High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics

M.Nagaraju[1], R.Surya Prakash[2], B.Vijay Bhaskar3
[1]M.Tech Student, Department of ECE, Avanthi's St.Theressa Institute of Engg & Tech College, Garividi.
[2]Assistant professor, Department of ECE, Avanthi's St.Theressa Institute of Engg & Tech College, Garividi.
[3] HOD,Associate professor.Department of ECE, Avanthi's St.Theressa Institute of Engg &Tech College, Garividi

.

ABSTRACT:

The main aim of the project is to improve the speed of the complex multiplier by using vedic mathematics. This 'Vedic Mathematics' is the name given to the ancient system of mathematics, or, to be precise, a unique technique of calculations based on simple rules and principles, with which any mathematical problem can done with the help of arithmetic, algebra, geometry or trigonometry can be solved. Traditionally complex multiplier provides less speed only, because it does not use Vedic Mathematics concept. By using 'Vedic Mathematics' concept we can skip carry propagation delay. The system is based on 16 Vedic *sutras*, in which we are using one kind of vedic sutra actually word-formulae describing natural ways of solving a whole range of mathematical problems.The main design features of the proposed system are the reconfigurability and flexibility. The proposed system is design using VHDL or Verilog HDL and is implemented through Xilinx ISE 9.1i navigator or modelsim6.0 softwares.

Kewords: Complex Multiplier, Exponent Determinant, High Speed Radix Selection Unit, Vedic Formulas.

1. INTRODUCTION

Complex multiplication is of immense importance in Digital Signal Processing (DSP) and Image Processing (IP). To implement the hardware module of Discrete Fourier Transformation (DFT), Discrete Cosine Transforation (DCT), Discrete Sine Transformation (DST) and modem broadband communications; large numbers of complex multipliers are required. Complex number multiplication is performed using four real number multiplications and two additions/ subtractions. In real number processing, carry needs to be propagated fom the least signifcant bit (LSB) to the most signifcant bit (MSB) when binary partial products are added [1]. Therefore, the addition and subtraction afer binary multiplications limit the overall speed. Many alterative method had so far been proposed for complex number multiplication [2-7] like algebraic transformation based implementation[2], bit-serial multiplication using offset binary and distributed arithmetic [3], the CORDIC (coordinate rotation digital computer) algorithm [4], the quadratic residue number system(QRNS)[5],and recently, the redundant complex numbersystem (RCNS) [6]. at the expense of three additions as compared to the direct method implementation. A left to right array [7] for the fast multiplication has been reported in 2005, and the method is not further extended for complex multiplication. But, all the above techniques require either large overhead for pre/postprocessing or long latency.

In algorithmic and structural levels, a lot of multiplication techniques had been developed to enhance the effciency of the multiplier; which encounters the reduction of the partial products and/or the methods for their partial products addition, but the principle behind multiplication was same in all cases. Vedic Mathematics is the ancient system of Indian mathematics which has a unique technique of calculations based on 16 Sutras (Formulae). "Urdhva-tiryakbyham" is a Sanskrit word means vertically and crosswise formula is used for smaller number multiplication. "Nihilam Navatascaramam Dasatah" also a Sanskrit term indicating "all fom 9 and last fom 10", formula is used for large number multiplication and subtraction. All these forulas are adopted fom ancient Indian Vedic Mathematics. In this work we formulate this mathematics for designing the complex multiplier architecture in transistor level wit two clear goals in mind such as: i) Simplicity and modularity multiplications for VLSI implementations and ii) The elimination of carry propagation for rapid additions and subtractions.

