DISSERTATION II REPORT
ON

COMPARATIVE ANALYSIS OF FAST MULTIPLIERS FOR DSP APPLICATIONS

Submitted in partial fulfilment of the requirement for the award of the degree of

MASTER OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING

Submitted by

Srinivas Pappala
Reg.no:11502777

Under the Guidance of

Dr. Ravi Shankar Mishra
Associate professor of ECE

Department of Electronics & communication Engineering

Lovely Professional University

Phagwara-144411, Punjab (India)

April 2017

TOPIC APPROVAL PERFORMA

School of Electronics and Electrical Engineering

Program : P175::M.Tech. (Electronics and Communication Engineering) [Full Time]

COURSE CODE :	ECE521	**REGULAR/BACKLOG :**	Regular	**GROUP NUMBER :**	EEERGD0256
Supervisor Name :	Dr. Ravi Shankar Mishra	**UID :**	19053	**Designation :**	Associate Professor

Qualification : ____________________ **Research Experience :** ____________________

SR.NO.	NAME OF STUDENT	REGISTRATION NO	BATCH	SECTION	CONTACT NUMBER
1	Pappala Srinivas	11502777	2015	E1514	9849303284

SPECIALIZATION AREA : VLSI Design **Supervisor Signature:** ________________

PROPOSED TOPIC : Comparative analysis of fast multiplier for DSP Applications

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	**Parameter**	**Rating (out of 10)**
1	Project Novelty: Potential of the project to create new knowledge	8.00
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	7.67
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	8.00
4	Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills.	7.67
5	Social Applicability: Project work intends to solve a practical problem.	7.67
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	7.67

PAC Committee Members		
PAC Member 1 Name: Anshul Mahajan	UID: 11495	Recommended (Y/N): Yes
PAC Member 2 Name: Dushyant Kumar Singh	UID: 13367	Recommended (Y/N): NA
PAC Member 3 Name: Cherry Bhargava	UID: 12047	Recommended (Y/N): Yes
PAC Member 4 Name: Anshul Mahajan	UID: 11495	Recommended (Y/N): Yes
DAA Nominee Name: Manie Kansal	UID: 15692	Recommended (Y/N): NA

Final Topic Approved by PAC: **Comparative analysis of fast multiplier for DSP Applications**

Overall Remarks: Approved

PAC CHAIRPERSON Name: 11211::Prof. Bhupinder Verma **Approval Date:** 08 Oct 2016

ABSTRACT

Multiplier is an essential functional block of a microprocessor because multiplication is needed to be performed repeatedly in almost all scientific calculations. Therefore, design of fast and low power and area of binary multiplier is very important particularly for Digital Signal Processors. A typical processor central processing unit devotes a considerable amount of processing time in performing arithmetic operations, particularly multiplication operations. Multiplication is one of the basic arithmetic operations and it requires substantially more hardware resources and processing time than addition and subtraction. In fact, 8.72% of all the instruction in typical processing units is multiplication. In this paper, comparative study of different multipliers is done for low power requirement and high speed and area. The paper gives information of Vedic multiplier, Wallace tree multiplier and Baugh wooley multiplier. These are utilized for multiplication to improve the speed, area and power parameters of multipliers for Digital Signal Processors.

Depending upon the parametric analysis these fast multipliers have utilized in MAC unit in DSP application.

CERTIFICATE

This is to certify that **Srinivas Pappala** have completed objective formulation of his Dissertation-II titled "**Comparative Analysis of Fast Multipliers for DSP Applications"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his original study and research. No part of the project has ever been submitted for any other degree at any University.

The project is fine for the submission and fulfilment of the condition for the award of degree of Master of Technology in Electronic and Communication Engineering.

Dr. Ravi Shankar Mishra
Associate Professor
Lovely Professional University
Phagwara-144411, Punjab.
Date**:**

ACKNOWLEDGEMENT

I would like to thank **Lovely Professional University** for giving me opportunity to use their resource and work in such a challenging environment. I am grateful to the individuals whom contributed their valuable time towards my dissertation.

I wish to express my sincere and heart full gratitude to my guide "**Dr. Ravi Shankar Mishra**" Associate professor, who guides me to take up this dissertation in sync with global trends in scientific approach.

Finally, I also express our sincere thanks to all our colleagues and friends for their suggestions and encouragement during the session.

Srinivas Pappala
Reg. no:11502777

DECLARATION

I student of M. TECH under ECE Discipline at LOVELY PROFESSIONAL UNIVERSITY , Phagwara; hereby declare that the Dissertation-II report entitled on **“Comparative Analysis of Fast Multipliers for DSP Applications”,** is an authentic record of my own work carried out as the requirements for the award of degree of Master of Technology on “Electronic and Communication” at Lovely Professional University, Phagwara; under the guidance of “**Dr. Ravi Shankar Mishra**”, an Associate professor of Deportment of Electronics and Communication Engineering, during January to May,2017.

Srinivas Pappala
Reg.no:11502777
Date:

It is certified that the above statement is correct to the best of my knowledge and belief.

Dr. Ravi Shankar Mishra

Associate Professor
Lovely Professional University
Phagwara-144411, Punjab.
Date**:**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Now a day, Multipliers plays an essential part in digital signal processing (D.S.P) and several other applications. In the high-performance systems like micro-processor, DSP etc. multiplication and addition is an important fundamental function and mostly used arithmetic logic operations. Basically, addition and multiplication are used in microprocessors and DSP is more than 68% instructions. So, the addition, multiplication operations control the execution time. That’s while there is need of high speed multiplier. Having with the advanced Technology, for designing a good multiplier, so many researchers have tried and trying. Multipliers are used basically for multiplication.

Usually shift and add method is not that much good multiplier and it is not appropriate for the VLSI implementation. The time delay point of also this multiplier is not good. There is some basic methods are proposed in the literature for the speed multiplication in the VLSI. Those are Wallace tree multiplier, Vedic multiplier and Baugh-Wooley multiplier. In this report we are present the several techniques to implement those multipliers. Basically In any VLSI circuit’s implementation is depends on the basic terms those are the speed, power, and time delay. The high speed and less power consumption VLSI circuits can be possible with several logic methods. In any VLSI design we seen three types most important factors, they are cell space, delay of the circuit and power consumption. Previously there are many methods for the speed and the power consumption and each method have the pros and cons for the VLSI circuit in terms of power and speed. The basic common multiplication method is adding& shift method. Basically, in the parallel multipliers, partial products are added are the important factors that can be deciding the multiplier whole performance. To reduce, this partial product to added, Booth multiplier is one of the better multiplier. For the speed purpose, Wallace tree method is better because the sequential adding steps.

And also we have, serial and parallel multipliers are there, in the area and power consumption factor also, it is good, in the speed factor this multiplier is not good. This serial and parallel multipliers are basically depends on the nature of the application. In this report, we are explained the multiplication algorithms and those method of architecture and comparing those methods in terms of the time delay, power, cell area, and speed of the multipliers.

1.1 Basic multiplication:-

As we all know multiplication basically having two terms, multiplicand and multiplier and output is product. Consider multiplicand is A(a0,a1,a2,a3) and multiplier is X(x0,x1,x2,x3) the output product is P=(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7).

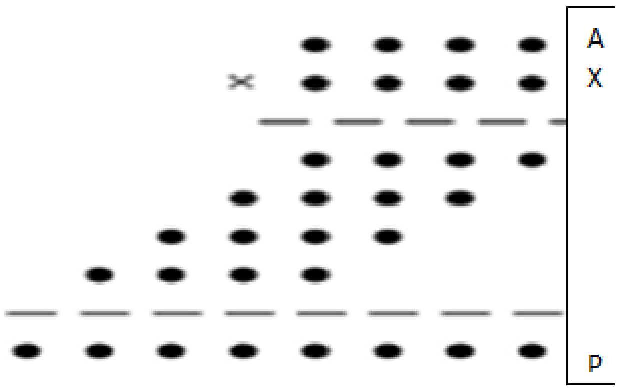


Fig.1.1. Basic multiplication structure

We have so many techniques to perform the binary multiplication. In that we choose based on the factors such as area, latency, design complexity, throughput. An efficient approach is use the array or full-adders tree to sum partial products. In, now a day we are using some standard designs used to implement the binary multipliers, those are suitable for the VLSI circuits.

Multiplier is performing multiplication process multipliers are mainly used in DSP application. Fast multipliers are mainly based on cell area, speed, power and accuracy.

- Area: In the multiplier cell area should be less
- Speed: the speed of multiplier should be fast.
- Power: the power should be less.
- Accuracy: the multiplier should give the correct result.

In any multiplication, there are mainly three steps:-

- ➢ Generation of the partial products.
- ➢ Reduction of the partial products.
- ➢ And final addition.

1.2 Main objectives of multipliers:-

- The good multiplier must be compact with the high speed and low power dissipation, timing delay.
- Mostly the designing a multiplier in VLSI based on these four factors only (time, speed, area, accuracy).

1.3 Classification of digital multipliers:-

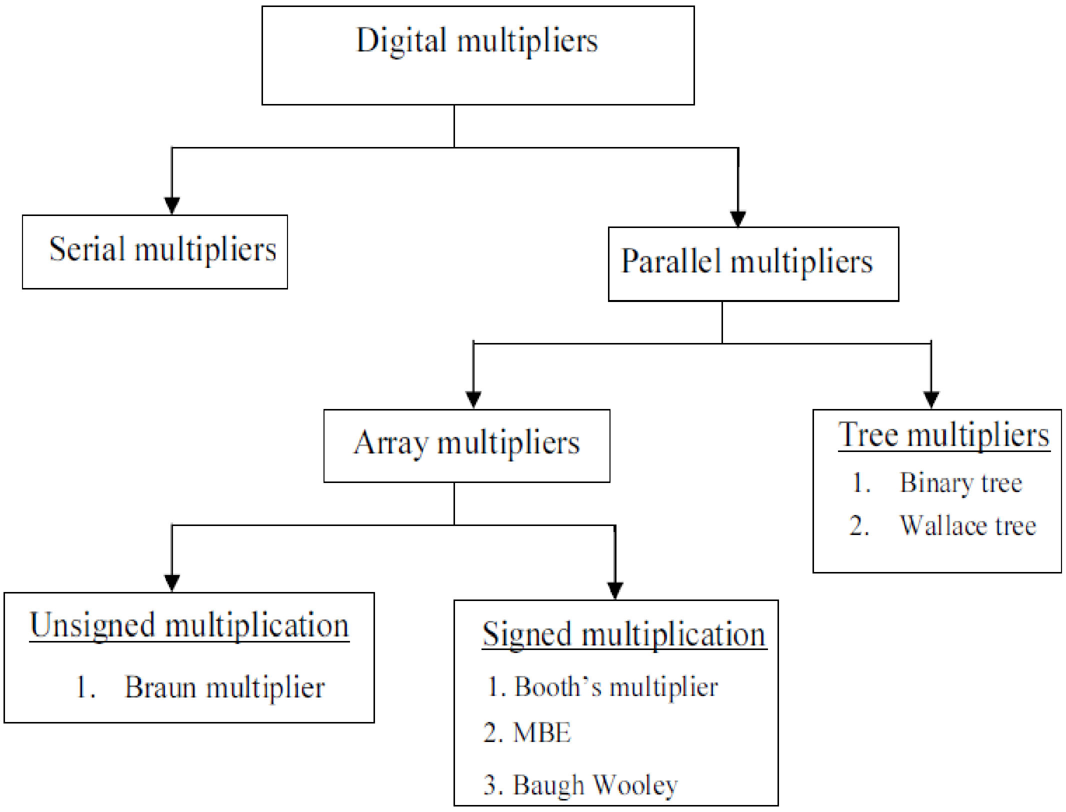


Fig.1.2. Classification of digital multiplier

1.3.1 Serial multipliers:-

Where the power and area is given importance and the delay is tolerated there the serial multiplier is used. Here they are used one simple adder to add the partial products. The circuit diagram is shown below for the 4bit multiplier. Multiplier and Multiplicand are the inputs have to arrange in a manner and synchronized with the behaviour of the circuit. And here two clocks are used, one of the clock for reset and another one for data. The first order of the time delay is U(x,y). in the circuit the time delay is T_D =[(x+1)y+1] t_{fa}..

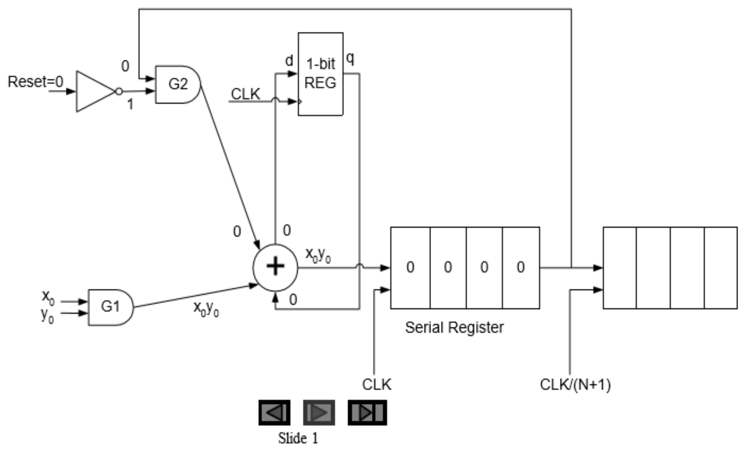


Fig.1.3. Serial multipliers

Here the separate partial products are formed individually. And the additions of the partial products are achieved by way of the intermediate values of partial products addition are put in storage in D-flip flop. This method is not proper for the large number of bits(X and Y).

1.3.2 Parallel multiplier:-

The basic architecture of the parallel multiplier is one of the best multipliers. First operand is served to the next circuit in the parallel. The partial products are made in each and every cycle. In each cycle can do the addition of multiplication of X*Y partial products. The last results are should be stored in the output register after X+Y cycles. And the area essential is Y-1 for X=Y.

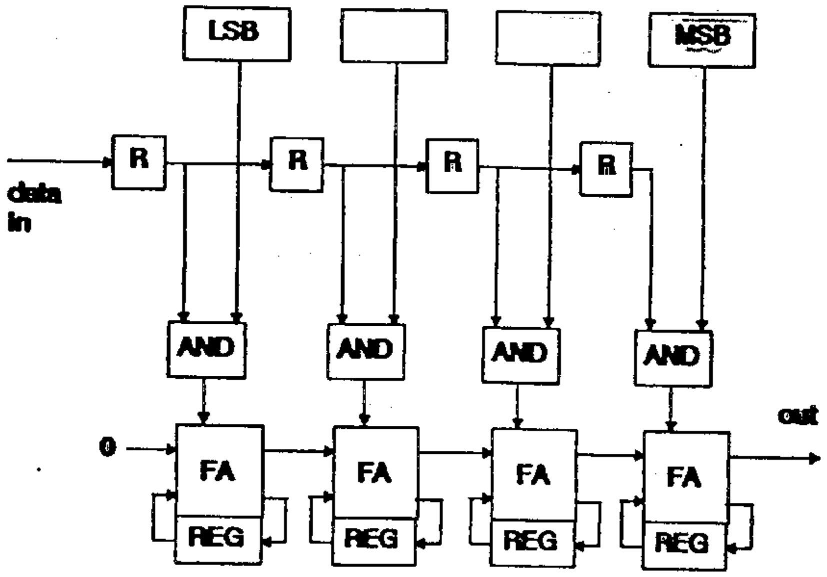


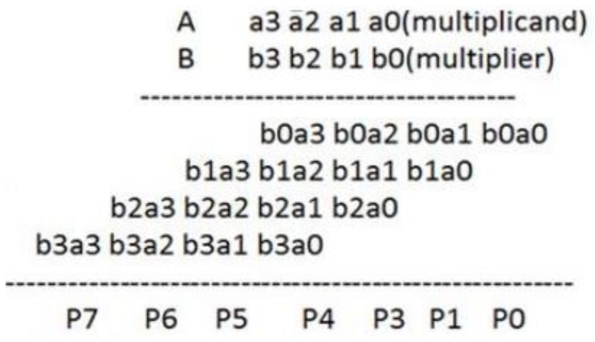
Fig.1.4. Parallel multiplier

The parallel multipliers are basically two types, they are

- Array multiplier
- Tree multipliers(Binary tree and Wallace tree)

1.3.3 Array Multiplier:-

The array multiplier is the well-known multiplier because of its basic structure. This multiplier is mainly based on the Add and shift algorithm. And the partial products are produced by the one bit multiplier is with multiplicand multiplication. The PP terms are shifted to the adder bits and after that should be added.



```
          A     a3 a2 a1 a0(multiplicand)
          B     b3 b2 b1 b0(multiplier)
        -----------------------------------
                 b0a3 b0a2 b0a1 b0a0
            b1a3 b1a2 b1a1 b1a0
       b2a3 b2a2 b2a1 b2a0
  b3a3 b3a2 b3a1 b3a0
------------------------------------------------
   P7    P6    P5    P4    P3   P1   P0
```

Fig.1.5 Basic array multiplier

In the array multiplier the partial product terms can be add by the simple carry propagate adder. If the N is the length of the multiplier we need N-1 adders. And here we use number of adders are x, full adders are x(y-2), the total number of adders are x(y-1).

