SIMULATION FOR VEDIC MULTIPLIERS FOR HIGH SPEED LOW POWER OPERATIONS WITH HDL

Thallapalli Saibaba[1], Ritafaria D[2]
Assistant Professor, CJITS, Janagoan, T.S, India

ABSTRACT

Performance of all microprocessors and microcontrollers it is depend on the multipliers. Hence better multiplier architectures are bound to increase the efficiency of the system. Vedic multiplier is one such promising solution. Its simple architecture coupled with increased speed forms an unparalleled combination for serving any complex multiplication computations. Power dissipation is another important constraint in an embedded system which cannot be neglected. In this paper we bring out a Vedic multiplier known as " Urdhva Tiryakbhayam multiplier". The Urdhva Tiryakbhayam literally means —.This will be implemented using reversible logic with HDl simulation . This multiplier may find applications in DSP like imaging, software defined radios, wireless communications. The purpose of this project is to implement a 8x8 vedic multiplier which are operated at very high speed. The functionality of RT is verified by using Mentor Graphics Tools and implemented on Spartan-3E FPGA kit.

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 microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time 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. Reducing the time delay and power consumption are very essential requirements for many applications. This work presents different multiplier architectures. Multiplier based on Vedic Mathematics is one of the fast and low power multiplier.

Minimizing power consumption for digital systems involves optimization at all levels of the design. This optimization includes the technology used to implement the digital circuits, the circuit style and topology, the architecture for implementing the circuits and at the highest level the algorithms that are being implemented. 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 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 have been many algorithms proposals in literature to perform multiplication, each offering different advantages and having tradeoff in terms of speed, circuit complexity, area and power consumption.

The multiplier is a fairly large block of a computing system. The amount of circuitry involved is directly proportional to the square of its resolution i.e. A multiplier of size n bits has n^2 gates. For multiplication algorithms performed in DSP applications latency and throughput are the two major concerns from delay perspective. Latency is the real delay of computing a function, a measure of how long the inputs to a device are stable is the final result available on outputs. Throughput is the measure of how many multiplications can be performed in a given period of time; multiplier is not only a high delay block but also a major source of power dissipation. That's why if one also aims to minimize power consumption, it is of great interest to reduce the delay by using various delay optimizations.

Digital multipliers are the core components of all the digital signal processors (DSPs) and the speed of the DSP is largely determined by the speed of its multipliers. Two most common multiplication algorithms followed in the digital hardware are array multiplication algorithm and Booth multiplication algorithm. The computation time taken by the array multiplier is comparatively less because the partial products are calculated independently in parallel. The delay associated with the array multiplier is the time taken by the signals to propagate through the gates that form the multiplication array. Booth multiplication is another important multiplication algorithm. Large booth arrays are required for high speed multiplication and exponential operations which in turn require large partial sum and partial carry registers. Multiplication of two *n*-bit operands using a radix-4 booth recording multiplier requires approximately *n* / (2*m*) clock cycles to generate the least significant half of the final product, where *m* is the number of Booth recorder adder stages. Thus, a large propagation delay is associated with this case. Due to the importance of digital multipliers in DSP, it has always been an active area of research and a number of interesting multiplication algorithms have been reported in the literature.

II.Algorithm for 4 x 4 bit Vedic multiplier Using UT(Vertically and crosswise) for two Binary numbers :

CP = Cross Product (Vertically and Crosswise)

```
X3 X2 X1 X0 Multiplicand
Y3 Y2 Y1 Y0 Multiplier
--------------------------------------------------------
H G F E D C B A
P7 P6 P5 P4 P3 P2 P1 P0 Product
--------------------------------------------------------
```

2.7.3 Algorithm for 8 X 8 Bit Multiplication Using UT(Vertically and crosswise) for two Binary numbers :

```
A = A7A6A5A4      A3A2A1A0
        X1            X0
B = B7B6B5B4      B3B2B1B0
        Y1            Y0
        X1 X0
            * Y1 Y0
--------------------------------------------------------
            F E D C
```

CP = X0 * Y0 = C

CP = X1 * Y0 + X0 * Y1 = D

CP = X1 * Y1 = E

Where CP = Cross Product.

