

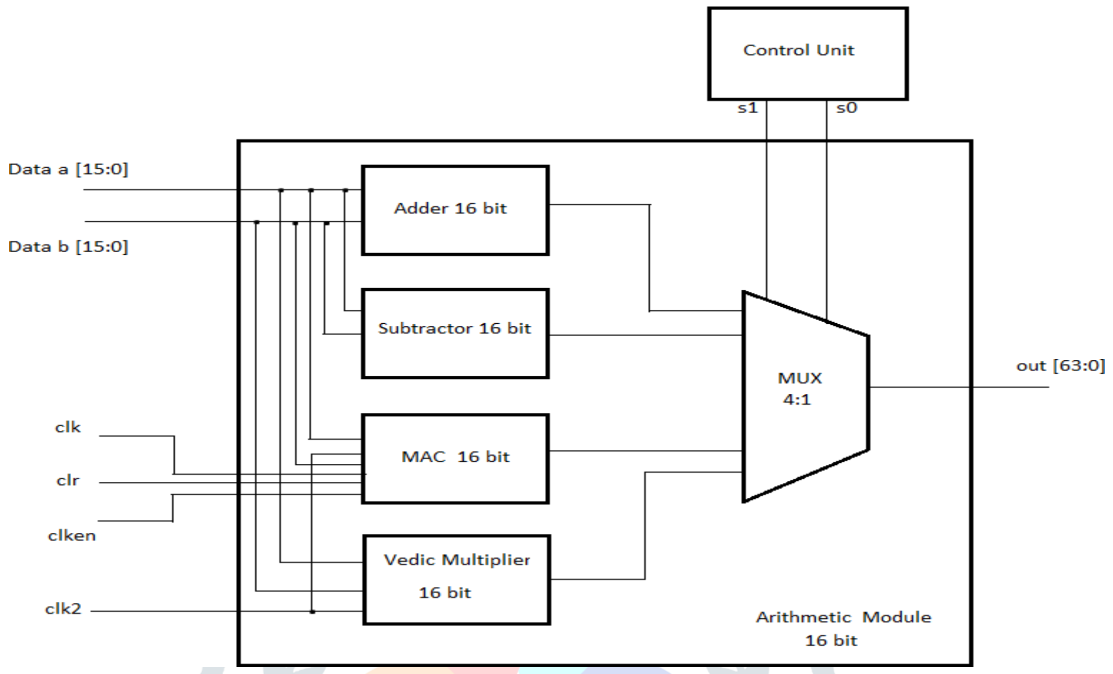
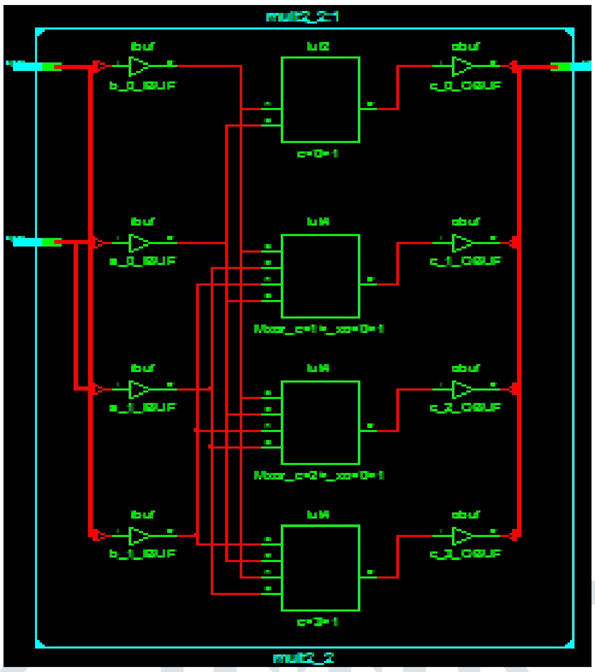