Mehta et al. [9] have been proposed a multiplier design using "Urdhva-tiryakbyham" sutras, which was adopted fom the Vedas. The formulation using this sutra is similar to the modem array multiplication, which also indicating the carry propagation issues. A multiplier design using "Nikhilam Navatascaramam Dasatah" sutras has been reported by Tiwari et. al [10] in 2009, but he has not implemented the hadware module for multiplication. That

Multiplier implementation in the gate level (FPGA) using Vedic Mathematics has already been reported but to the best of our knowledge till date there is no report on transistor level (ASIC) implementation of such complex multiplier. By employing the Vedic mathematics, an N bit complex number multiplication was transformed into four multiplications for real and imaginary terms of the final product. this sutra is used for the multiplication purpose, with less number of partial products generation, in comparison with array based multiplication. When compared with existing methods such as the direct method or the strength reduction technique, our approach resulted not only in simplifed arithmetic operations, but also in a regular arraylike structure. The multiplier is fully parameterized, so any confguration of input and output word-lengths could be elaborated. Transistor level implementation for perfonance parameters such as propagation delay, dynamic leakage power and dynamic switching power consumption calculation of the proposed method was calculated by Model sim using 90 nm standard CMOS technology and compared with the other design like distributed.

2. ASIC

The term **'ASIC'** stands for '**application-specific integrated circuit**'. An ASIC is basically an integrated circuit designed specifically for a special purpose or application. Strictly speaking, this also implies that an ASIC is built only for one and only one customer. An example of an ASIC is an IC designed for a specific line of cellular phones of a company, whereby no other products can use it except the cell phones belonging to that product line. The opposite of an ASIC is a standard product or general purpose IC, such as a logic gate or a general purpose microcontroller, both of which can be used in any electronic application by anybody of the complex values

side from the nature of its application, an ASIC differs from a standard product in the nature of its availability.

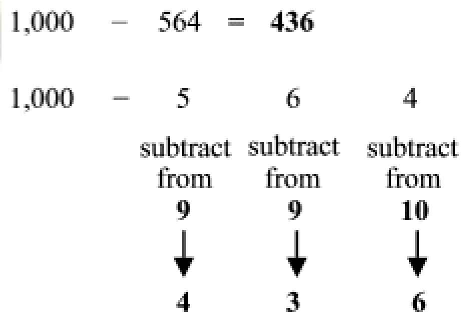
The intellectual property, design database, and deployment of an ASIC is usually controlled by just a single entity or company, which is generally the end-user of the ASIC too. Thus, an ASIC is proprietary by nature and not available to the general public. A standard product, on the other hand, is produced by the manufacturer for sale to the general public. Standard products are therefore readily available for use by anybody for a wider range of applications. The first ASIC's, known as uncommitted logic array or ULA's, utilized gate array technology. Having up to a few thousand gates, they were customized by varying the mask for metal interconnections. Thus, the functionality of such a device can be varied by modifying which nodes in the circuit are connected and which are not. Later versions became more generalized, customization of which involve variations in both the metal and polysilicon layers.

ASIC's are usually classified into one of three categories: full-custom, semi-custom, and structured.

3.VEDIC SUTRAS

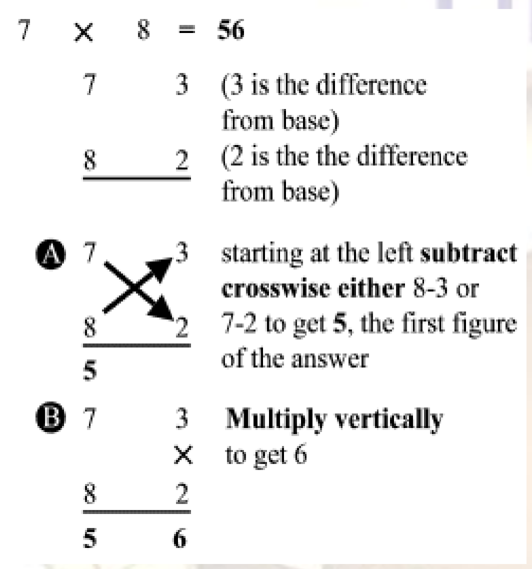
The gifs of the ancient Indian mathematics in the world history of mathematical science are not well recognized. The contributions of saint and mathematician in the feld of number theory, 'Sri Bharati Krsna Thirthaji Maharaja', in the fon of Vedic Sutras (formulas) [11] are signifcant for calculations. 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", and "Nikhilam Navatascaramam Dasatah" formulas and other formulas are beyond the scope of this paper.