To illustrate the multiplication algorithm, let us consider the multiplication of two binary numbers a3a2a1a0 and b3b2b1b0. As the result of this multiplication would be more than 4 bits, we express it as... r3r2r1r0. Line diagram for multiplication of two 4-bit numbers is shown in Fig. 2.2 which is nothing but the mapping of the Fig.2.1 in binary system. For the simplicity, each bit is represented by a circle. Least significant bit r0 is obtained by multiplying the least significant bits of the multiplicand and the multiplier. The process is followed according to the steps shown in Fig.

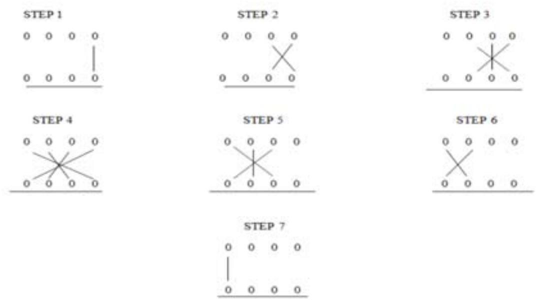


Figure 2.10: Line diagram for multiplication of two 4 - bit numbers

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 bit (referred as rn) and 11 as the carry (referred as cn). It should be clearly noted that cn may be a multi-bit number.
Thus we get the following expressions:
r0=a0b0*;* (1)
c1r1=a1b0+a0b1*;* (2)
c2r2=c1+a2b0+a1b1 + a0b2*;* (3)
c3r3=c2+a3b0+a2b1 + a1b2 + a0b3*;* (4)
c4r4=c3+a3b1+a2b2 + a1b3*;* (5)
c5r5=c4+a3b2+a2b3*;* (6)
c6r6=c5+a3b3 (7)
With c6r6r5r4r3r2r1r0 being the final product.
Hence this is the general mathematical formula applicable to all cases of multiplication.

2.8 MATHEMATICAL FORMULATION OF VEDIC SUTRA

The gifts of the ancient Indian mathematics in the world history of mathematical science are not well recognized. The contributions of saint and mathematician in the field of number theory, 'Sri Bharati Krsna Thirthaji Maharaja', in the fond of Vedic Sutras (formulas) [11] are significant for calculations. He had explored the mathematical potentials from Vedic primers and showed that the mathematical operations can be carried out mentally to produce fast answers using the Sutras. In this paper we are concentrating on "Urdhva-tiryakbyham" sutra.

2.8.1 "URDHVA-TIRYAKBYHAM" SUTRA

The meaning of this sutra is "Vertically and crosswise" and it is applicable to all the multiplication operations. Fig. 2.11 represents the general multiplication procedure of the 4x4 multiplication. This procedure is simply known as array multiplication technique.
Mathematical Background of "UT" Sutra :
Assume that X and Y are two numbers, to be multiplied. Mathematically X and Y can be represented as:

A = $\sum_{i=0}^{n-1} A_i 10^i$

B = $\sum_{j=0}^{n-1} B_j 10^j$

Assume that, their product is equal to . Then P can be represented as:
P=AB
P= $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} A_i B_j 10^{i+j}$
…….. eq (c)
Where ($A_i, B_j \in \{0, 1, \ldots\ldots 9\}$) and 'n' may be any number .
From the above expression, equation no (c), it can observed that each digit is multiplied consecutively and shifted towards the proper positions for partial product generation. Finally the partial products are added with the previous carry to produce the final results.

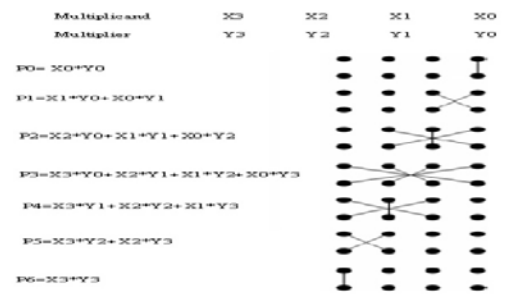


Fig2.11: Multiplication procedure using "UT" sutra

HARDWARE ARCHITECTURE

The design starts first with Multiplier design that is 4x4 bit multiplier. Here, "Urdhva Tiryakbhyam Sutra" or "Vertically and Crosswise Algorithm" for multiplication has been effectively used to develop digital multiplier architecture. This algorithm is quite different from the traditional method of multiplication that is to add and shift the partial products. This Sutra shows how to handle multiplication of a larger number (N x N, of N bits each) by breaking it into smaller numbers of size (N/2 = n, say) and these smaller numbers can again be broken into smaller numbers (n/2 each) till we reach multiplicand size of (4 x 4).Thus simplifying the whole multiplication process. For Multiplier, first the basic blocks that are the 4x4 bit multipliers have been made and then, using these blocks, 8x8 block has been made. The main advantage of the Vedic multiplication algorithm (UT Sutra) stems from the fact that it can be easily realized on hardware.