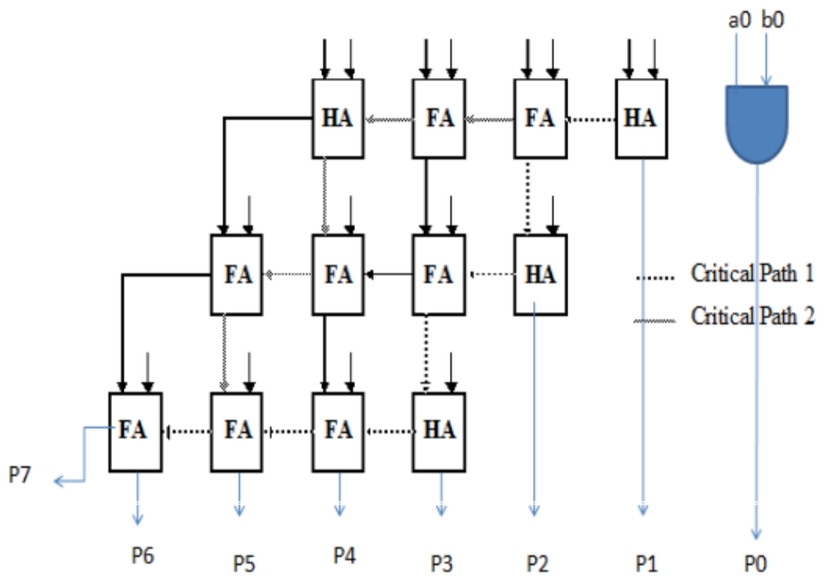


Fig.1.6. Circuit diagram of array multiplier

The array multiplier further divided into two types

- Signed multiplication
- Un Signed multiplication

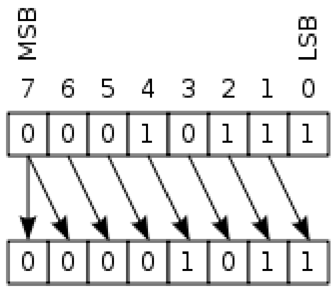
In the signed multiplication booth and baugh wooley multipliers are the best multipliers.

1.3.4 Booth multiplier:-

The Booth multiplication is the one of the multiplication which multiplying the binary numbers in the 2's complement signed array representation.

Before doing the booth multiplier we need to know the right shift arithmetic (RSA) and right sift circular (RSC).

Right shift arithmetic (RSA):- it is a shift operator. It is defined, when we are adding two binary numbers and the result is shift to the one bit position. Let's take the resultant bit is 1010. Now we can apply RSA then the result is 11010. In RSA there are two basic types 1. Right shift arithmetic and 2. Left shift arithmetic.



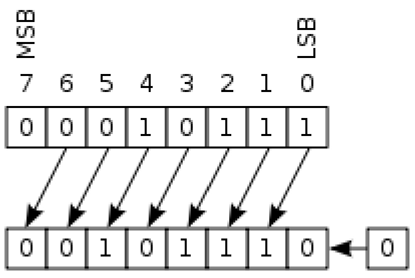
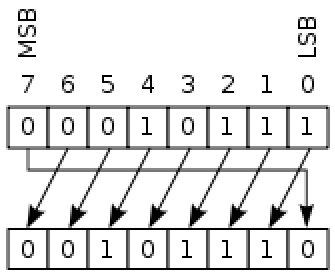


Fig.1.7. Right shift arithmetic Left shift arithmetic

Right shift circular(RSC):- It is simple shifting the bit, in RSC also two types they are Right circular shift and Left circular shift. Let's take one sequence 01011 and the RSC of the sequence is 01011.



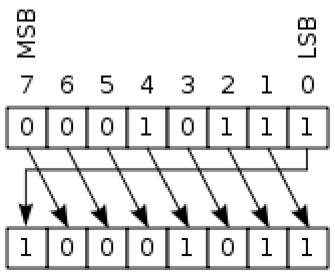


Fig.1.8. Right shift circular Left shift circular

Steps to implement the booth algorithm:-

Step 1:-

Create the Booth Table: In this table, we will take the 4columns for the one is for the multiplier(X), and one is for the previous bit of multiplier(X-1), and other two for partial products (U and V).

U	V	X	X-1

Table1.1. Basic Booth table

- First we choose multiplier X and the multiplicand is Y.
- And next to do the 2's complement for the multiplicand Y.
- Load X value, and X-1 value is kept =0.
- And the U and V values initially taken as 0, these will have the product of the X & Y at the last resultant.

Step 2:-

In the table, the LSB nit of the X and the single bit in the X-1, is to be analyse that will have fallowing actions.

If the action is = "00" then there is no action.

If the action is = "01" then add Y and U, the result is Right shifts.

If the action is = "10" then subtract Y from U and do the result is right shifts.

If the action is = "11" then right shift the value in U one bit position.

Step 3:-

Right shift circular of X. Go to step 2 and repeat the method till the X has been right shift circular to its original position. Finally we get the product of X and Y.

1.3.5 Combinational multiplier:-

The combinational multipliers are the, two unsigned binary numbers multiplication. And this multiplier is done with the two signed number multiplication also. Every bit in the multiplier is multiplied to the multiplicand. The final product is come when we add the partial products and then it form a final result.

This multiplier main advantage is the generation of the partial products are easy compare to other multipliers. The basic circuit diagram shown below, in there partial products can be adding by using AND gate. And here using half adder and full adder also and finally we get the result of the combinational multiplier.

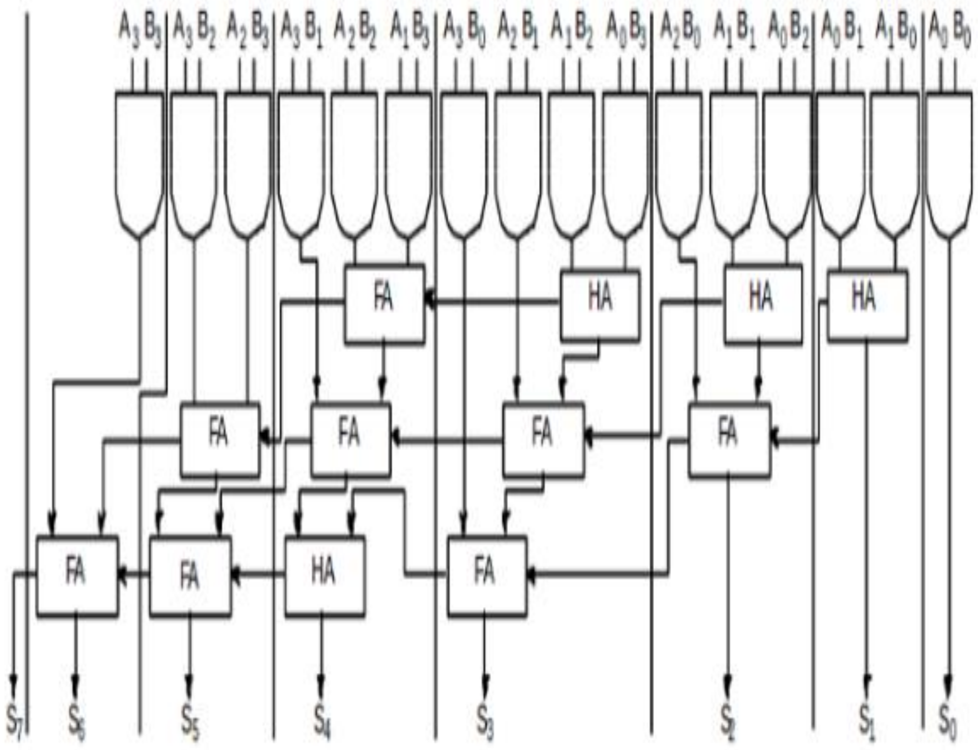


Fig.1.9. combinational multiplier

This multiplier good in terms of the power consumption and it can need fewer components. But in terms of time delay factor this multiplier is not good. It requires large number of logic gates so area is also high. It is a less economical multiplier and faster multiplier but high hardware complexity.

CHAPTER 2

REVIEW OF LITERATURE

The chapter is focused on the review of literature about different fast multipliers designs. To conclude this topic so many journals, articles and conference papers have been studied. Some of them have been described below as.

R. Raju, S.Veerakumar (2016), "Design and Implementation of Low Power and High Performance Vedic Multiplier" [1]In this paper authors, main aim is to designing and developing a high speed, less power dissipation of a16bit vedic multiplier by using basic 8bit vedic multiplier, 4bit vedic multiplier and the basic 2bit multiplier. For adding the partial products here author used ripple carry adder to decrease the time delay in the multiplier. Here the 16bit Vedic multiplier is done by using "Urdhva Tiryakbhyam Sutra" from ancient Indian Vedic mathematics. Author mainly focused on, to reduce the logic levels thus reducing the logic delays. The entire process is done by using Xilinx-ISIM and synthesis done by using Xilinx XST. And the total execution is done in the FPGA (Spartan-kit). Finally the author concluded even though Urdhva Tiryakbhyam Sutra fast and area efficient. But the large number of partial products occurs by using of 2bit and 4bit and so on. And also large fan-out for input signals x and y. by using the other algorithms in 4x4 multipliers then faster multiplication is possible.

Rakesh Kumar, Pradeep Kumar (2014), "An Efficient Baugh-Wooley Multiplication Algorithm for 32-bit Synchronous Multiplication" [2]In This paper, all about the effective and high speed 32bit synchronous Baugh wooley multiplier. In this paper mainly the multipliers partial products can be added by the fast speed and area less adder named as BK (Brent-kung) adder. By using this adder not only improves speed of adder performance but also improves the multiplier. Codeing in VHDL and synthesis was done by using Xilinx ISIM and synthesized by Xilinx XST for the both synchronous and asynchronous Baugh wooley multiplier. Finally the author concluded that the Look up tables (LUT's) are less in asynchronous as compared to the synchronous BW multiplier. But synchronous multiplier is faster than the asynchronous BW multiplier. The path delay in the synchronous BW multiplier is 6.496ns, it is good result when compared to the other multipliers. Finally he resulted that the combination of synchronous BW multiplier and Brent-kugh adder, gives the better speed and less area compared to the all other multipliers.

Kokila Bharti Jaiswal, Nithish Kumar V, Pavithra Seshadri (2015): "Low Power Wallace Tree Multiplier Using Modified Full Adder" [3]Authors main aim to designing a less power consumption of multiplier, here they are taken Wallace tree multipliers with modified fulladder using multiplexer. The whole process is done in Verilog HDL and simulation is done by using Quatus II. And the whole process is synthesized by using SAED90nm CMOS technology in Synopsys Design complier. Finally author concluded In the ASIC synthesis of the Wallace tree Multiplier by using mux based fulladder the results shows in terms of power consumption average reduction is 37.45%, in terms of area 45.75% and the time delay average reduction is 17.65% compared to the all current methods finally the proposed Wallace tree multiplier is good for the application which we need less power and lesser area applications.

Indrayani Patle, Akansha Bhargav, Prashant Wanjari(2013): "Implementation of Baugh-Wooley Multiplier Based on Soft-Core Processor" [4]the author says in this paper, they done execution of 16bit Baugh wooley multiplier in Verilog HDL. And the multiplier based on the soft core processor. This is embedded soft core processor with high performance by XILINX Company. This soft core processer is high configurable and the designer to design required own hardware platform. Baugh wooley multiplier is using in this processer to utilized fast and efficient processing capacity. Finally they are concluded increasing the speed of the custom hardware of multiplier block designed and interface with MicroBlaze processor. And also power optimized in the 16bit baugh wooley multiplier is optimized, and the power is 163mW. And they are planning to implement 32, 16, 8-bit FFT by using this fast and less power Baugh wooley multiplier.

Ms. G. R. Gokhale, Mr. S. R. Gokhale, (2015) "Design of Area and Delay Efficient Vedic Multiplier Using Carry Select Adder" [5]In this paper authors, main aim is to designing and developing a less area, less delay of a vedic multiplier. For adding the partial products here author used carry select adder, to decrease the time delay in the multiplier. The carry select adder(CLSA) is used because the less number of gates used compared to the binary to excess one convert(BEC) carry select adder and modified carry select adder(MCSLA). The resultant in terms of area, carry select adder is 21% small area than modified carry select adder and 44% small area than binary to excess one convert carry select adder. The Vedic Multiplier is 43% small area than Booth multiplier. And in terms of time delay 15% less compared to booth multiplier. In the parameter, area and time delay the proposed Vedic Multiplier is similar to the booth multiplier. Finally author says proposed Vedic multiplier is better in area, speed and delay.

Soniya, Suresh Kumar, (2013) "A Review of Different Type of Multipliers and Multiplier-Accumulator Unit" [6]In this paper authors discuss about fast speed, less power multiplier accumulator unit, and several types of multipliers those are used in the DSP applications(FFT, FIR, Convolution)etc. in this authors explain several multipliers array , Wallace tree, booth, sequential and combinational multipliers. In those multipliers several types of techniques applied to check the multiplier is speed and less power consumption. The applied techniques are pipelined technique, Spurious Power Suppression Technique (SPST) technique and block enabling technique. In the pipelined technique is used in booth multiplier to decrease the time delay in every stage. In the SPST Tech. is used to remove the useless portion of data for decrease the power. Enabling technique is also used to reducing the power. Finally authors concluded in terms of speed, pipeline technique is better for booth and Spurious Power Suppression Technique (SPST) and enabling technique is better in terms of less area and power.

Abhishek Mukherjee, Abhijit Asati, (2013) "Generic Modified Baugh Wooley Multiplier" [7**]**in this paper, they are done with the HDL code of Baugh Wooley multiplier. The main aim of the author is comparing the conventional Baugh Wooley multiplier and simple default multiplier and modified multiplier. In conventional Baugh-Wooley multiplier they are added partial products using ripple carry adder, and it can be replace again in modified Baugh- Wooley multiplier with the carry select adder, and the calculate the parameters are the area, power, speed of the multiplier. And the whole process is done is done by synthesis results and they are taken operand size ranging N is (4 to 60) using in 90nm technology. Finally the author concluded the modified Baugh Wooley multiplier is better speed than the conventional multiplier and as well in terms of area and power similar to the conventional and simple default multiplies according to the synthesis report.

Sumit Vaidya, Deepak Dandekar, (2010) "Delay-Power Performance Comparison of Multipliers in VLSI Circuit Design" [8]In this paper, they described the comparisons of the Vedic multiplier with the other multiplier like Wallace tree multiplier and array multiplier. And the Vedic multiplier is done with two sutras "Urdhva Tiryakbhyam" algorithm and another sutra is "Nikhilam Sutra". The author briefly explained about Wallace tree and array multiplier. Here also compared the multipliers are in terms of speed, area, time delay and power factors. By using the nikhilam sutra we can get the faster multiplication than the Urdhva Tiryakbhyam algorithm of the multipliers, because the speed of the multiplier is increasing by reducing the number of iterations. The time delay comparison is done with array and booth multiplier in 8 bit and also in 16 bit, in the array multipliers time delay is 47ns, 92ns and in booth multiplier 117ns, 232ns and in the vedic multiplier in 8bit 27ns and in 16bit 39ns. And they are compared multiplier in three different logics. They are CMOS and complementary pass transistor and pass transistor logics.

Pramodini Mohanty, (2013) "An Efficient Baugh-Wooley Architecture for Signed & Unsigned Fast Multiplication" [9**]**This paper presents the good, efficient, and high speed multiplier with the method shift & add method for Baugh-wooley multiplier. In this baugh wooley multiplier we are suing less adder and then we can iterative steps are less. The area is also less compared to the serial multipliers. This multiplication good because in the fabrication the chips, and the good systems need less components circuits. The results are saying the proposed circuit is correct performance and less hardware components and low power. The dynamic power they get 15.3mW and the timing delay is 3.912ns. they get best results compare to their base paper, For improving the multiplier characteristics they are using pipelining resistor technique.

Amrita Nanda, Shreetam Behera, (2014) "Design and Implementation of Urdhva-Tiryakbhyam Based Fast 8×8 Vedic Binary Multiplier" [10**]**in this paper , author design a high speed 8bit multiplier by using the Indian ancient Vedas, vedic mathematics, there so many multiplication techniques, one of that is Urdhva-Tiryakbhyam sutra from the vedic mathematics. First he proposed 4bit Vedic multiplier by using the four bit adder to reduce the time delay. And using this 4bit Vedic multiplier, authors designed a 8bit vedic multiplier with using the fast adders. The time delay is good in this multiplier compare to the array, booth multiplier. The multiplier is done by, VHDL coding. And the synthesis is done by using the Xilinx ISE14.4 Software. Finally they applied this code in FPGA Spartan 3e board. Finally author says that the designed multiplier is good in terms of time delay.

Taye Girma, (2013) "Designing and Synthesizing a Wallace Tree Multiplier for High Speed Performance"[15] In this paper, describes the designing and synthesis of the 8bit Wallace tree multiplier. Theses multipliers are used basically in DSP applications and microprocessors. The basic operation of the Wallace tree multiplier is addition of partial products. Here a new algorithm for the Wallace tree multiplier. 3stages to do this multiplier they are PP matrix generated, and next stage is reduce the PP terms by using the fulladder and half adder. And here author used carry look-ahead adder. The time delay is reducing, due to the route, logic gates. Finally the author concluded that the presented method is efficient for speed multiplication.