"Urdhva-tiryakbyham " Sutra: The meaning of this sutra is "Vertically and crosswise" and it is applicable to all the multiplication operations. Fig. 1 represents the general multiplication procedure of the 4x4 multiplication. This procedure is simply known as array multiplication technique [12]. It is an effcient multiplication technique when the multiplier and multiplicand lengths are small, but for the larger length multiplication this technique is not suitable because a large amount of carr propagation delays are involved in these cases. To overcome this problem we are describing Nikhilam sutra for calculating the multiplication of two larger numbers.The remarkable system of Vedic maths was rediscovered from ancient Sanskrit texts early last century. The system is based on 16 sutras or aphorisms, such as: "by one more than the one before" and "all from nine and the last from 10". These describe natural processes in the mind and ways of solving a whole range of mathematical problems. For example, if we wished to subtract 564 from 1,000 we simply apply the sutra "all from nine and the last from 10". Each figure in 564 is subtracted from nine and the last figure is subtracted from 10, yielding 436.



This can easily be extended to solve problems such as 3,000 minus 467. We simply reduce the first figure in

3,000 by one and then apply the sutra, to get the answer 2,533.We have had a lot of fun with this type of sum, particularly when dealing with money examples, such as £10 take away £2. 36. Many of the children have described how they have challenged their parents to races at home using many of the Vedic techniques - and won. This particular method can also be expanded into a general method, dealing with any subtraction sum.The sutra "vertically and crosswise" has many uses. One very useful application is helping children who are having trouble with their tables above 5x5. For example 7x8. 7 is 3 below the base of 10, and 8 is 2 below the base of 10.


7 × 8 = **56**

7 3 (3 is the difference from base)

8 2 (2 is the the difference from base)

A 7 3 starting at the left **subtract crosswise either** 8-3 or 7-2 to get **5**, the first figure of the answer

8 2

5

B 7 3 **Multiply vertically** × to get 6

8 2

5 6

The whole approach of Vedic maths is suitable for slow learners, as it is so simple and easy to use.The sutra "vertically and crosswise" is often used in long multiplication. Suppose we wish to multiply 32 by 44. We multiply vertically 2x4=8. Then we multiply crosswise and add the two results: 3x4+4x2=20, so put down 0 and carry 2. Finally we multiply vertically 3x4=12 and add the carried 2 =14. Result: 1,408. We can extend this method to deal with long multiplication of numbers of any size. The great advantage of this system is that the answer can be obtained in one line and mentally. By the end of Year 8, I would expect all students to be able to do a "3 by 2" long multiplication in their heads. This gives enormous confidence to the pupils who lose their fear of numbers.The mathematical expression for the proposed algorithm is shown below. Broadly this algorithm is divided into three parts. (i) Radix Selection Unit (ii) Exponent Determinant (iii) Multiplier.Consider two n bit numbers X and Y. kl and k2 are the exponent of X and Y respectively. X and Y can be represented as:

$$X = 2^{k_1} \pm z_1 \qquad (12)$$

$$Y = 2^{k_2} \pm z_2 \qquad (13)$$

For the fast multiplication using Nikhilam sutra the bases of the multiplicand and the multiplier would be same, (here we have considered different base) thus the equation can be rewritten as hardware implementation of this mathematics is shown in Fig. 2. The architecture can be decomposed into three main subsections: (i) Radix Selection Unit (RSU) (ii) Exponent

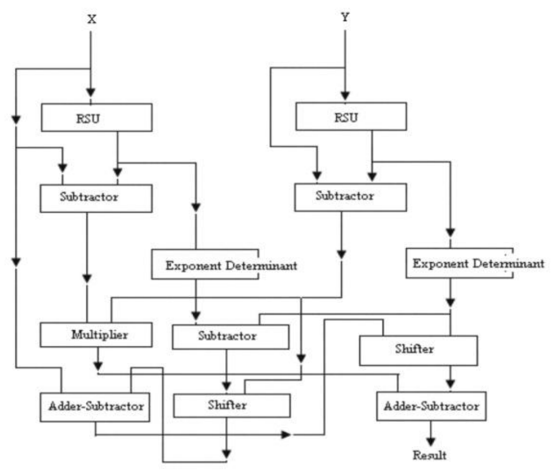


Fig. 2: Hardware implementation of "Nikhilam Sutra"

The Block level architecture of RSU is shown in Fig. 3.RSU consists of three main subsections: (i) Exponent Determinant (ED), (ii) Mean Determinant (MD) and (iii) Comparator. 'n' number bit from input X is fed to the ED block. The maximum power of X is extracted at the output which is again fed to shifter and the adder block.