In this project we are going to implement 8x8 bit VM using reversible logic gates.

Steps to design and implement 8x8 bit Vedic multiplier are:

1. We have to implement 2x2 Vedic multiplier by using Vedic mathematics algorithm
2. By using four 2x2 multipliers we implemented 4x4 Vedic multiplier.
3. Now implement 8x8 multiplier using four 4x4multipliers.

Implementation of 8x8 bit Reversible Vedic multiplier:

The block diagram of the 8x8 Vedic Multiplier is presented in the figure3.1.It consists of four 2X2 multipliers each of which procedures 16 bits as inputs; 8 bits from the multiplicand and 8 bits from the multiplier. The output of the first 4x4 multiplier multiplier are entrapped as the lowest four bits of the final result of multiplication .the second 4x4 multiplier output and third 4x4 multiplier are given as input to the ripple carry bit 8 adder. The output from the 8-bit carry adder along with the four zeros are concatenated upper four bits of the output of the first 4X4 multiplier are given as input to the nine bit ripple carry adder.the upper four bits are entrapped as the middle bits in the final result. The upper bit output of the nine bit ripple carry adder concatenated with two zeroes along with the fouth output of 4x4 multiplier gives the final higher output bits.These bits form the upper bits of the final result.

The ripple carry adder is consummated (realized) using the HNG Gate. The number of bits that need to be ripple carried verdicts the number of HNG gates to be used. Thus a 8 bit ripple carry adder needs 8 HNG gates and the 9 bit adder requires 9 HNG gates. This design also does not take into consideration the fan out gates.

Here "a" and "b" are two numbers to be multiplied and "y" is the product. With this design we are now ready to code this in verilog easily using reversible gates. To make the design more modular we write code for gates first and then instantiate it to have the final product.

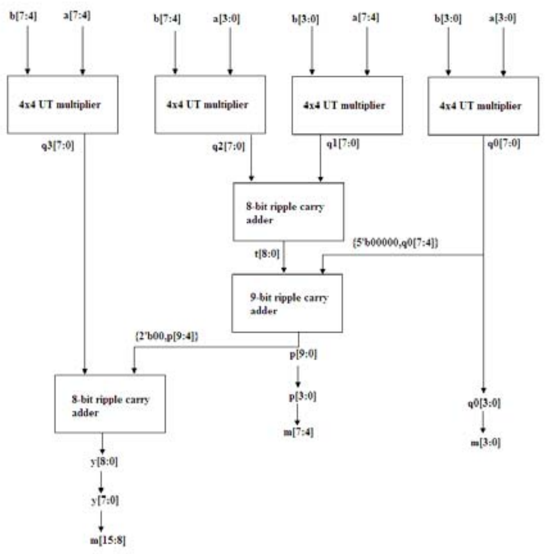


Figure 3.1: Reversible Logic Implementation of 8X8 UT Multiplier

Ripple Carry Adders

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage. In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry in of that stage occurs. A propagation delay inside the logic circuitry is the reason behind this. Propagation delay is time elapsed between the application of an input and occurrence of the corresponding output. Consider a NOT gate, When the input is “0″ the output will be “1″ and vice versa. The time taken for the NOT gate’s output to become “0″ after the application of logic “1″ to the NOT gate’s input is the propagation delay here. Similarly the carry propagation delay is the time elapsed between the application of the carry in signal and the occurrence of the carry out (Cout) signal.