Pramod S. Aswale, Mukesh P. Mahajan, Manjul V. Nikumbh, Omkar S. Vaidya, (2015), "Implementation of Baugh-Wooely Multiplier and Modified Baugh Wooely Multiplier Using Cadence (Encounter) RTL" [16]In this paper, authors describe the less power and speed is high by using the shift and add algorithm using of baugh wooley multiplier. And here done with 5bit baugh wooley multiplier using cadence RTL compiler. And he concluded that the modified baugh wooley is better than the conventional baugh wooley multiplier, and the 5bit BW multiplier operating frequency is 160Mhz. this multiplier depends upon the required application. The author concluded that the modified baugh wooley is 109 x speeds than the conventional array multiplier. And 102x speeds than the conventional baugh wooley multipler

CHAPTER 3
COMPLEX MULTIPLIERS

3.1 VEDIC MULTIPLIER

Vedic multiplier is comes from the Vedic mathematics and It is the ancient Indian method of mathematics. This was reconstructed from Vedas by Sri Bharati Krishna Tirthaji (1884-1960) after his 8years of analysis on Vedas. He proposed Vedic mathematics is mostly depends on sixteen sutras, in one of that is Urdhva-Tiryakbhyam sutra.

Sixteen sutras in Vedic mathematics:-

1. Yaavadunam (At all the extent of its lack).
2. Vyashtisamanstih (Share and Complete).
3. Urdhva-Tiryakbhyam (Vertically and crosswise).
4. Sopaantyadvayamantyam (The critical and double the penultimate).
5. Shunyam Saamyasamuccaye (if the sum is the same then the sum is zero).
6. Sankalana-vyavakalanabhyam (by addition& by subtraction).
7. Shesanyankena Charamena (The remainders by the last digit).
8. Paraavartya Yojayet (Transposeing & adjust).
9. Puranapuranabyham (it is the completion / Non-completion).
10. Gunitasamuchyah (The POS is equal to the SOP).
11. Nikhilam Navatashcaramam Dashatah (All from nine & last from ten).
12. Ekanyunena Purvena (By one < the before one).
13. Gunakasamuchyah (all The factors of the sum = sum of the factors).
14. Chalana Kalanabyham (Differences & Similarities).
15. Ekadhikina Purvena (By one higher than the before one).
16. (Anurupye) Shunyamanyat (If one thing in ratio, the other should be zero).

3.1.1 Urdhva-Tiryakbhyam Sutra:-

The Vedic multiplier is mainly depends on Urdhva Tiryakbhyam sutra from the Vedic mathematics. This sutra is also called (vertical and crosswise) sutra. This sutra is basically used for the multiplication of two decimal numbers. And this method used for binary numbers also. The process of vertical and crosswise algorithm is firstly the least significant bits(a0,b0) are multiplicand which gives the least significant bit of the end product(Vertical).Carry will add to the second multiplicand. In second step a0, b1 act as crosswise multiplicand and a1, b0 act as vertical multiplicand. Here also carry will add to third step. In every step the carry will add to the next step of the algorithm. This process is going on till we get the final product.

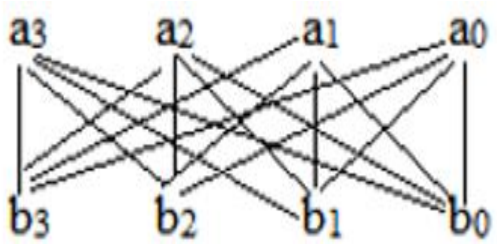


Fig.3.1. Vertical and crosswise algorithm

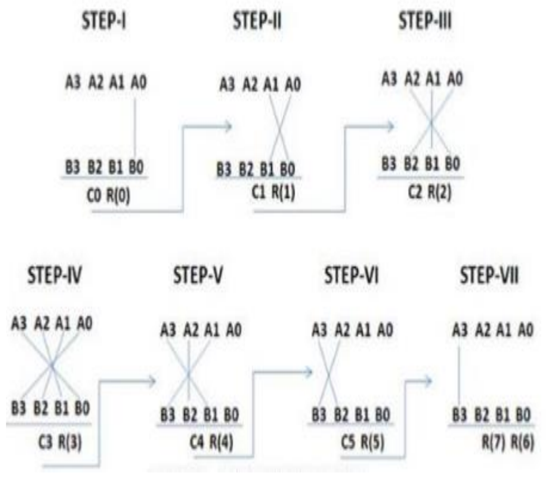


Fig.3.2 Example for the vertical and crosswise algorithm

By using half adder and full adder we can add the partial product terms. The Basic gate-level architecture of Vedic Multiplier

3.1.2 Vedic 2x2 bit Multiplier:-

In the 2bit Vedic Multiplier let us take two bit numbers y and z; here y is (a1, a0) and z is (b1, b0). In this multiplication firstly the LSB bit is multiplied and it gives the LSB of the final product in vertical. And multiplicand is multiplied with the highest bit of the multiplier and that can be included with the result of least significant of multiplier and the coming bit of multiplier of crosswise. This process we can do until get the final product of the multiplier.

Sum= a0b0; Carry1sum1= (a1b0 + a0b1) ; Carry2sum2= (c1+ a1b1)

The final product will be P= (carry2.sum2.sum1.sum0).

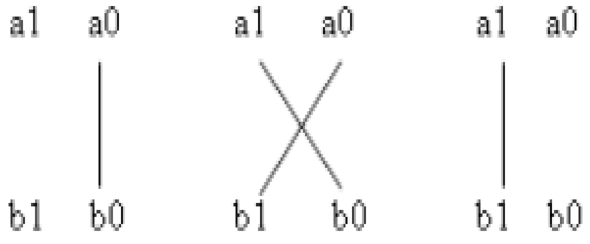


Fig.3.3 2bit Vedic multiplication method

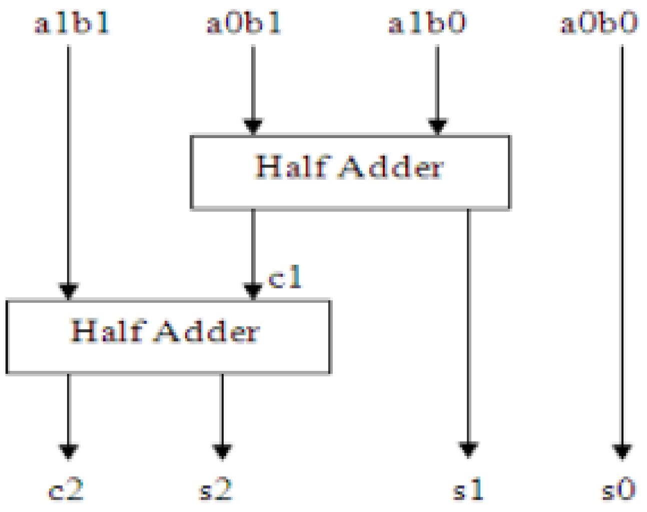


Fig.3.4. Block diagram of 2bit Vedic multiplier

3.1.3 Vedic 4bit Multiplier:-

In the 4bit Vedic multiplier let us take two bit numbers y and z; here y is (a_3, a_2, a_1, a_0) and z is (b_3, b_2, b_1, b_0). In this multiplication firstly the LSB bit is multiplied and it gives the LSB of the final product in vertical. And multiplicand is multiplied with the highest bit of the multiplier and that can be included with the result of least significant of multiplier and the coming bit of multiplier of crosswise. This process we can do until get the final product of the multiplier.

The final product will be P= (p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0)

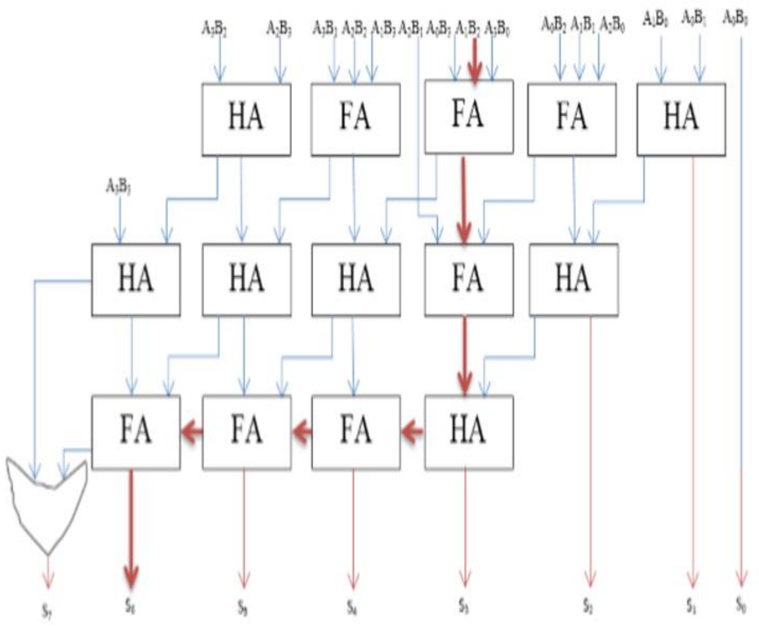


Fig.3.5. Block diagram of 4bit Vedic multiplier

3.1.4 Vedic 8x8 multiplier:-

In the 4bit Vedic multiplier let us take two bit numbers y and z; here y is (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0) and z is (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). In this multiplication firstly the LSB bit is multiplied and it gives the LSB of the final product in vertical. And multiplicand is multiplied with the highest bit of the multiplier and that can be included with the result of least significant of multiplier and the coming bit of multiplier of crosswise. This process we can do until get the final product of the multiplier.

The final product will be P= (p_{15}, p_{14}, p_{13}, p_{12}, p_{11}, p_{10}, p_9, p_8, p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0)

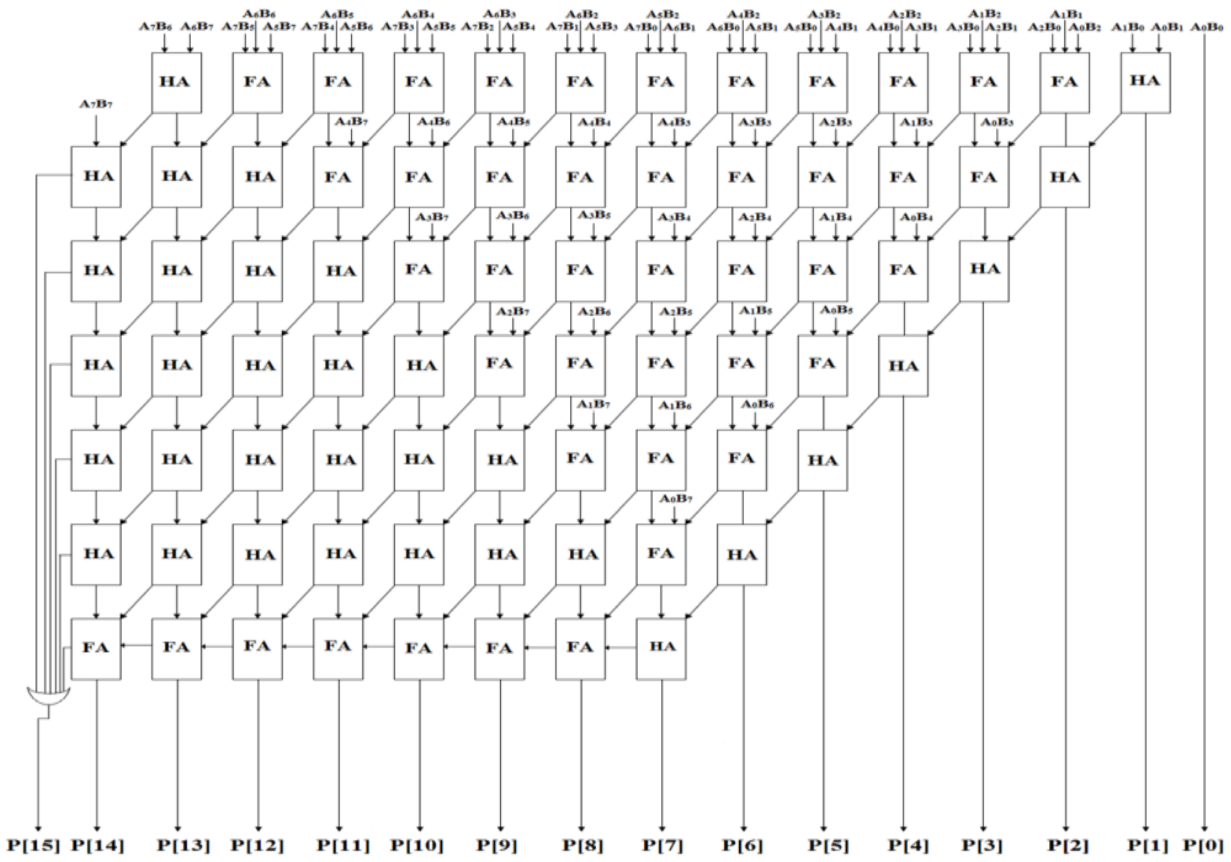


Fig.3.6. Block diagram of 8bit Vedic multiplier

3.2 WALLACE-TREE MULTIPLIER

A Wallace tree multiplier is unique and one of the best method to design a digital circuit that multiplies two digital numbers. Wallace tree multiplier is based on Wallace tree algorithm. In 1964 Australian scientist ChirsWallace introduced this Wallace tree algorithm. Wallace tree multiplier is basically done by three steps.

- Step1 is Generating of partial products,
- Step2 is grouping and reducing a partial product,
- Step3 is Final addition.

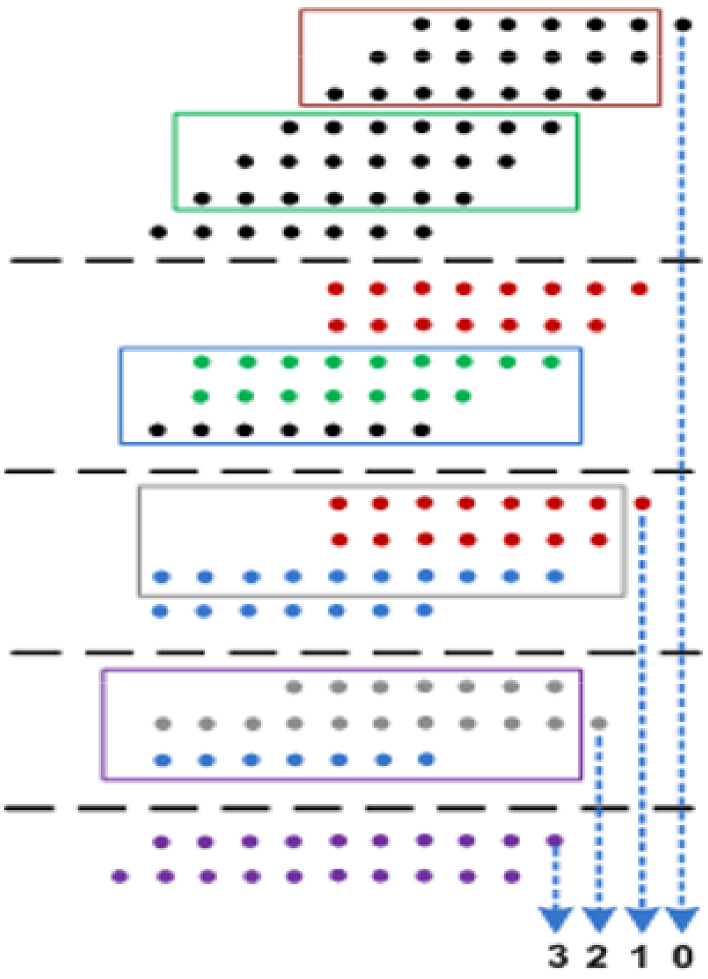


Fig.3.7. Wallace- Tree algorithm

3.2.1 Wallace tree 4x4 bit multiplier:-

The Wallace tree algorithm is the three single bit signals are added by the full-adder and output sum is given to the following stage fulladder of same bit and the output carry is given to the following stage fulladder of a one bit higher position. In 4x4 multiplier, lets taken multiplicand as a (a_3, a_2, a_1, a_0), and multiplier as b (b_3, b_2, b_1, b_0). The Final output (multiplication) is P= (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0).