The second input to the shifter is the (n+1) bit representation of decimal '1'.If the maximum power of X from the ED unit is (n-1) then the output of the shifter is 2^{n-1}. The adder unit is needed to increment the value of the maximum power of X by '1'. The second shifter is needed to generate the value of 2".Here n is the incremented value taken from the adder block. The Mean Determinant unit is required to compute the mean of $(2^{n-1}+2^n)$. The Comparator compares the actual input with the mean value of $(2^{n-1}+2^n)$. If the input is greater than the mean then 2^n is selected as the required radix. If the input is less than the mean then 2^{n-1} is selected as the radix. The select input to the multiplexer block is taken from the output of the comparator.

A. Mathematical expression for RSU

Consider an 'n' bit binary number X, and it can be represented as

$X = \sum_{i=0}^{n-1} x_i 2^i$ Where $x_i \epsilon \{0,1\}$. Then the values of X must lie in the range $2^{n-1} \leq X < 2^n$. Consider the mean of the range is equals to A.

$$A = \frac{2^{n-1}+2^n}{2} \quad (20)$$

$= 2^{n-2} \times 3$

If X >A Then radix = 2^n

If $X \leq A$ Then radix = 2^{n-1}

***B.Exponent Determinant*:** The hardware implementation of the exponent determinant is shown in Fig. 4.The integer part or exponent of the number from the binary fixed point number can be obtained by the maximum power of the radix. For the nonzero input, shifting operation is executed using parallel in parallel out (PIPO) shift registers. The number of select lines (in FigA it is denoted as S1, So) of the PIPO shifter is chosen as per the binary representation of the number (N-1)IO. 'Shift' pin is assigned in PIPO shifter to check whether the number is to be shifted or not (to initialize the operation 'Shift' pin is initialized to low). A decrementer [13] has been integrated in this architecture to follow the maximum power of the radix. A sequential searching procedure has been implemented here to search the first 'I' starting from the MSB side by using shifting technique. For an N bit number, the value (N-I)1O is

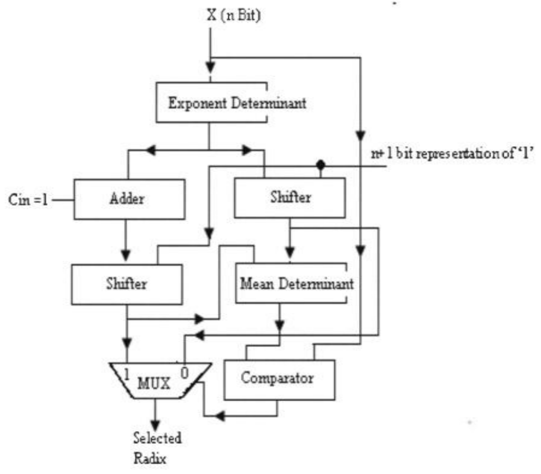


Fig. 3: Hardware implementation of RSU

fed to the input of decrementer. The decrementer is decremented based on a control signal which is generated by the searched result. If the searched bit is '0' then the control signal becomes low then decrementer start decrementing the input values. The searched bit is used as a controller of the decrementer. When the searched bit is 'I' then the control signal becomes high and the decrementer stops further decrementing and shifter also stops shifting operation. The output of the decrementer shows the integer part (exponent) of the number.