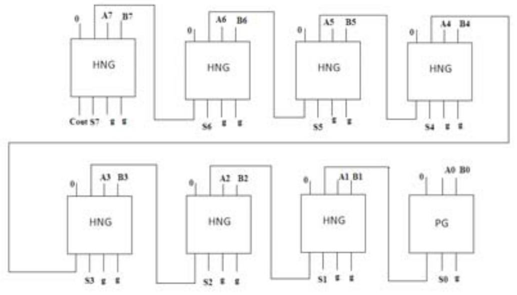


Figure 3.2: 8-Bit Ripple Carry Adder

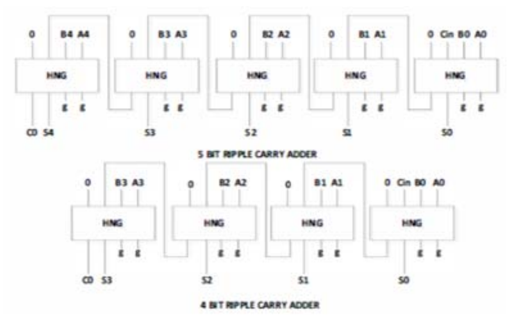


Figure 3.3: 9-Bit Ripple Carry Adder

4x4 UT Vedic multiplier

The block diagram of the 4X4 Vedic Multiplier is presented in the figure3.3.It consists of four 2X2 multipliers each of which procedures four bits as inputs; two bits from the multiplicand and two bits from the multiplier. The lower two bits of the output of the first 2X2 multiplier are entrapped as the lowest two bits of the final result of multiplication. Two zeros are concatenated with the upper two bits and given as input to the four bit ripple carry adder. The other four input bits for the ripple carry adder are obtained from the second 2X2 multiplier. Likewise the outputs of the third 2X2 multipliers and output of four bit ripple carry adder are given as inputs to the five bit ripple carry adder. The upper bit output of the five bit ripple carry adder and output from the fourth 2x2 multiplier are final output bits .These bits form the upper bits of the final result.

The ripple carry adder is consummated (realized) using the HNG Gate. The number of bits that need to be ripple carried verdicts the number of HNG gates to be used. Thus a 4 bit ripple carry adder needs 4 HNG gates and the 5 bit adder requires 5 HNG gates. This design also does not take into consideration the fan out gates.

Here "a" and "b" are two numbers to be multiplied and "y" is the product. With this design we are now ready to code this in verilog easily using reversible gates. To make the design more modular we write code for gates first and then instantiate it to have the final product.

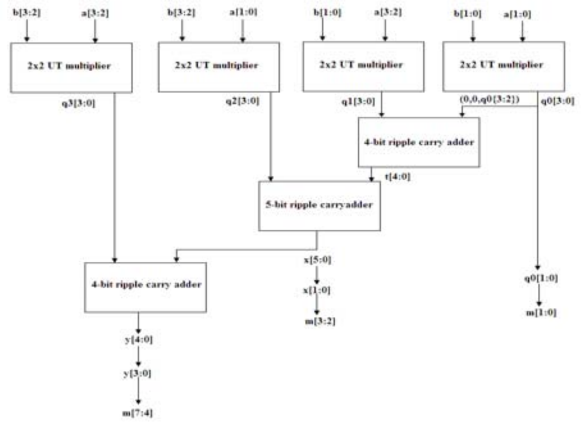


Figure 3.4: 4X4 UT Multiplier

2x 2UT Vedic multiplier

Figure illustrates the steps to multiply two 2 bit numbers. Converting the above figure to a hardware equivalent we have 5 Peres gate and 1 CNOT gate which will act as 2 bit multipliers and half adders to add the products to get the final product. Here is the hardware detail of the multiplier

Here "a" and "b" are two numbers to be multiplied and "y" is the product. With this design we are now ready to code this in verilog easily using reversible gates. To make the design more modular we write code for gates first and then instantiate it to have the final product.

The circuit requires a total of six reversible logic gates out of which five are Peres gates and remaining one is the Feynman Gate

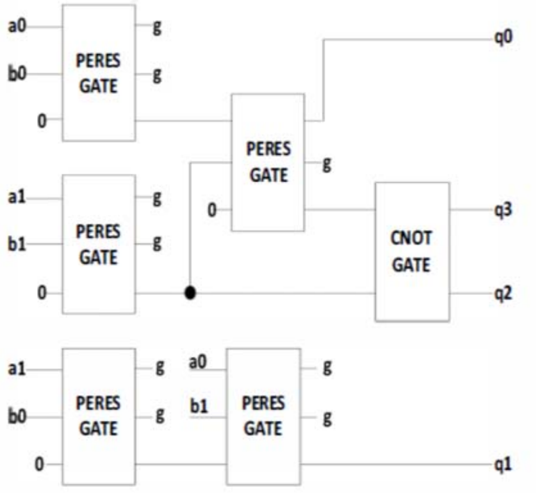


Figure 3.5: 2X2 UT Multiplier

SIMULATION RESULTS

8X8 UT Vedic multiplier design functionality is verified using mentor graphic's "modelsim" software tool.