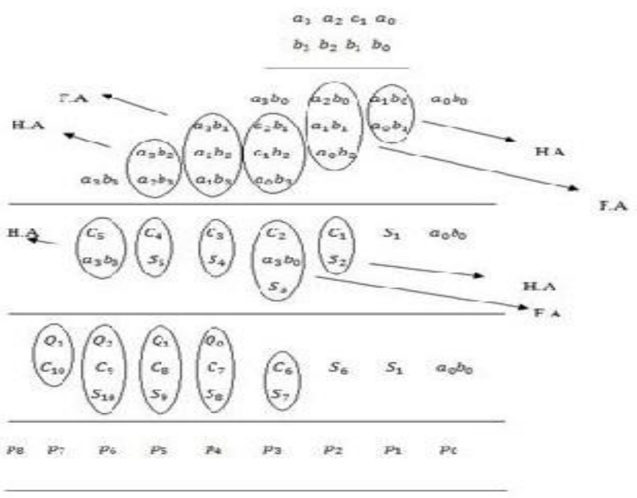


Fig.3.8. 4bit Wallace tree algorithm

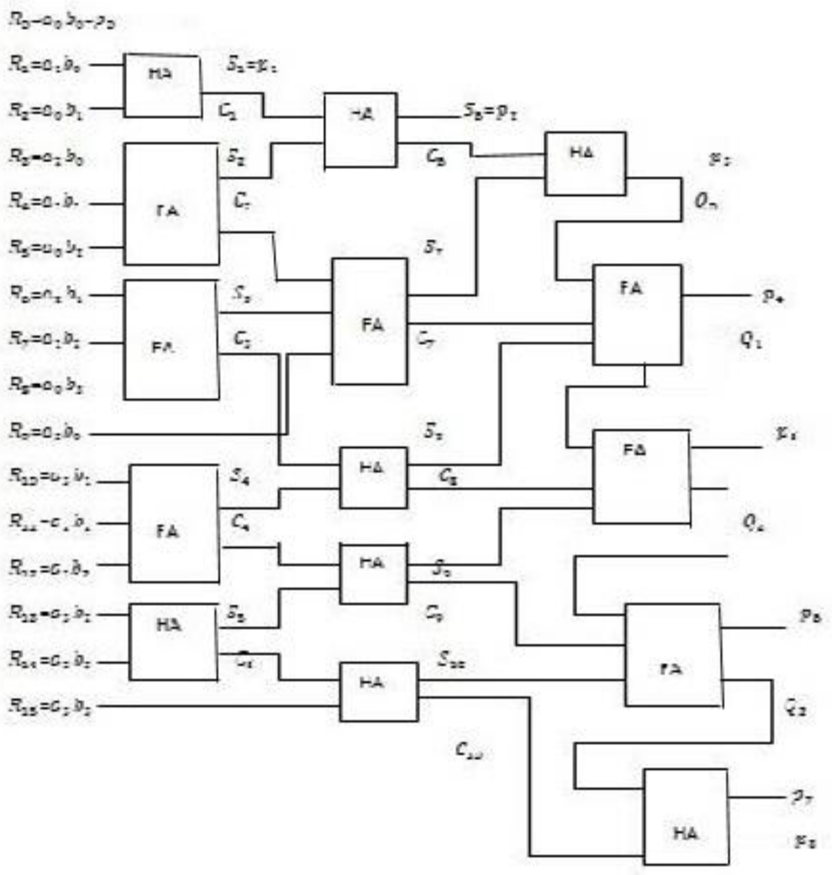


Fig.3.9. Block diagram of 4bit Wallace tree multiplier

After generation of partial products, we will do the grouping and reduction. The grouping of partial products and they are adding by using full adder and half adder. We can continue the process until we get the result P= (q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0).

3.2.2 Wallace tree 8x8 bit multiplier:-

The Wallace tree algorithm is the three single bit signals are added by the full-adder and output sum is given to the following stage fulladder of same bit and the output carry is given to the following stage fulladder of a one bit higher position. In 8x8 multiplier, lets taken multiplicand as a (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0), and multiplier as b (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). The Final output (multiplication) is P= (p_{15}, p_{14}, p_{13}, p_{12}, p_{11}, p_{10}, p_9, p_8, p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0).

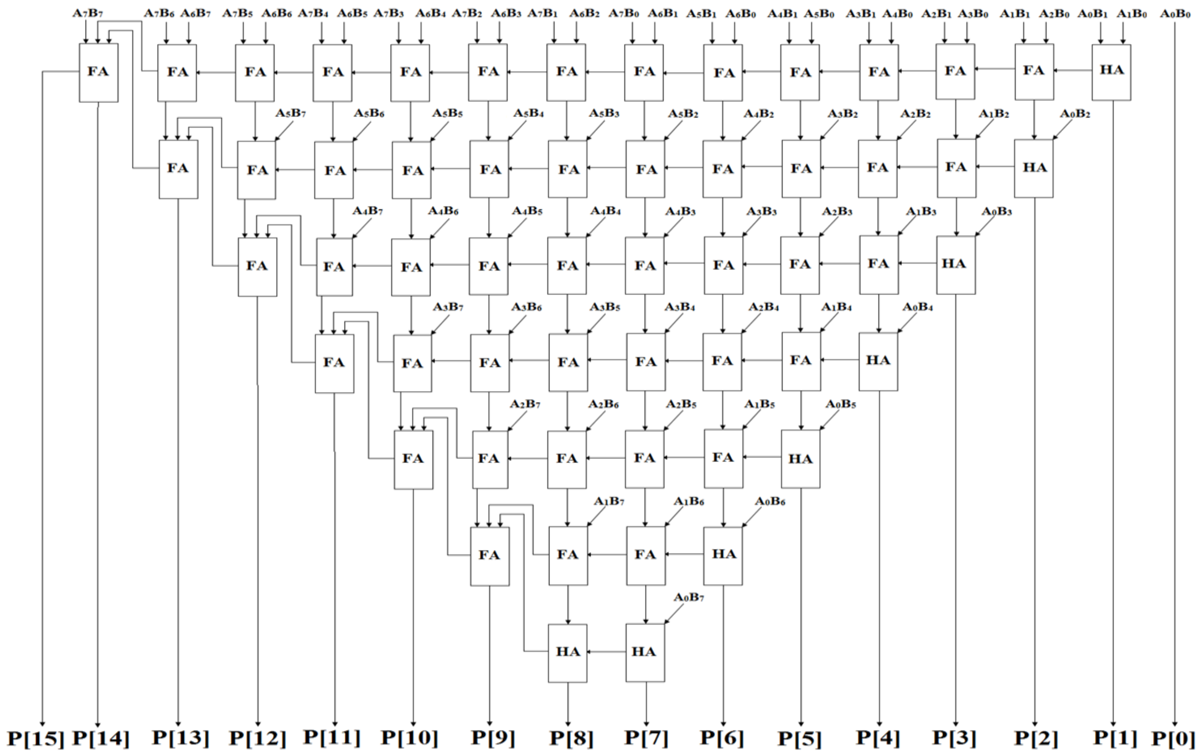

FA
HA
P[15] P[14] P[13] P[12] P[11] P[10] P[9] P[8] P[7] P[6] P[5] P[4] P[3] P[2] P[1] P[0]

Fig.3.10. Block diagram of 8bit Wallace tree multiplier

3.3 BAUGH WOOLEY MULTIPLIER

This was proposed by Baugh and Wooley which is the method for direct 2's complement Array multiplication. The indications of all summands are positive this is the main advantage of this method, therefore permitting the array to be developed completely with the normal full adders.

```
                0 1 1 0 1   -13
  X             0 0 1 0 1   - 5
  -------------------------
                0 1 1 0 1
              0 0 0 0 0
            0 1 1 0 1
        0 0 0 0 0 0
        1 0 0 0 0
        1       0
+ 1             0
-----------------------------
  0 0 0 1 0 0 0 0 0 1   - 65
```

For unsigned numbers

```
              1 0 0 1 1   - -13
  X           1 1 0 1 1   - -5
  -------------------------
              0 0 0 1 1
            0 0 0 1 1
          1 0 0 0 0
      1 0 0 0 1 1
      0 1 1 0 0
      0       1
+ 1           1
-------------------------
  0 0 0 1 0 0 0 0 0 1   - 65
```

For signed numbers

This uniform structure is exceptionally attractive for VLSI. Baugh-Wooley Multiplier is utilized for the both signed unsigned numbers. The Signed Number operands which are spoken to in 2's complemented form. Incomplete Products are balanced with the end step that negative sign move to last stride, which thusly augment the consistency of the multiplication exhibit. Baugh-Wooley Multiplier works on signed operands with 2's complement representation to ensure that the indications of every fractional are positive.

3.3.1 Baugh wooley 4x4 multiplier:- In the 4bit multiplier The Signed Number operands which are spoken to in 2's complemented form. Incomplete Products are balanced with the end step that negative sign move to last stride, which thusly augment the consistency of the multiplication exhibit. Baugh-Wooley Multiplier works on signed operands with 2's complement representation to ensure that the indications of every fractional are positive.

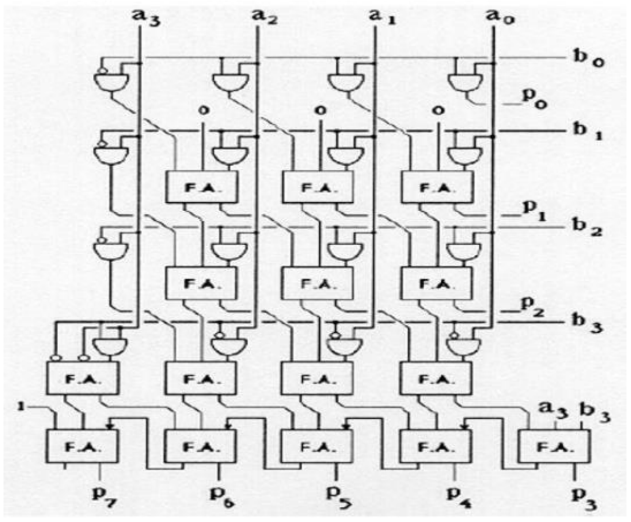


Fig.3.11. Architecture of 4bit baugh wooley multiplier

3.3.2 Baugh Wooley 5x5 Multiplier:

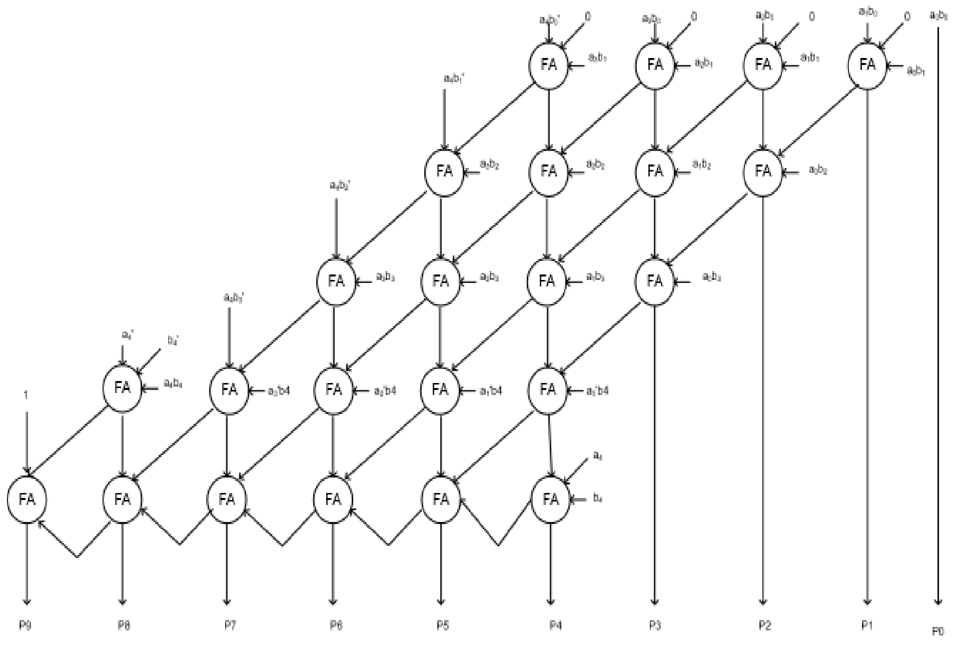


Fig.3.12. 5x5 bit Baugh-Wooley 2’s complement multiplier

```
                    0  1  1  0  1    =13
   X                1  1  0  1  1    = -5
   ---------------------------------
                    0  1  1  0  1
                 0  1  1  0  1
              0  0  0  0  0
       0  0  1  1  0  1
       1  0  0  1  0
       0           0
 +  1              1
   ---------------------------------
    1  1  1  0  1  1  1  1  1  1    = -65
```

```
                    1  1  0  1  1    = -5
   X                0  1  1  0  1    =13
   ---------------------------------
                    0  1  0  1  1
                 1  0  0  0  0
              0  1  0  1  1
       0  0  1  0  1  1
       0  0  1  0  0
       1           1
 +  1              0
   ---------------------------------
    1  1  1  0  1  1  1  1  1  1    = -65
```

```
                    0  1  1  0  1    =13
   X                0  0  1  0  1    = 5
   ---------------------------------
                    0  1  1  0  1
                 0  0  0  0  0
              0  1  1  0  1
       0  0  0  0  0  0
       1  0  0  0  0
       1           0
 +  1              0
   ---------------------------------
    0  0  0  1  0  0  0  0  0  1    = 65
```

```
                    1  0  0  1  1    = -13
   X                1  1  0  1  1    = -5
   ---------------------------------
                    0  0  0  1  1
                 0  0  0  1  1
              1  0  0  0  0
       1  0  0  0  1  1
       0  1  1  0  0
       0           1
 +  1              1
   ---------------------------------
    0  0  0  1  0  0  0  0  0  1    = 65
```

Fig.3.13. Example of the Baugh-wooley multiplier

3.3.3 Baugh wooley 8x8 multiplier :-

In the 8x8 bit multiplier The Signed Number operands which are spoken to in 2's complemented form. Incomplete Products are balanced with the end step that negative sign move to last stride, which thusly augment the consistency of the multiplication exhibit. Baugh-Wooley Multiplier works on signed operands with 2's complement representation to ensure that the indications of every fractional are positive.

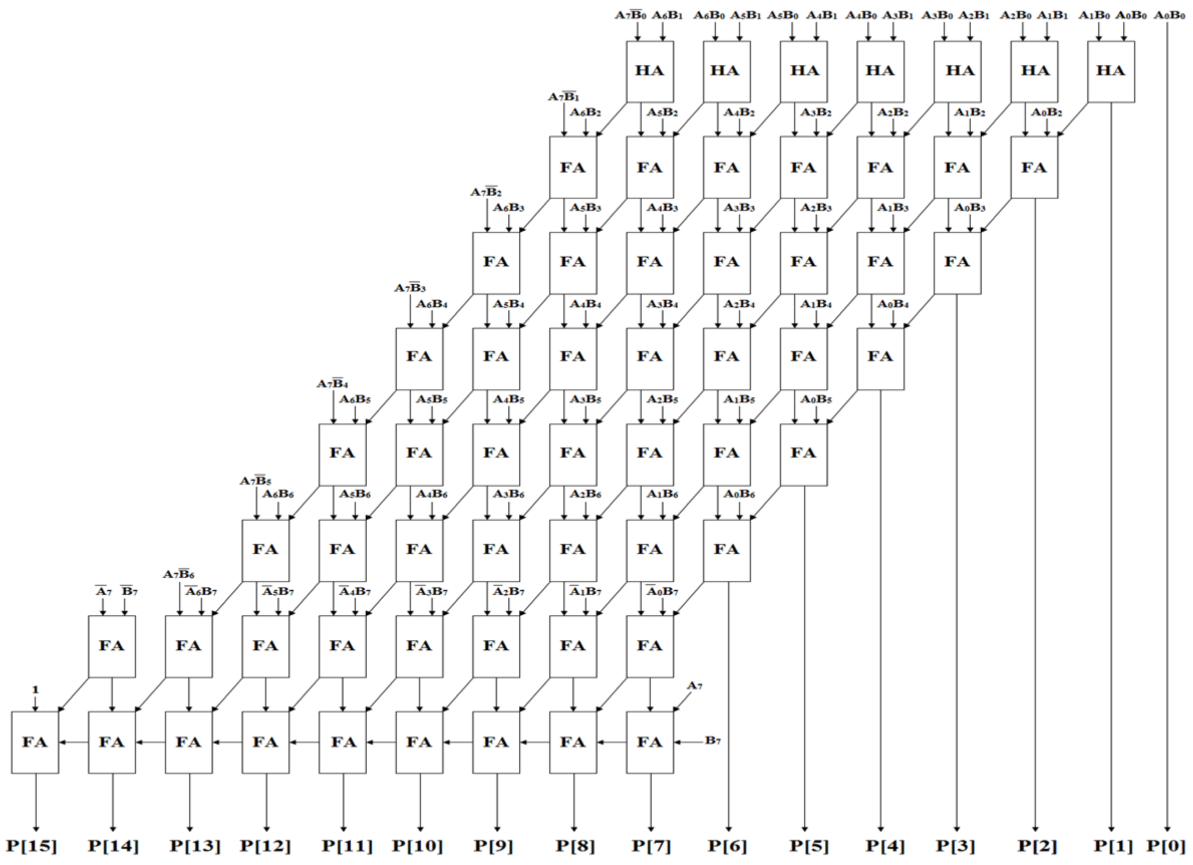


Fig.3.14. Architecture of 8bit baugh wooley multiplier

CHAPTER 4
OBJECTIVE AND SCOPE OF STUDY

4.1 Objective of the study:-

- To design 4,8-bit Vedic multiplier, Wallace tree multiplier, Baugh wooley multiplier.
- To design Verilog code i.e., main module and test bench for all the three multipliers.
- To implement the code in Xilinx and Cadence NCsim and generate the waveform and RTL on different technologies.
- To analysis the Timing, Area, Power on 45nm, 90nm, 180nm technologies.
- To compare the different parameters for different technologies.