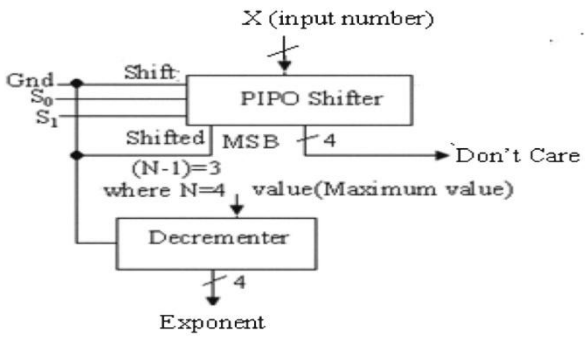


Fig. 4: Hardware implementation of exponent determinant

4.COMPLUX MULTIPLIER

Complex multiplier design by using parallel adders and subtractors has been designed by Saha[I]. Fig. 5 shows the direct method for implementation of the complex multiplier design. In this paper complex multiplier design is done using Vedic Multipliers and Vedic subtractors. *Multiplication Algorithm* The Complex multiplier design and its functionality were discussed in the previous chapters. Now this chapter deals with the simulation and synthesis results of the Complex multiplier. Here Modelsim tool is used in order to simulate the design and checks the functionality of the design.

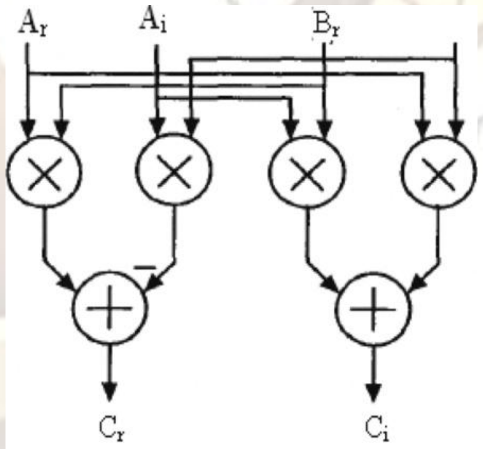
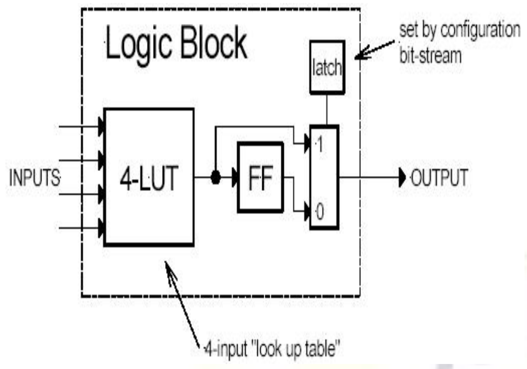
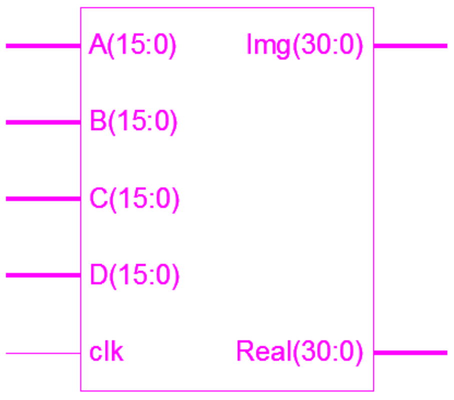


Fig. 5: Implementation method of complex multiplier

Above shows the basic steps involved in implementation. The initial design entry of may be VHDL, schematic or Boolean expression. The optimization of the Boolean expression will be carried out by considering area or speed. In technology mapping, the transformation of optimized Boolean expression to FPGA logic blocks, that is said to be as Slices. Here area and delay optimization will be taken place. During placement the algorithms are used to place each block in FPGA array. Assigning the FPGA wire segments, which are programmable, to establish connections among FPGA blocks through routing. The configuration of final chip is made in programming unit.



RTL Schematic: The RTL (Register Transfer Logic) can be viewed as black box after synthesize of design is made. It shows the inputs and outputs of the system. By double-clicking on the diagram we can see gates, flip-flops and MUX.