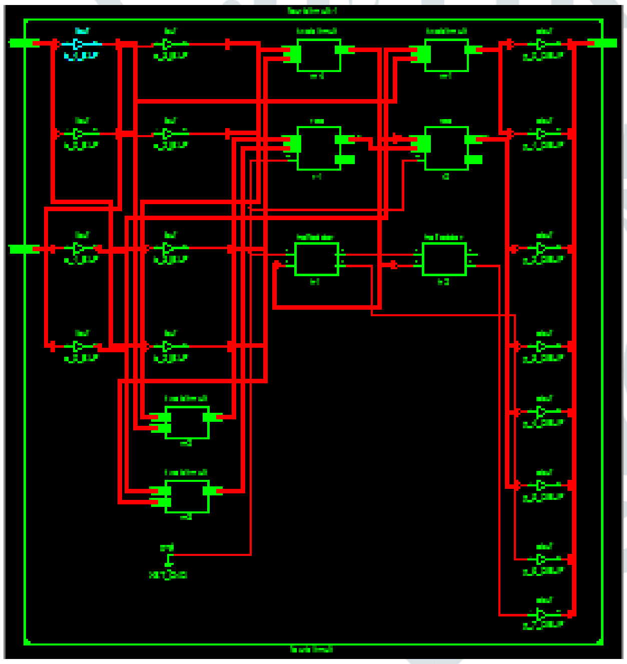
Fig 2.3 Block Diagram of Arithmetic module

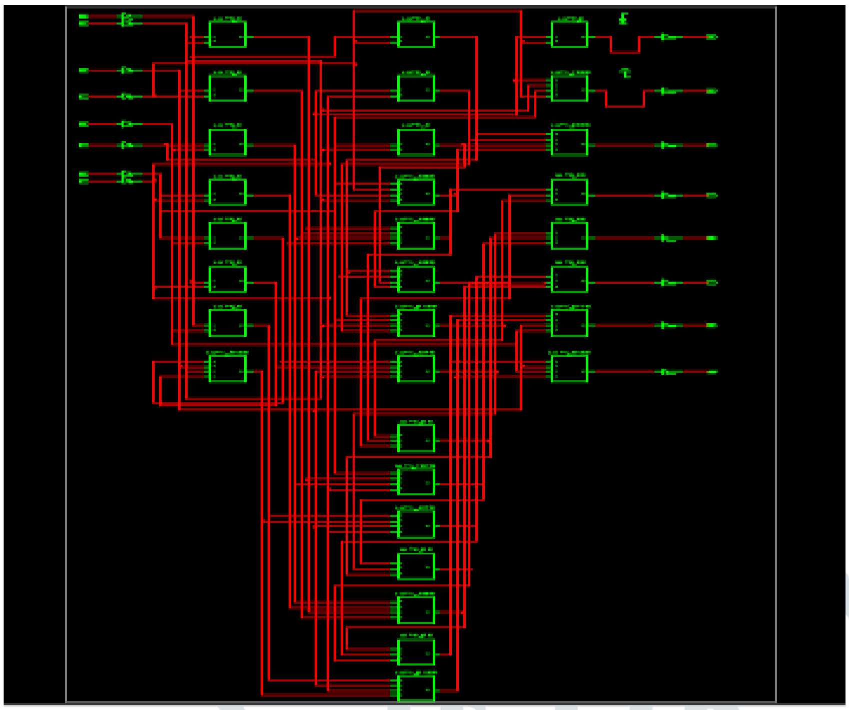
III. RESULTS AND DISCUSSION

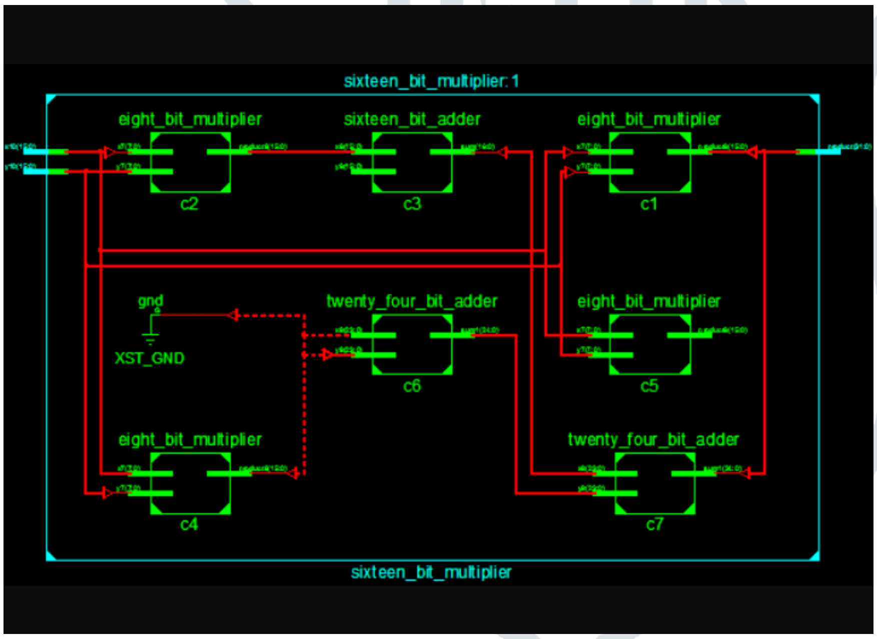
In this project architecture is designed by Verilog HDL, this is simulated by using Modelsim6.4b, verifiy the functionality. We have verified the functionality for 32 bit. Below fig shows the addition of two 32 bit numbers. The developed project is simulated and verified their functionality. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. Here in this Spartan 6 family, many different devices were available in the Xilinx ISE tool. In order to synthesis this design the device named as “XC3S500E” has been chosen and the package as “FG320” with the device speed such as “-4”. This design is synthesized and its results were analyzed as follows.