4.2 Tools/ Software are used:

Xilinx, Cadence NCsim

4.3 Technologies used:

45nm, 90nm, 180nm.

4.4 Scope of the study:-

In the digital world, Multipliers are the most important components of the central processing unit. Multipliers are required in Athematic and logical unit (ALU)s. For the calculating memory address and the floating point calculation in the multiplication. Multipliers are also very important components in the digital signal processing (DSP), microprocessors. As day by day, digitalization is increasing in every application, so that the speed of the processors is increasing. So, to fulfil the demand of digitalization we need adders which provide us accurate outputs with very less consumption of the constraints given below.

The speed and the accurate results of a digital system are mostly influenced by the operation of respective multipliers. The main constrain for the designing of multiplier are the area, power, speed, time delay.

So, multipliers with optimized area, power efficient i.e., consumes very much less power, high speed and also performing the operation using the less number of cells.

CHAPTER 5
RESEARCH METHODOLOGY

5.1 PROBLEM FORMULATION:

As day by day digitalization is increasing in every application and so that the speed of processors are increasing. So, to fulfil the demand of digitalization we need multipliers which provide us accurate outputs with very less consumption of the constraints given below.

The speed and the accurate results of a digital system are mostly influenced by the operation of respective multipliers. The main constrain for the designing of multiplier are the area, power, speed, time delay.

5.2 DESIGN APPROACH:

For designing fast multipliers, extensive literature survey was done. To implement them we need Verilog code i.e., main module and test bench. After getting the module for respective multiplier we tried to simulate the module in Xilinx and cadence NCsim tool, at this time we get the simulation result for different input combination. For synthesis of delay and power we use fast.lib and slow.lib where we check the delay and power at 45nm, 90nm, 180nm technologies. So, finally our main aim is to compare these techniques at the different technologies.

CHAPTER 6
RESULTS AND DISCUSSION

Simulation and Synthesis Results of Multipliers: -

6.1 VEDIC MULTIPLIERS:-

6.1.1 Vedic 4bit Multiplier:-

Vedic multiplier has been implemented in Xilinx and NC simulation using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

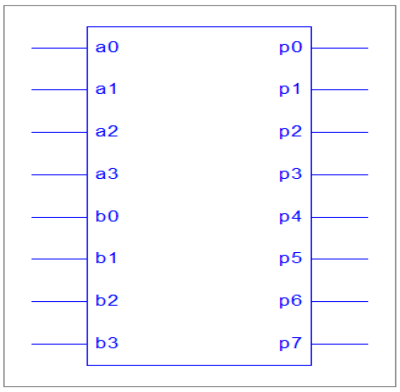


Fig.6.1. 4bit Vedic multiplier

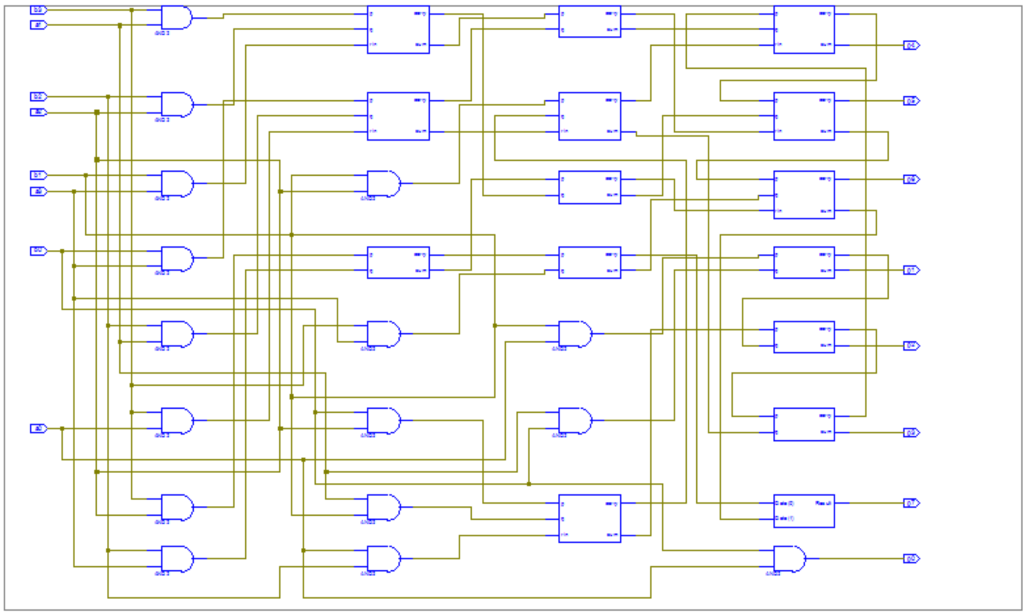
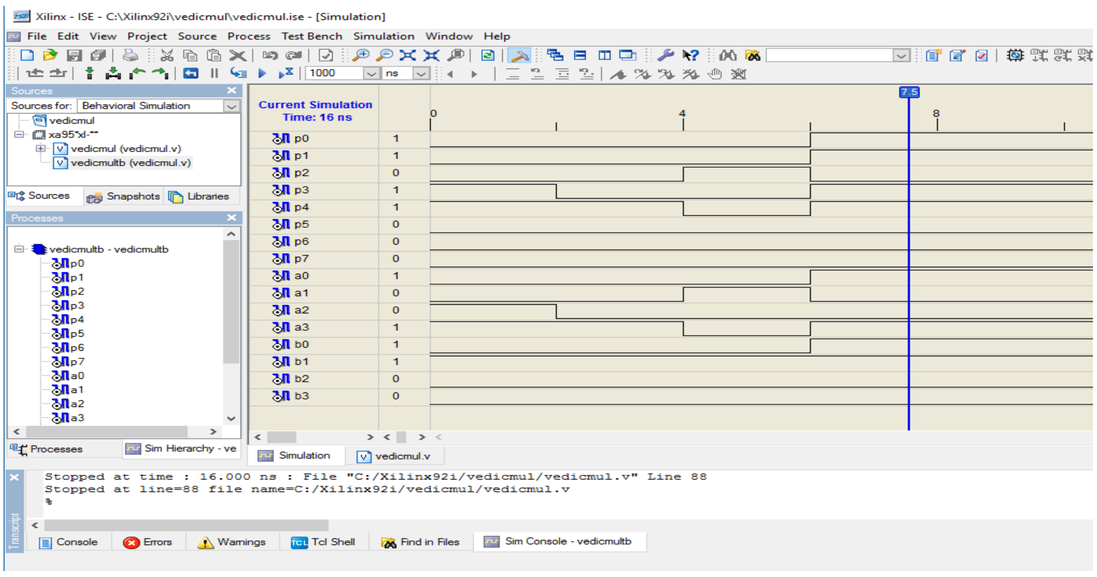


Fig.6.2.RTL View of 4bit Vedic Multiplier



```
Stopped at time : 16.000 ns : File "C:/Xilinx92i/vedicmul/vedicmul.v" Line 88
Stopped at line=88 file name=C:/Xilinx92i/vedicmul/vedicmul.v
%
```

Fig.6.3. Output waveform of 4bit Vedic multiplier.

6.1.2 Power Synthesis Result for 4bit vedic multiplier:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

Fast.lib

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:           Apr 26 2017  02:12:50 pm
  Module:                 vedicmul
  Technology library:     gpdk045wc
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
vedicmul    31    11.700  6209.750  6221.449
  f1         1     0.675   385.150   385.826
  f2         1     0.675   370.834   371.509
  f3         1     0.675   435.141   435.817
  f4         1     0.675   404.882   405.557
  f5         1     0.675   290.618   291.293
  f6         1     0.675   290.805   291.481
  f7         1     0.675   263.773   264.448
  h1         1     0.566   162.398   162.964
  h2         1     0.566   299.183   299.750
  h3         1     0.566   175.679   176.245
  h4         1     0.566   569.044   569.610
  h5         1     0.566   309.255   309.821
  h6         1     0.566   212.696   213.262
  h7         1     0.566   191.645   192.211
```

Slow.lib

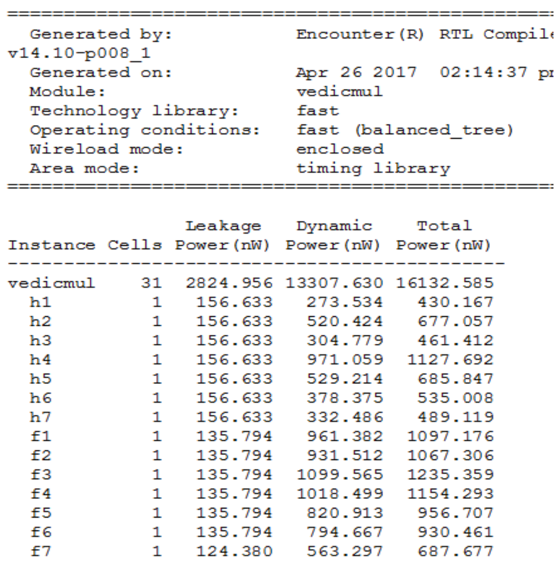
```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:           Apr 26 2017  02:18:54 pm
  Module:                 vedicmul
  Technology library:     gpdk045bc
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
vedicmul    31     3.800  4310.416  4314.216
  h1         1     0.181   107.884   108.065
  h2         1     0.181   195.243   195.424
  h3         1     0.181   115.094   115.275
  h4         1     0.181   365.864   366.045
  h5         1     0.181   199.502   199.683
  h6         1     0.181   137.084   137.265
  h7         1     0.181   125.553   125.734
  f1         1     0.168   248.108   248.276
  f2         1     0.168   236.909   237.077
  f3         1     0.168   281.651   281.819
  f4         1     0.168   260.367   260.535
  f5         1     0.168   187.858   188.025
  f6         1     0.168   186.714   186.882
  f7         1     0.168   172.713   172.881
```

Power synthesis report for 4bit Vedic multiplier using 45nm

A. Using 90nm Technology

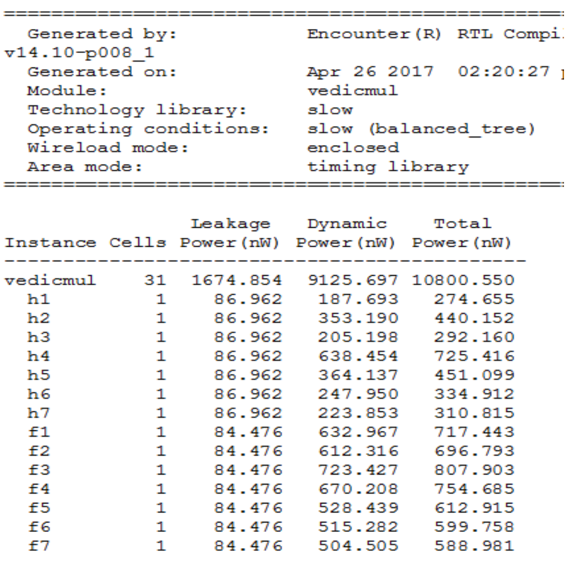
Fast.lib



```
==============================================
  Generated by:              Encounter(R) RTL Compil
v14.10-p008_1
  Generated on:              Apr 26 2017  02:14:37 pr
  Module:                    vedicmul
  Technology library:        fast
  Operating conditions:      fast (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
==============================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
vedicmul    31  2824.956 13307.630 16132.585
  h1         1   156.633   273.534   430.167
  h2         1   156.633   520.424   677.057
  h3         1   156.633   304.779   461.412
  h4         1   156.633   971.059  1127.692
  h5         1   156.633   529.214   685.847
  h6         1   156.633   378.375   535.008
  h7         1   156.633   332.486   489.119
  f1         1   135.794   961.382  1097.176
  f2         1   135.794   931.512  1067.306
  f3         1   135.794  1099.565  1235.359
  f4         1   135.794  1018.499  1154.293
  f5         1   135.794   820.913   956.707
  f6         1   135.794   794.667   930.461
  f7         1   124.380   563.297   687.677
```

Slow.lib



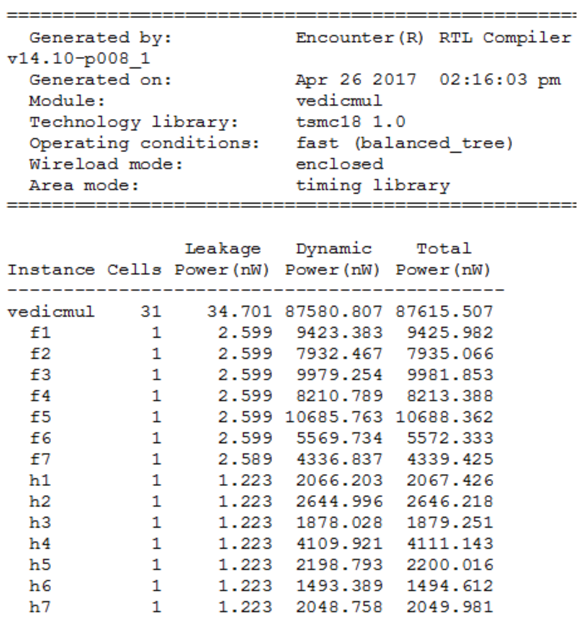
```
==============================================
  Generated by:              Encounter(R) RTL Compi
v14.10-p008_1
  Generated on:              Apr 26 2017  02:20:27
  Module:                    vedicmul
  Technology library:        slow
  Operating conditions:      slow (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
==============================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
vedicmul    31  1674.854  9125.697 10800.550
  h1         1    86.962   187.693   274.655
  h2         1    86.962   353.190   440.152
  h3         1    86.962   205.198   292.160
  h4         1    86.962   638.454   725.416
  h5         1    86.962   364.137   451.099
  h6         1    86.962   247.950   334.912
  h7         1    86.962   223.853   310.815
  f1         1    84.476   632.967   717.443
  f2         1    84.476   612.316   696.793
  f3         1    84.476   723.427   807.903
  f4         1    84.476   670.208   754.685
  f5         1    84.476   528.439   612.915
  f6         1    84.476   515.282   599.758
  f7         1    84.476   504.505   588.981
```

Power synthesis report for 4bitVedic multiplier using 90nm

A. Using 180nm Technology:

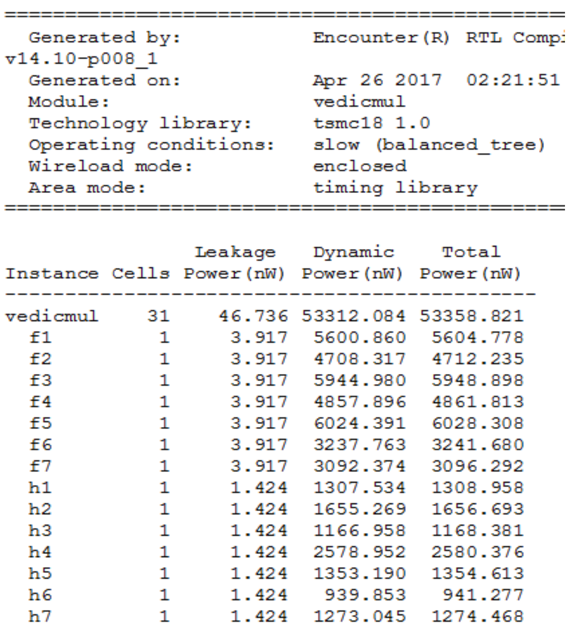
Fast.lib:



```
==============================================
  Generated by:              Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:              Apr 26 2017  02:16:03 pm
  Module:                    vedicmul
  Technology library:        tsmc18 1.0
  Operating conditions:      fast (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
==============================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
vedicmul    31    34.701 87580.807 87615.507
  f1         1     2.599  9423.383  9425.982
  f2         1     2.599  7932.467  7935.066
  f3         1     2.599  9979.254  9981.853
  f4         1     2.599  8210.789  8213.388
  f5         1     2.599 10685.763 10688.362
  f6         1     2.599  5569.734  5572.333
  f7         1     2.589  4336.837  4339.425
  h1         1     1.223  2066.203  2067.426
  h2         1     1.223  2644.996  2646.218
  h3         1     1.223  1878.028  1879.251
  h4         1     1.223  4109.921  4111.143
  h5         1     1.223  2198.793  2200.016
  h6         1     1.223  1493.389  1494.612
  h7         1     1.223  2048.758  2049.981
```

Slow.lib:



```
==============================================
  Generated by:              Encounter(R) RTL Comp
v14.10-p008_1
  Generated on:              Apr 26 2017  02:21:51
  Module:                    vedicmul
  Technology library:        tsmc18 1.0
  Operating conditions:      slow (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
==============================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
vedicmul    31    46.736 53312.084 53358.821
  f1         1     3.917  5600.860  5604.778
  f2         1     3.917  4708.317  4712.235
  f3         1     3.917  5944.980  5948.898
  f4         1     3.917  4857.896  4861.813
  f5         1     3.917  6024.391  6028.308
  f6         1     3.917  3237.763  3241.680
  f7         1     3.917  3092.374  3096.292
  h1         1     1.424  1307.534  1308.958
  h2         1     1.424  1655.269  1656.693
  h3         1     1.424  1166.958  1168.381
  h4         1     1.424  2578.952  2580.376
  h5         1     1.424  1353.190  1354.613
  h6         1     1.424   939.853   941.277
  h7         1     1.424  1273.045  1274.468
```

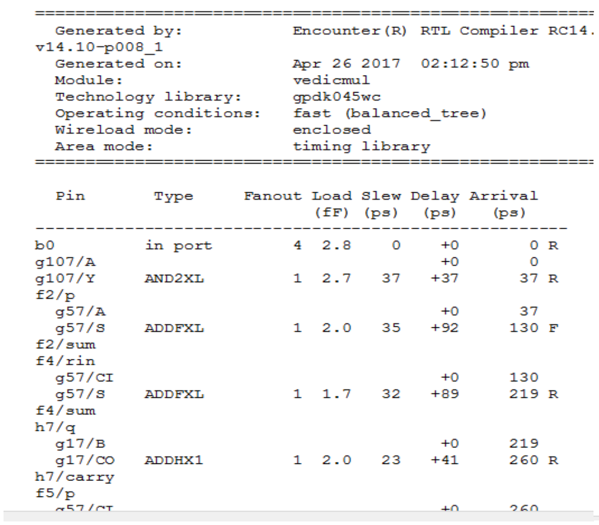
Power synthesis report for 4bit Vedic multiplier using 180nm

6.1.3 Delay and Timing Synthesis Results of 4bit vedic multiplier:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

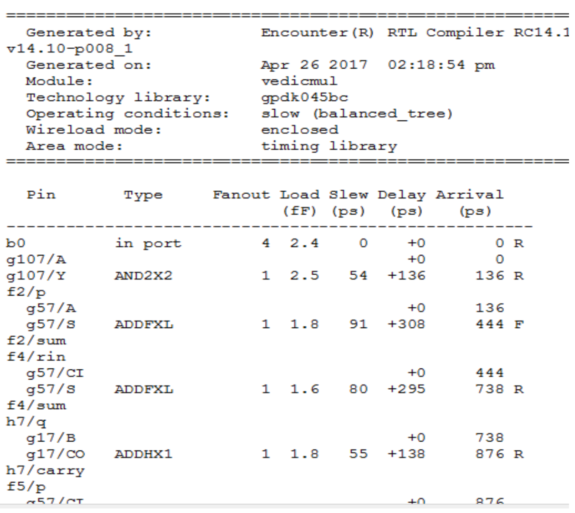
Fast.lib



```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:           Apr 26 2017  02:12:50 pm
  Module:                 vedicmul
  Technology library:     gpdk045wc
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b0           in port        4  2.8    0    +0       0 R
g107/A                                     +0       0
g107/Y       AND2XL         1  2.7   37   +37      37 R
f2/p
  g57/A                                    +0      37
  g57/S      ADDFXL         1  2.0   35   +92     130 F
f2/sum
f4/rin
  g57/CI                                   +0     130
  g57/S      ADDFXL         1  1.7   32   +89     219 R
f4/sum
h7/q
  g17/B                                    +0     219
  g17/CO     ADDHX1         1  2.0   23   +41     260 R
h7/carry
f5/p
  g57/CI                                   +0     260
```

Slow.lib



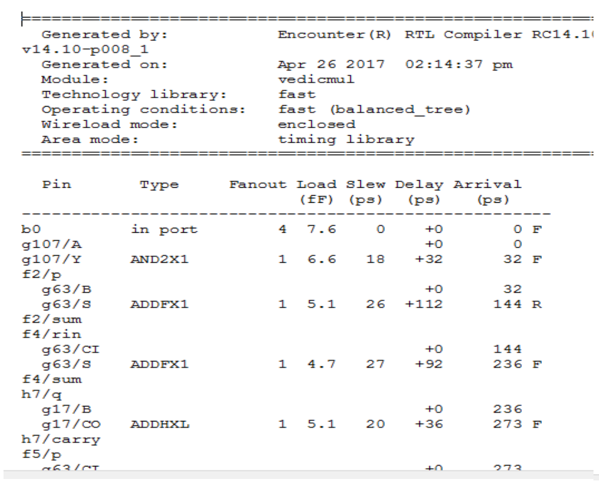
```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:           Apr 26 2017  02:18:54 pm
  Module:                 vedicmul
  Technology library:     gpdk045bc
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b0           in port        4  2.4    0    +0       0 R
g107/A                                     +0       0
g107/Y       AND2X2         1  2.5   54  +136     136 R
f2/p
  g57/A                                    +0     136
  g57/S      ADDFXL         1  1.8   91  +308     444 F
f2/sum
f4/rin
  g57/CI                                   +0     444
  g57/S      ADDFXL         1  1.6   80  +295     738 R
f4/sum
h7/q
  g17/B                                    +0     738
  g17/CO     ADDHX1         1  1.8   55  +138     876 R
h7/carry
f5/p
  g57/CI                                   +0     876
```

Timing Synthesis Report for 4bit Vedic multiplier using 45nm

A. Using 90nm Technology

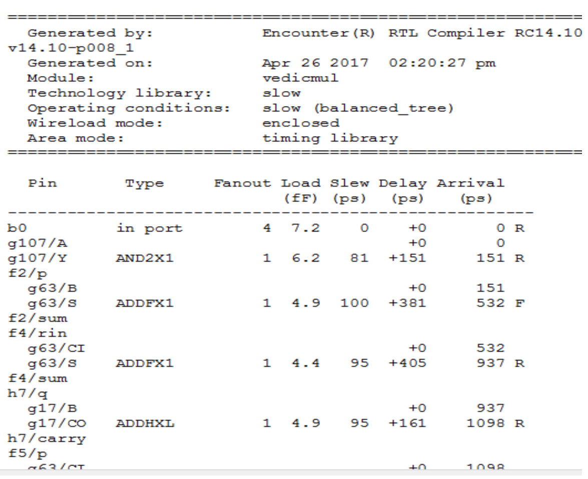
Fast.lib



```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:           Apr 26 2017  02:14:37 pm
  Module:                 vedicmul
  Technology library:     fast
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b0           in port        4  7.6    0    +0       0 F
g107/A                                     +0       0
g107/Y       AND2X1         1  6.6   18   +32      32 F
f2/p
  g63/B                                    +0      32
  g63/S      ADDFX1         1  5.1   26  +112     144 R
f2/sum
f4/rin
  g63/CI                                   +0     144
  g63/S      ADDFX1         1  4.7   27   +92     236 F
f4/sum
h7/q
  g17/B                                    +0     236
  g17/CO     ADDHXL         1  5.1   20   +36     273 F
h7/carry
f5/p
  g63/CI                                   +0     273
```

Slow.lib



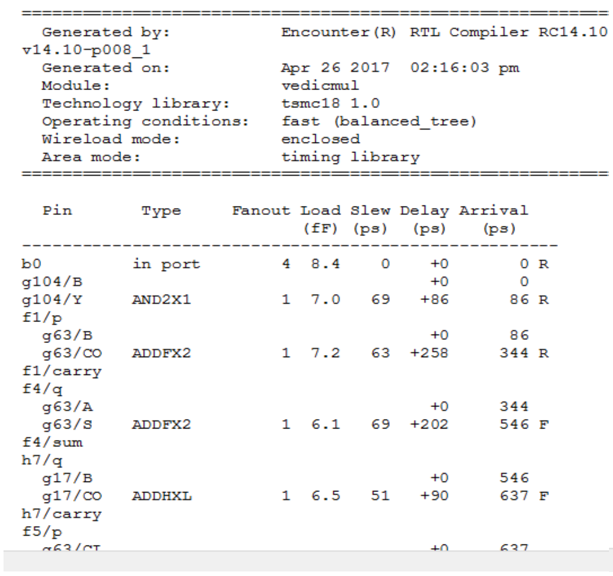
```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:           Apr 26 2017  02:20:27 pm
  Module:                 vedicmul
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b0           in port        4  7.2    0    +0       0 R
g107/A                                     +0       0
g107/Y       AND2X1         1  6.2   81  +151     151 R
f2/p
  g63/B                                    +0     151
  g63/S      ADDFX1         1  4.9  100  +381     532 F
f2/sum
f4/rin
  g63/CI                                   +0     532
  g63/S      ADDFX1         1  4.4   95  +405     937 R
f4/sum
h7/q
  g17/B                                    +0     937
  g17/CO     ADDHXL         1  4.9   95  +161    1098 R
h7/carry
f5/p
  g63/CI                                   +0    1098
```

Fig.6.8. Timing Synthesis Report for 4bit Vedic multiplier using 90nm

A. Using 180nm Technology:

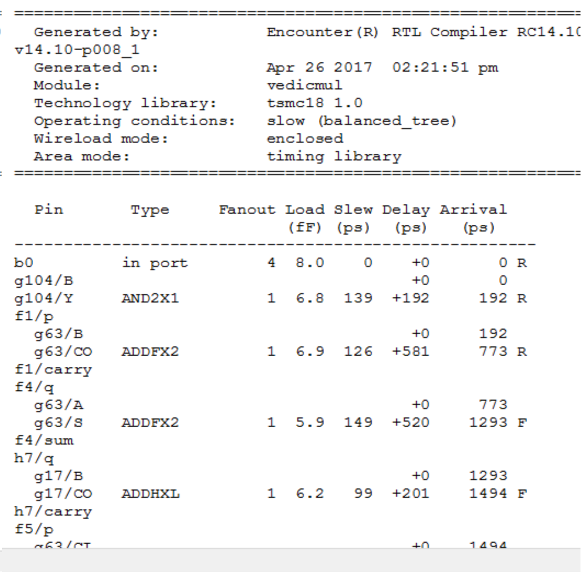
Fast.lib



```
===========================================================
  Generated by:           Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:           Apr 26 2017  02:16:03 pm
  Module:                 vedicmul
  Technology library:     tsmc18 1.0
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
===========================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
------------------------------------------------------
b0           in port        4  8.4     0    +0       0 R
g104/B                                      +0       0
g104/Y       AND2X1         1  7.0    69   +86      86 R
f1/p
  g63/B                                     +0      86
  g63/CO     ADDFX2         1  7.2    63  +258     344 R
f1/carry
f4/q
  g63/A                                     +0     344
  g63/S      ADDFX2         1  6.1    69  +202     546 F
f4/sum
h7/q
  g17/B                                     +0     546
  g17/CO     ADDHXL         1  6.5    51   +90     637 F
h7/carry
f5/p
  g63/CI                                    +0     637
```

Slow.lib



```
===========================================================
  Generated by:           Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:           Apr 26 2017  02:21:51 pm
  Module:                 vedicmul
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
===========================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
------------------------------------------------------
b0           in port        4  8.0     0    +0       0 R
g104/B                                      +0       0
g104/Y       AND2X1         1  6.8   139  +192     192 R
f1/p
  g63/B                                     +0     192
  g63/CO     ADDFX2         1  6.9   126  +581     773 R
f1/carry
f4/q
  g63/A                                     +0     773
  g63/S      ADDFX2         1  5.9   149  +520    1293 F
f4/sum
h7/q
  g17/B                                     +0    1293
  g17/CO     ADDHXL         1  6.2    99  +201    1494 F
h7/carry
f5/p
  g63/CI                                    +0    1494
```

Timing Synthesis Report for 4bit Vedic multiplier using 180nm

6.1.4 Vedic 8bit Multiplier:-

Vedic multiplier has been implemented in Xilinx and NC simulation using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

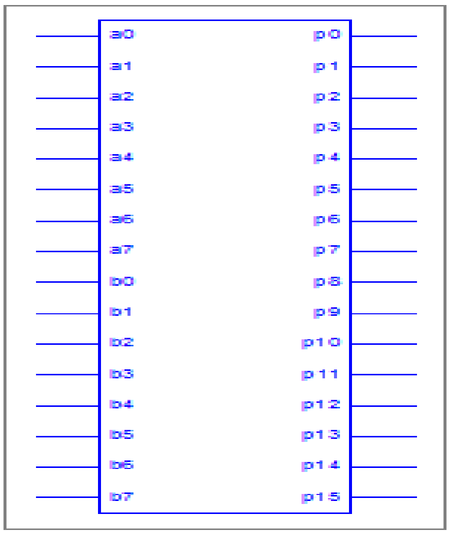


Fig.6.4. 8bit Vedic multiplier

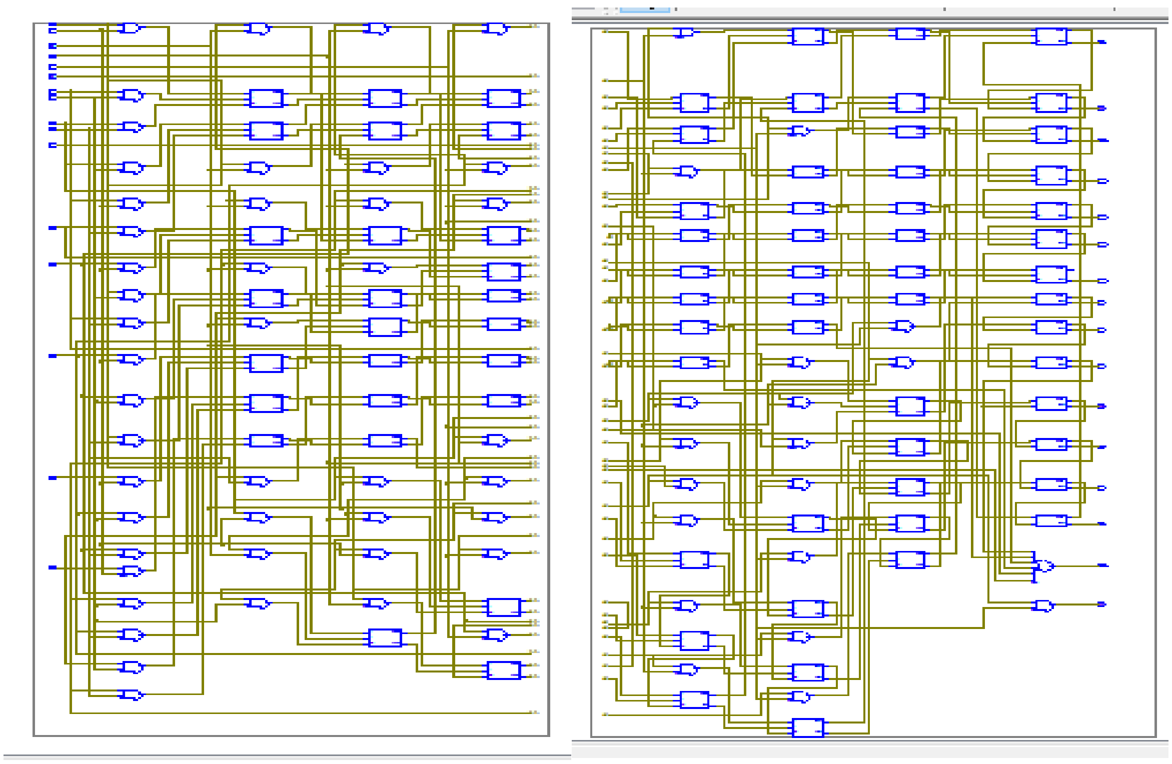
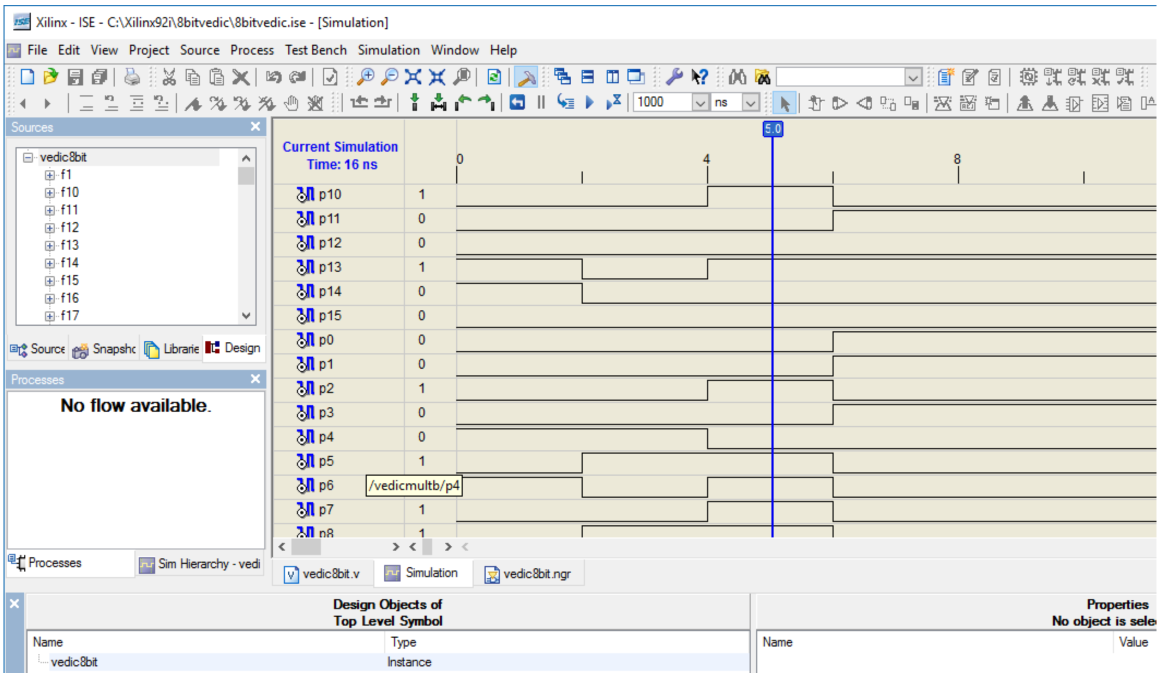


Fig.6.5. RTL view of 8bit vedic multiplier


Xilinx - ISE - C:\Xilinx92i\8bitvedic\8bitvedic.ise - [Simulation]
File Edit View Project Source Process Test Bench Simulation Window Help
Sources
Processes
No flow available.
Current Simulation Time: 16 ns
/vedicmultb/p4
vedic8bit.v
Simulation
vedic8bit.ngr
Design Objects of Top Level Symbol
Properties
No object is sele