5. Simulation Result Analysis

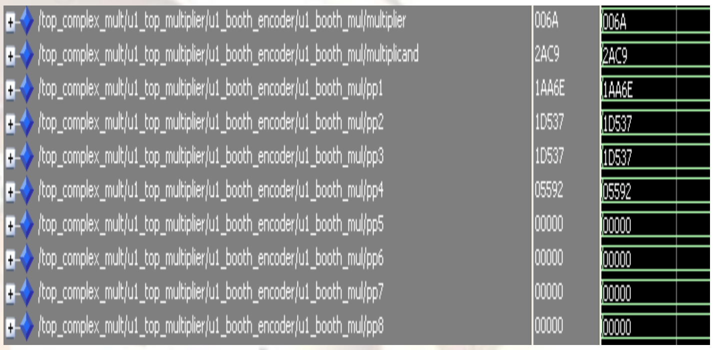
All the algorithm of this paper was simulated and their functionality was examined by using Spice Spectre. Performance parameters such as propagation delay and power consumptions analysis of this paper using standard 90nm CMOS technology. To evaluate the performance parameters, we give the values of the computational effort using array multiplier and Vedic multiplier. As shown, the application of the Vedic method for multiplication cuts the amount of the hardware as well as increases the performance parameters such as propagation delay, dynamic switching power consumptions, and dynamic leakage power consumptions. The performance parameters analysis using array multiplication and Vedic multiplication is shown in Table I. Input data is taken as a regular fashion for experimental purpose. We have kept our main concentration for reducing the propagation delay, dynamic switching power and dynamic leakage power consumption and energy delay product.

A comparison between different architecture in terms of propagation delay and dynamic switching power consumption arithmetic, parallel adder based implementation, and algebraic transformation based implementation. Once the functional verification is done, the design will be taken to the Xilinx tool for Synthesis process and the netlist file generation.

The Appropriate test cases have been identified in order to test this modelled Complex multiplier design. Based on the identified values, the simulation results which describes the operation of the Complex multiplier design has been achieved. This proves that the modelled design works properly as per the process.

Simulations Results:

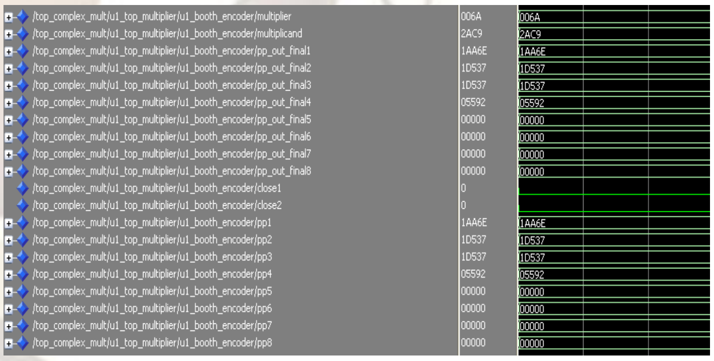
Partial products Generators:



```
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/multiplier     006A   006A
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/multiplicand   2AC9   2AC9
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp1            1AA6E  1AA6E
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp2            1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp3            1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp4            05592  05592
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp5            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp6            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp7            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/u1_booth_mul/pp8            00000  00000
```

Figure :Simulation results for Partial product generator

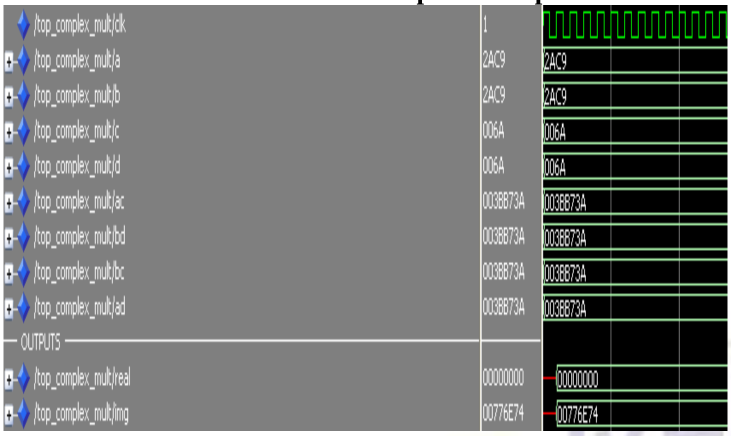
Booth Encoder:



```
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/multiplier     006A   006A
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/multiplicand   2AC9   2AC9
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final1  1AA6E  1AA6E
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final2  1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final3  1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final4  05592  05592
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final5  00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final6  00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final7  00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp_out_final8  00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/close1         0
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/close2         0
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp1            1AA6E  1AA6E
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp2            1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp3            1D537  1D537
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp4            05592  05592
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp5            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp6            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp7            00000  00000
/top_complex_mult/u1_top_multiplier/u1_booth_encoder/pp8            00000  00000
```