Simulation Results Of 2x2 Bit Multiplier

The below figures illustrates the output wave form for 2x2 Vedic multiplier for different input values for multiplier and multiplicand is shown in figure. The first 2 bits indicates the input and i indicates the output of the multiplier. The shown fig gives the description as follows input bit a='11' and b='11' and the result i gives the product of a & b.

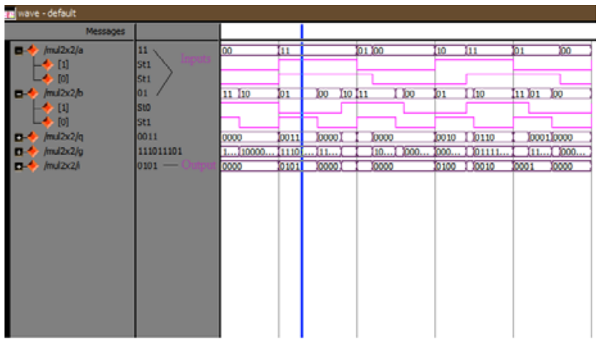


Figure 4.1: Simulation Result for 2x2UT Multiplier

4.1.2 Simulation Results Of 4x4 Bit Multiplier

The below figures illustrates the output wave form for 8x8 Vedic multiplier for different input values for multiplier and multiplicand is shown in figure. The first 2 bits indicates the input and m indicates the output of the multiplier. The shown fig gives the description as follows input bit a='1111' and b='1111' and the result m gives the product of a & b.

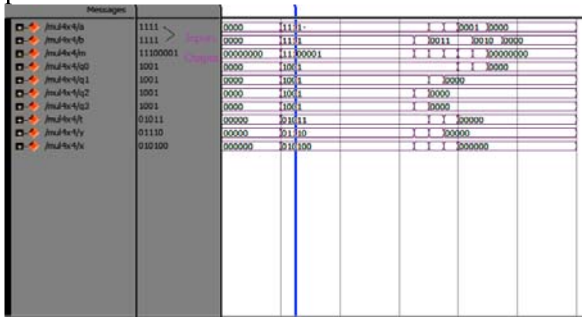


Figure 4.2: Simulation Result for 4x4 UT Multiplier

Simulation Results Of 8x8 Bit Multiplier

The below figures illustrates the output wave form for 8x8 Vedic multiplier for different input values for multiplier and multiplicand is shown in figure. The first 2 bits indicates the input and m indicates the output of the multiplier. The shown fig gives the description as follows input bit a='11111111' and b='11111111' and the result gives the product of a & b.

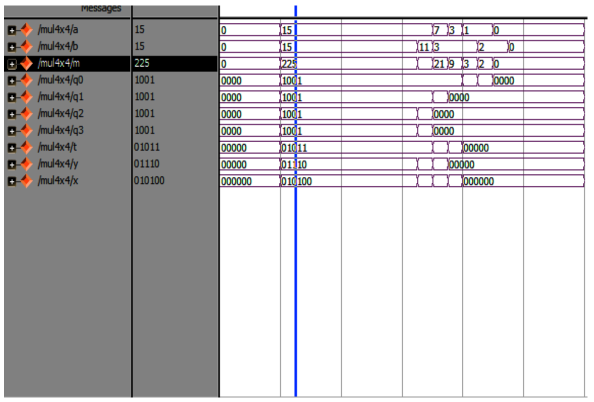
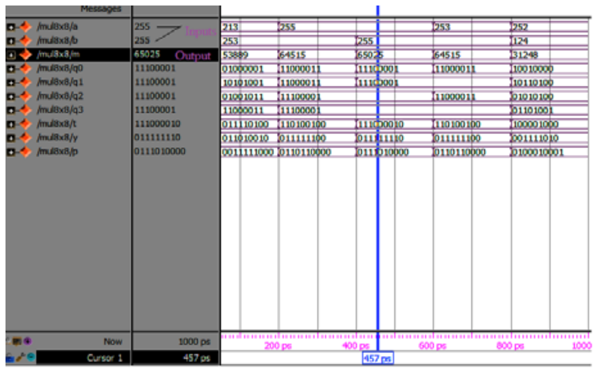