```
Sources
vedic8bit
  f1
  f10
  f11
  f12
  f13
  f14
  f15
  f16
  f17

Current Simulation
Time: 16 ns
p10   1
p11   0
p12   0
p13   1
p14   0
p15   0
p0    0
p1    0
p2    1
p3    0
p4    0
p5    1
p6
p7    1
p8    1

Name        Type
vedic8bit   Instance
```

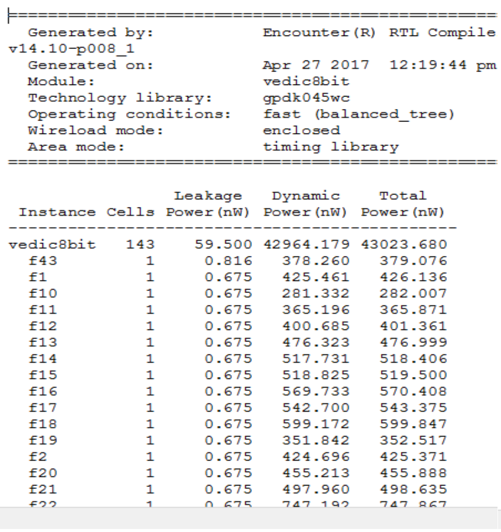
Fig.6.6. Output waveforms of the 8bit vedic multiplier

6.1.5 Power Synthesis Result of 8bit vedic multiplier:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

B. Using 45nm Technology

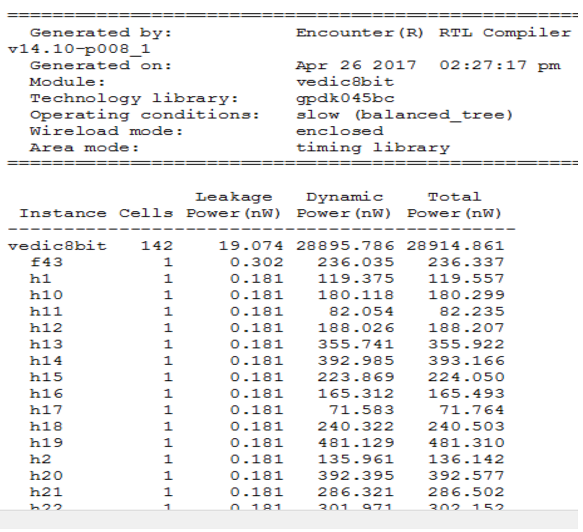
Fast.lib



```
=============================================================
  Generated by:             Encounter(R) RTL Compile
v14.10-p008_1
  Generated on:             Apr 27 2017  12:19:44 pm
  Module:                   vedic8bit
  Technology library:       gpdk045wc
  Operating conditions:     fast (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=============================================================

                     Leakage   Dynamic    Total
 Instance Cells Power(nW) Power(nW) Power(nW)
------------------------------------------------
vedic8bit   143    59.500 42964.179 43023.680
  f43         1     0.816   378.260   379.076
  f1          1     0.675   425.461   426.136
  f10         1     0.675   281.332   282.007
  f11         1     0.675   365.196   365.871
  f12         1     0.675   400.685   401.361
  f13         1     0.675   476.323   476.999
  f14         1     0.675   517.731   518.406
  f15         1     0.675   518.825   519.500
  f16         1     0.675   569.733   570.408
  f17         1     0.675   542.700   543.375
  f18         1     0.675   599.172   599.847
  f19         1     0.675   351.842   352.517
  f2          1     0.675   424.696   425.371
  f20         1     0.675   455.213   455.888
  f21         1     0.675   497.960   498.635
  f22         1     0.675   747.192   747.867
```

Slow.lib



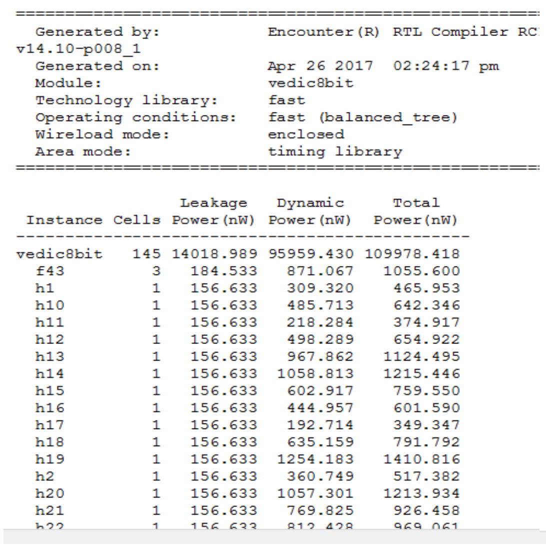
```
=============================================================
  Generated by:             Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:             Apr 26 2017   02:27:17 pm
  Module:                   vedic8bit
  Technology library:       gpdk045bc
  Operating conditions:     slow (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=============================================================

                     Leakage   Dynamic    Total
 Instance Cells Power(nW) Power(nW) Power(nW)
------------------------------------------------
vedic8bit   142    19.074 28895.786 28914.861
  f43         1     0.302   236.035   236.337
  h1          1     0.181   119.375   119.557
  h10         1     0.181   180.118   180.299
  h11         1     0.181    82.054    82.235
  h12         1     0.181   188.026   188.207
  h13         1     0.181   355.741   355.922
  h14         1     0.181   392.985   393.166
  h15         1     0.181   223.869   224.050
  h16         1     0.181   165.312   165.493
  h17         1     0.181    71.583    71.764
  h18         1     0.181   240.322   240.503
  h19         1     0.181   481.129   481.310
  h2          1     0.181   135.961   136.142
  h20         1     0.181   392.395   392.577
  h21         1     0.181   286.321   286.502
  h22         1     0.181   301.971   302.152
```

Power synthesis report for 8bit Vedic multiplier using 45nm

B. Using 90nm Technology

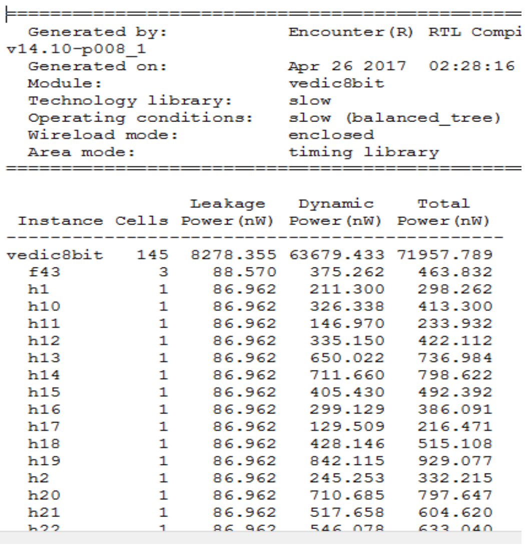
Fast.lib



```
=====================================================
  Generated by:            Encounter(R) RTL Compiler RC
v14.10-p008_1
  Generated on:            Apr 26 2017  02:24:17 pm
  Module:                  vedic8bit
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=====================================================

                     Leakage   Dynamic     Total
 Instance  Cells  Power(nW)  Power(nW)   Power(nW)
----------------------------------------------------
vedic8bit    145  14018.989  95959.430  109978.418
  f43          3    184.533    871.067   1055.600
  h1           1    156.633    309.320    465.953
  h10          1    156.633    485.713    642.346
  h11          1    156.633    218.284    374.917
  h12          1    156.633    498.289    654.922
  h13          1    156.633    967.862   1124.495
  h14          1    156.633   1058.813   1215.446
  h15          1    156.633    602.917    759.550
  h16          1    156.633    444.957    601.590
  h17          1    156.633    192.714    349.347
  h18          1    156.633    635.159    791.792
  h19          1    156.633   1254.183   1410.816
  h2           1    156.633    360.749    517.382
  h20          1    156.633   1057.301   1213.934
  h21          1    156.633    769.825    926.458
  h22          1    156.633    812.428    969.061
```

Slow.lib



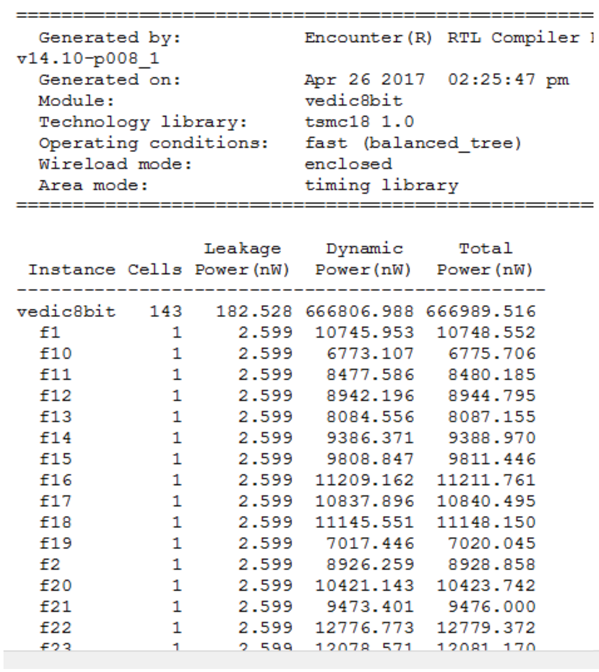
```
=====================================================
  Generated by:            Encounter(R) RTL Compi
v14.10-p008_1
  Generated on:            Apr 26 2017  02:28:16
  Module:                  vedic8bit
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=====================================================

                     Leakage   Dynamic     Total
 Instance  Cells  Power(nW)  Power(nW)   Power(nW)
----------------------------------------------------
vedic8bit    145   8278.355  63679.433  71957.789
  f43          3     88.570    375.262    463.832
  h1           1     86.962    211.300    298.262
  h10          1     86.962    326.338    413.300
  h11          1     86.962    146.970    233.932
  h12          1     86.962    335.150    422.112
  h13          1     86.962    650.022    736.984
  h14          1     86.962    711.660    798.622
  h15          1     86.962    405.430    492.392
  h16          1     86.962    299.129    386.091
  h17          1     86.962    129.509    216.471
  h18          1     86.962    428.146    515.108
  h19          1     86.962    842.115    929.077
  h2           1     86.962    245.253    332.215
  h20          1     86.962    710.685    797.647
  h21          1     86.962    517.658    604.620
  h22          1     86.962    546.078    633.040
```

Power synthesis report for 8bit Vedic multiplier using 90nm

B. Using 180nm Technology:

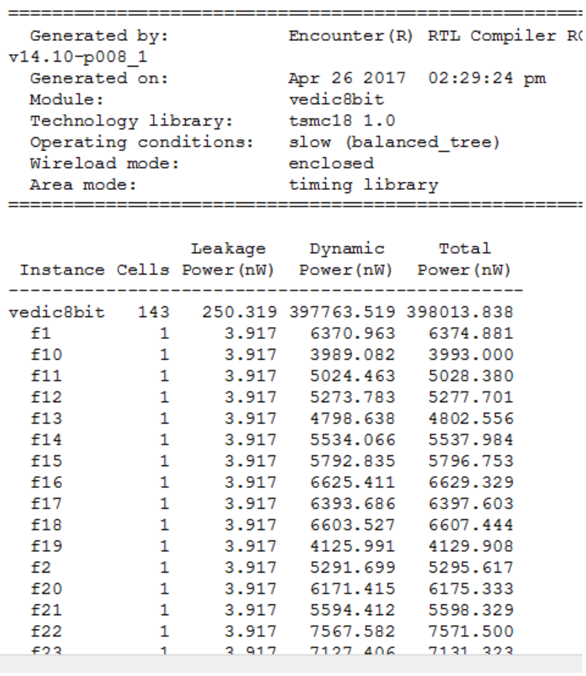
Fast.lib:



```
=====================================================
  Generated by:            Encounter(R) RTL Compiler I
v14.10-p008_1
  Generated on:            Apr 26 2017  02:25:47 pm
  Module:                  vedic8bit
  Technology library:      tsmc18 1.0
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=====================================================

                     Leakage    Dynamic      Total
 Instance  Cells  Power(nW)   Power(nW)   Power(nW)
-----------------------------------------------------
vedic8bit    143    182.528  666806.988  666989.516
  f1           1      2.599   10745.953   10748.552
  f10          1      2.599    6773.107    6775.706
  f11          1      2.599    8477.586    8480.185
  f12          1      2.599    8942.196    8944.795
  f13          1      2.599    8084.556    8087.155
  f14          1      2.599    9386.371    9388.970
  f15          1      2.599    9808.847    9811.446
  f16          1      2.599   11209.162   11211.761
  f17          1      2.599   10837.896   10840.495
  f18          1      2.599   11145.551   11148.150
  f19          1      2.599    7017.446    7020.045
  f2           1      2.599    8926.259    8928.858
  f20          1      2.599   10421.143   10423.742
  f21          1      2.599    9473.401    9476.000
  f22          1      2.599   12776.773   12779.372
  f23          1      2.599   12078.571   12081.170
```

Slow.lib:



```
=====================================================
  Generated by:            Encounter(R) RTL Compiler R(
v14.10-p008_1
  Generated on:            Apr 26 2017  02:29:24 pm
  Module:                  vedic8bit
  Technology library:      tsmc18 1.0
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=====================================================

                     Leakage    Dynamic      Total
 Instance  Cells  Power(nW)   Power(nW)   Power(nW)
-----------------------------------------------------
vedic8bit    143    250.319  397763.519  398013.838
  f1           1      3.917    6370.963    6374.881
  f10          1      3.917    3989.082    3993.000
  f11          1      3.917    5024.463    5028.380
  f12          1      3.917    5273.783    5277.701
  f13          1      3.917    4798.638    4802.556
  f14          1      3.917    5534.066    5537.984
  f15          1      3.917    5792.835    5796.753
  f16          1      3.917    6625.411    6629.329
  f17          1      3.917    6393.686    6397.603
  f18          1      3.917    6603.527    6607.444
  f19          1      3.917    4125.991    4129.908
  f2           1      3.917    5291.699    5295.617
  f20          1      3.917    6171.415    6175.333
  f21          1      3.917    5594.412    5598.329
  f22          1      3.917    7567.582    7571.500
  f23          1      3.917    7127.406    7131.323
```

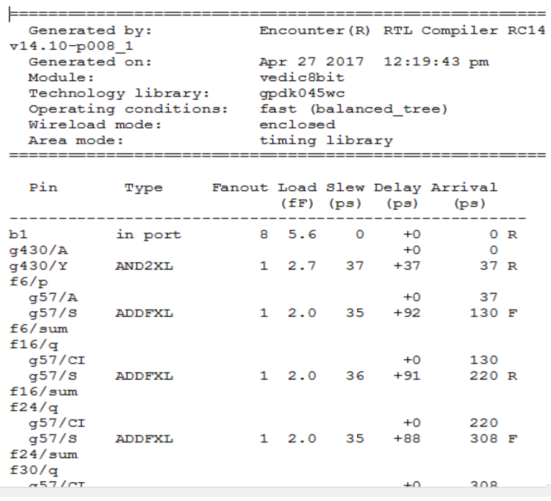
Power synthesis report for 8bit Vedic multiplier using 180nm

6.1.6 Delay and Timing Synthesis Results:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

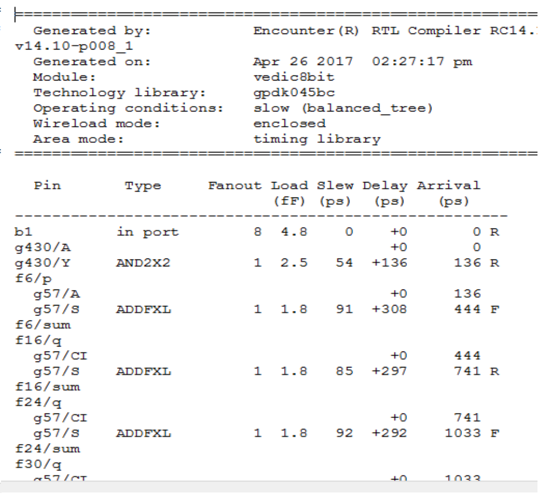
B. Using 45nm Technology

Fast.lib Slow.lib