Figure :Simulation results for Booth Encoder

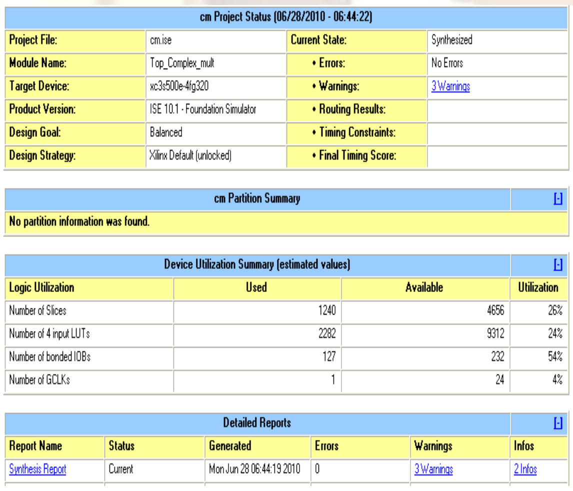
Simulation results for Complex Multiplier :



6.SYNTHESIS RESULT

The developed Complex multiplier design 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. This Complex multiplier design can be synthesized on the family of Spartan 3E.Here in this Spartan 3E 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”. The design of Reversible Watermarking Algorithm is synthesized and its results were analyzed as follows.

Device utilization summary:



cm Project Status (06/28/2010 - 06:44:22)			
Project File:	cm.ise	Current State:	Synthesized
Module Name:	Top_Complex_mult	• Errors:	No Errors
Target Device:	xc3s500e-4fg320	• Warnings:	3 Warnings
Product Version:	ISE 10.1 - Foundation Simulator	• Routing Results:	
Design Goal:	Balanced	• Timing Constraints:	
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	

cm Partition Summary	[-]
No partition information was found.	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	1240	4656	26%
Number of 4 input LUTs	2282	9312	24%
Number of bonded IOBs	127	232	54%
Number of GCLKs	1	24	4%

Detailed Reports					[-]
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon Jun 28 06:44:19 2010	0	3 Warnings	2 Infos

This device utilization includes the following.

- Logic Utilization
- Logic Distribution
- Total Gate count for the Design

The device utilization summery is shown above in which its gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown above.

Timing Summary: Speed Grade: -4

Minimum period: No path found

Minimum input arrival time before clock: 28.239ns

Maximum output required time after clock: 4.283ns

Maximum combinational path delay: No path found

7.CONCLUSION

In this paper we report on a novel complex number multiplier design based on the formulas of the ancient Indian Vedic Mathematics, highly suitable for high speed complex arithmetic circuits which are having wide application in VLSI signal processing. The implementation was done in Spice spectre and compared with the mostly used architecture like distributed.The developed Complex Multiplier design is modelled and is simulated using the Modelsim tool.The simulation results are discussed by considering different cases.The RTL model is synthesized using the Xilinx tool in Spartan 3E and their synthesis results were discussed with the help of generated reports.

8.REFERENCES

1. P. K. Saha, A. Banerjee, and A. Dandapat, "High Speed Low Power Complex Multiplier Design Using Parallel Adders and Subtractors," *International Journal on Electronic and Electrical Engineering, (IEEE)*, vol 07, no. II, pp 38-46, Dec. 2009.
2. R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, Reading, MA: Addison-Wesley, 1987.
3. S. He, and M. Torkelson, "A pipelined bit-serial complex multiplier using distributed arithmetic," in *proceedings IEEE International Symposium on Circuits and Systems*, Seattle, WA, April 30-May-03, 1995,pp. 2313-2316.
4. J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput*, vol. EC-8, pp. 330-334, Sept. 1959.
5. R. Krishnan, G. A. Jullien, and W. C. Miller, "Complex digital