RTL Schematic:

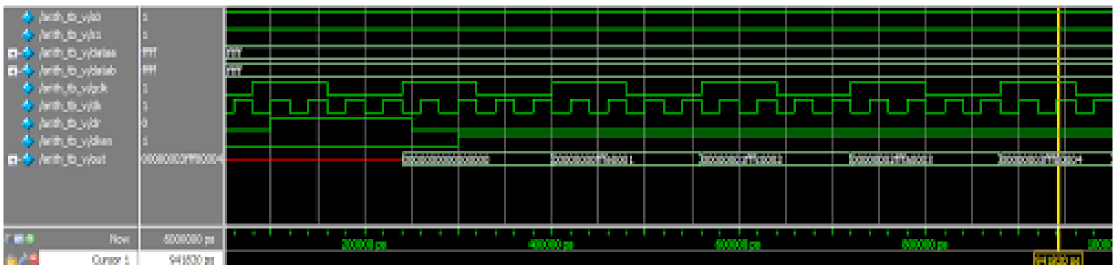

mult2_2:1
ibuf
lut2
obuf
b_0_IBUF
c_0_OBUF
c<0>1
ibuf
lut4
obuf
a_0_IBUF
c_1_OBUF
Mxor_c<1>_xo<0>1
ibuf
lut4
obuf
a_1_IBUF
c_2_OBUF
Mxor_c<2>_xo<0>1
ibuf
lut4
obuf
b_1_IBUF
c_3_OBUF
c<3>1
mult2_2







Output waveform



IV. REFERENCES

[1] www.en.wikipedia.com

[2] Jagadguru Swami Sri Bharati Krishna Tirthji Maharaja,"Vedic Mathematics", Motilal Banarsidas, Varanasi, India, 1986.

[3] 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.org Spring 2008.

[4] Shripad Kulkarni, "Discrete Fourier Transform (DFT) by using Vedic Mathematics", report, vedicmathsindia.blogspot.com, 2007.

[5] Himanshu Thapliyal, Saurabh Kotiyal and M. B Srinivas, "Design and Analysis of A Novel Parallel Square and Cube Architecture Based On Ancient Indian Vedic Mathematics", Centre for VLSI and Embedded System Technologies, International Institute of Information Technology, Hyderabad, 500019, India, 2005 IEEE

[6] Shamim Akhter, "VHDL Implementation of Fast NXN Multiplier Based on Vedic Mathematics",

Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE

[7] Himanshu Thapliyal and M.B Srinivas, "An Efficient Method of Elliptic Curve Encryption Using Ancient Indian Vedic Mathematics", IEEE, 2005.

[8] El hadj youssef wajih, Zeghid Medien, Machhout Mohsen, Bouallegue Belgacem, Tourki Rached,"Efficient Hardware Architecture of Recursive Karatsuba-Ofman Multiplier", 2008 International Conference on Design & Technology of Integrated Systems in Nanoscale Era

[9] www.mathworld.wolfram.com/KaratsubaMultiplication.html

[10] Abhijeet Kumar, Dilip Kumar, Siddhi, "Hardware Implementation of 16*16 bit Multiplier.