Figure 4.3: Simulation Result for 8x8 UT Multiplier

SYNTHESIS RESULTS

Device utilization summary:
Selected Device: 3s500efg320-4
Number of Slices: 92 out of 4656 1%
Number of Slice Flip Flops:3 out of 9312 0%
Number of 4 input LUTs: 160 out of 9312 1%
Number of IOs: 37
Number of bonded IOBs:37 out of 232 15%
Number of GCLKs:1 out of 24 4%
Timing Summary:
Minimum period: 2.610ns (Maximum Frequency: 383.142MHz)
Minimum input arrival time before clock: No path found.
Maximum output required time after clock: 4.450ns
Maximum combinational path delay: 27.427ns



RTL SCHEMATIC

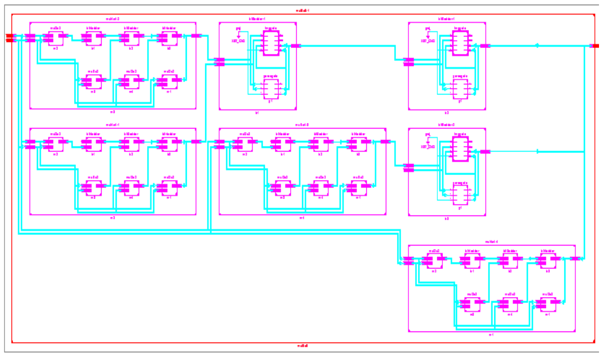


Figure 4.4: RTL Schematic 8x8 UT Multiplier

FPGA IMPLEMENTATION

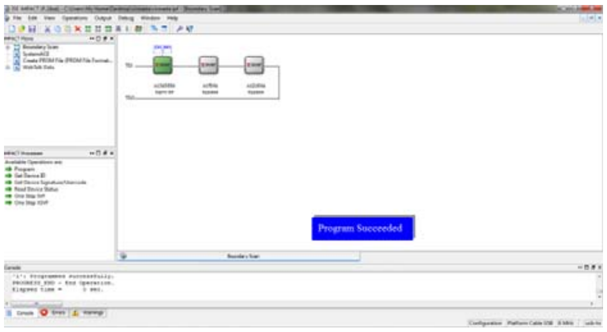


Figure 4.5: FPGA Implementation Output

Figure 4.6: Design Implementation on XCS500E SPARTAN 3E

REFERENCES

[1] H. Thapliyal , M. B. Shrinivas and H. Arbania, ―Design and Analysis of a VLSI Based High Performance Low Power Parallel Square Architecture‖, Int. Conf. Algo. Math. Comp. Sc., LasVegas, June 2005, pp. 72-76

[2] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda, Delhi (1965)..

[3] P. D. Chidgup kar and M. T. Karad, ―The Imp lamentation of Vedic Algorithms in Digital Signal Processing‖, Global J. of Engg. Edu, vol.8, no.2, 2004.

[4] Shamim Akhter,‖VHDL Implementation Of Fast NXN Multiplier Based On Vedic Mathematics‖, Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE.

[6] Rakshith Saligram and Rakshith T.R. "Design of Reversible Multipliers for linear filtering Applications in DSP" International Journal of VLSI Design and Communication systems, Dec-12

[7] R. Landauer,"Irreversibility and Heat Generation in the Computational Process", IBM Journal of Research and Development, 5, pp.183-191, 1961.

[8] H. Thapliyal and M.B. Srinivas, "Novel Reversible Multiplier Architecture Using Reversible TSG Gate", Proc.IEEE International Conference on Computer Systems and Applications, pp. 100-103, March 2006.

[9] Shams, M., M. Haghparast and K. Navi, Novel reversible multiplier circuit in nanotechnology. World Appl. Sci. J., 3(5): 806-810.

[10] Somayeh Babazadeh and Majid Haghparast, "Design of a Nanometric Fault Tolerant Reversible Multiplier Circuit" Journal of Basic and Applied Scientific Research, 2012.

[11] Thapliyal, H., M.B. Srinivas and H.R. Arabnia, 2005, A Reversible Version of 4x4 Bit Array Multiplier with Minimum Gates and Garbage Outputs, Int. Conf. Embedded System, Applications (ESA'05), Las Vegas, USA, pp: 106 114.

[12] M. Haghparast et al. , "Design of a Novel Reversible Multiplier Circuit using HNG Gate in Nanotechnology," in World Applied Science Journal, Vol. 3, No. 6, pp. 974-978,2008.

[13] M. S. Islam et al. , "Realization of Reversible Multiplier Circuit," in Information Tech. 1, Vol. 8, No. 2, pp. 117-121,2005.