```
|=====================================================
  Generated by:             Encounter(R) RTL Compiler RC14
v14.10-p008_1
  Generated on:             Apr 27 2017  12:19:43 pm
  Module:                   vedic8bit
  Technology library:       gpdk045wc
  Operating conditions:     fast (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=====================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)    (ps)
---------------------------------------------------
b1          in port         8  5.6    0    +0       0 R
g430/A                                     +0       0
g430/Y      AND2XL          1  2.7   37   +37      37 R
f6/p
  g57/A                                    +0      37
  g57/S     ADDFXL          1  2.0   35   +92     130 F
f6/sum
f16/q
  g57/CI                                   +0     130
  g57/S     ADDFXL          1  2.0   36   +91     220 R
f16/sum
f24/q
  g57/CI                                   +0     220
  g57/S     ADDFXL          1  2.0   35   +88     308 F
f24/sum
f30/q
  g57/CI                                   +0     308
```



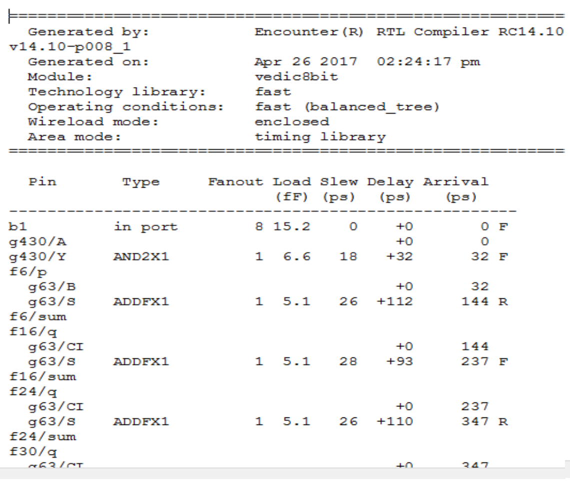
```
|=====================================================
  Generated by:             Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:             Apr 26 2017  02:27:17 pm
  Module:                   vedic8bit
  Technology library:       gpdk045bc
  Operating conditions:     slow (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=====================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)    (ps)
---------------------------------------------------
b1          in port         8  4.8    0    +0       0 R
g430/A                                     +0       0
g430/Y      AND2X2          1  2.5   54  +136     136 R
f6/p
  g57/A                                    +0     136
  g57/S     ADDFXL          1  1.8   91  +308     444 F
f6/sum
f16/q
  g57/CI                                   +0     444
  g57/S     ADDFXL          1  1.8   85  +297     741 R
f16/sum
f24/q
  g57/CI                                   +0     741
  g57/S     ADDFXL          1  1.8   92  +292    1033 F
f24/sum
f30/q
  g57/CI                                   +0    1033
```

Timing Synthesis Report for 8bit Vedic multiplier using 45nm

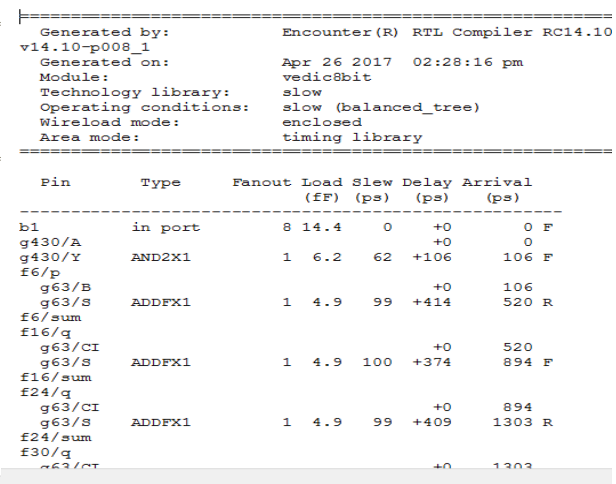
B. Using 90nm Technology

Fast.lib Slow.lib



```
|=====================================================
  Generated by:             Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:             Apr 26 2017  02:24:17 pm
  Module:                   vedic8bit
  Technology library:       fast
  Operating conditions:     fast (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=====================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)    (ps)
---------------------------------------------------
b1          in port         8 15.2    0    +0       0 F
g430/A                                     +0       0
g430/Y      AND2X1          1  6.6   18   +32      32 F
f6/p
  g63/B                                    +0      32
  g63/S     ADDFX1          1  5.1   26  +112     144 R
f6/sum
f16/q
  g63/CI                                   +0     144
  g63/S     ADDFX1          1  5.1   28   +93     237 F
f16/sum
f24/q
  g63/CI                                   +0     237
  g63/S     ADDFX1          1  5.1   26  +110     347 R
f24/sum
f30/q
  g63/CI                                   +0     347
```



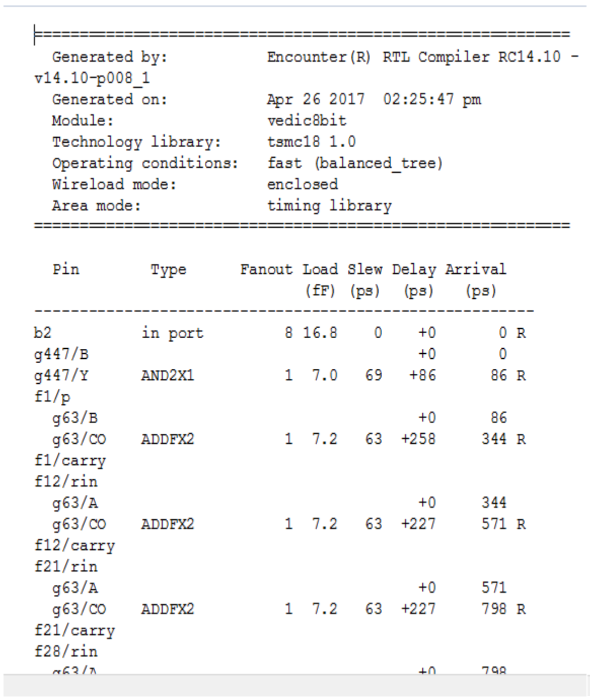
```
|=====================================================
  Generated by:             Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:             Apr 26 2017  02:28:16 pm
  Module:                   vedic8bit
  Technology library:       slow
  Operating conditions:     slow (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
=====================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)    (ps)
---------------------------------------------------
b1          in port         8 14.4    0    +0       0 F
g430/A                                     +0       0
g430/Y      AND2X1          1  6.2   62  +106     106 F
f6/p
  g63/B                                    +0     106
  g63/S     ADDFX1          1  4.9   99  +414     520 R
f6/sum
f16/q
  g63/CI                                   +0     520
  g63/S     ADDFX1          1  4.9  100  +374     894 F
f16/sum
f24/q
  g63/CI                                   +0     894
  g63/S     ADDFX1          1  4.9   99  +409    1303 R
f24/sum
f30/q
  g63/CI                                   +0    1303
```

Timing Synthesis Report for 8bit Vedic multiplier using 90nm

B. Using 180nm Technology:

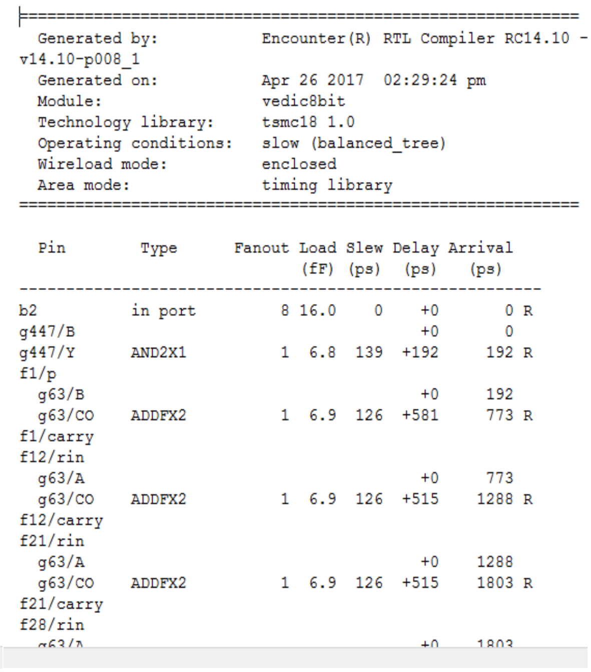
Fast.lib



```
==========================================================
  Generated by:           Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:           Apr 26 2017  02:25:47 pm
  Module:                 vedic8bit
  Technology library:     tsmc18 1.0
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==========================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF)  (ps)  (ps)   (ps)
------------------------------------------------------
b2          in port       8 16.8    0    +0       0 R
g447/B                                   +0       0
g447/Y      AND2X1        1  7.0   69   +86      86 R
f1/p
  g63/B                                  +0      86
  g63/CO    ADDFX2        1  7.2   63  +258     344 R
f1/carry
f12/rin
  g63/A                                  +0     344
  g63/CO    ADDFX2        1  7.2   63  +227     571 R
f12/carry
f21/rin
  g63/A                                  +0     571
  g63/CO    ADDFX2        1  7.2   63  +227     798 R
f21/carry
f28/rin
  g63/A                                  +0     798
```

Slow.lib



```
==========================================================
  Generated by:           Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:           Apr 26 2017  02:29:24 pm
  Module:                 vedic8bit
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==========================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF)  (ps)  (ps)   (ps)
------------------------------------------------------
b2          in port       8 16.0    0    +0       0 R
g447/B                                   +0       0
g447/Y      AND2X1        1  6.8  139  +192     192 R
f1/p
  g63/B                                  +0     192
  g63/CO    ADDFX2        1  6.9  126  +581     773 R
f1/carry
f12/rin
  g63/A                                  +0     773
  g63/CO    ADDFX2        1  6.9  126  +515    1288 R
f12/carry
f21/rin
  g63/A                                  +0    1288
  g63/CO    ADDFX2        1  6.9  126  +515    1803 R
f21/carry
f28/rin
  g63/A                                  +0    1803
```

Timing Synthesis Report for 8bit Vedic multiplier using 180nm

6.2 WALLACE TREE MULTIPLIER

4bit Wallace tree

The multiplier has been designed in Xilinx and NCsim using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

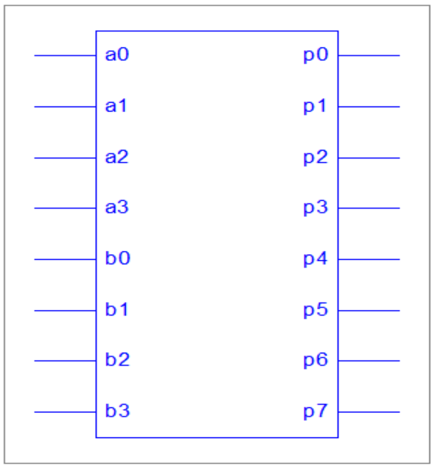


Fig6.7. Wallace tree 4bit multiplier

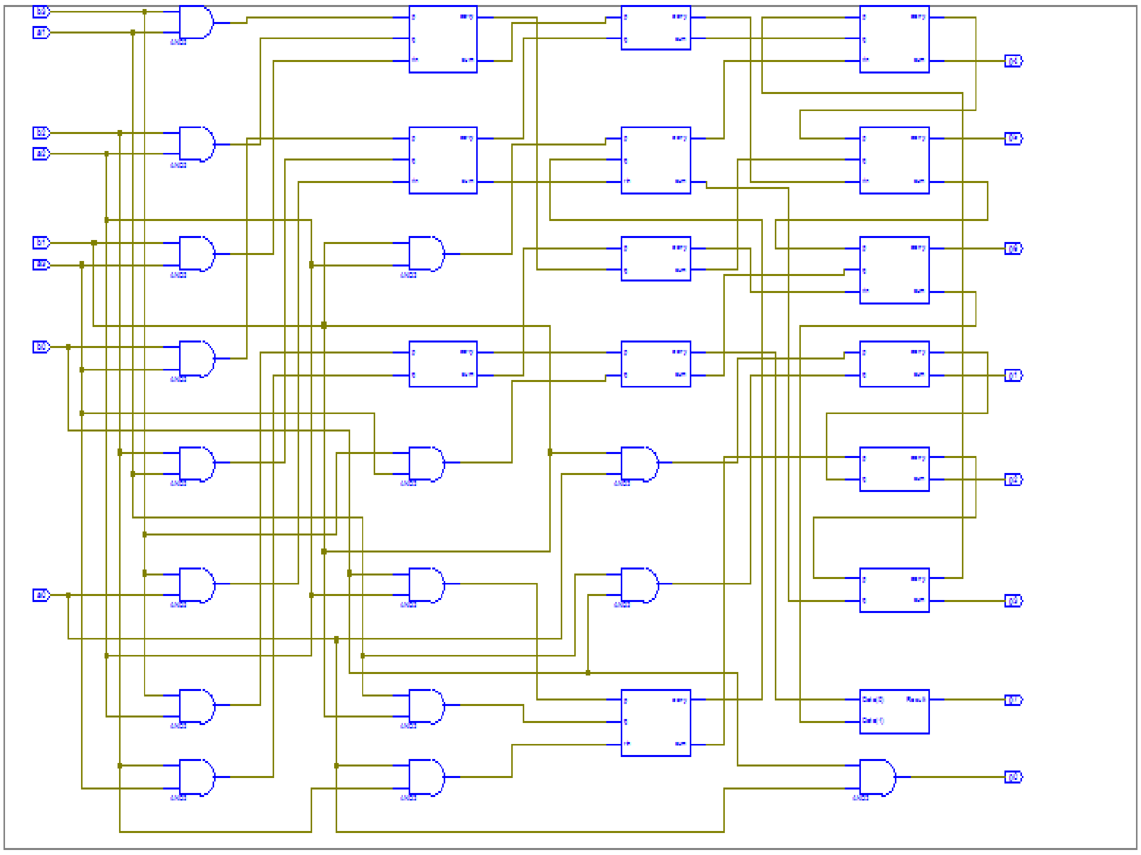
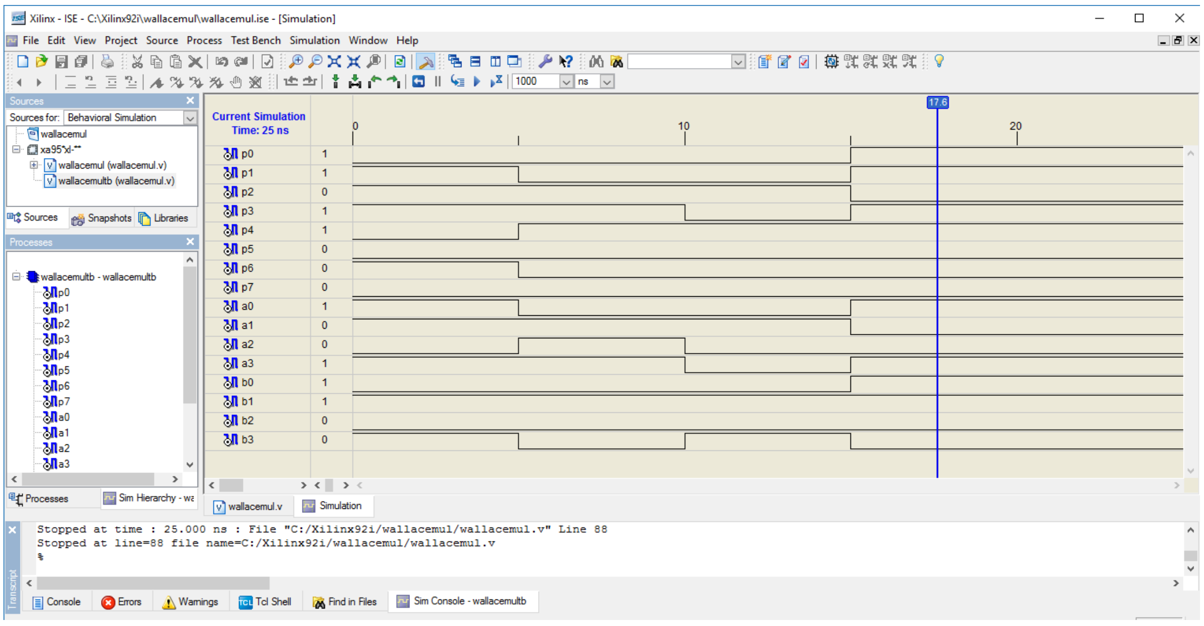


Fig6.8. RTL View of Wallace tree 4bit multiplier



```
Stopped at time : 25.000 ns : File "C:/Xilinx92i/wallacemul/wallacemul.v" Line 88
Stopped at line=88 file name=C:/Xilinx92i/wallacemul/wallacemul.v
%
```

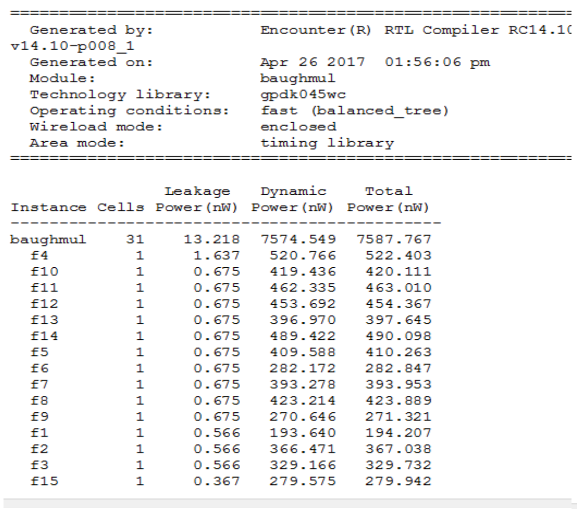
Fig.6.9. Output waveform of Wallace tree 4bit multiplier.

6.2.1 Power Synthesis Result:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

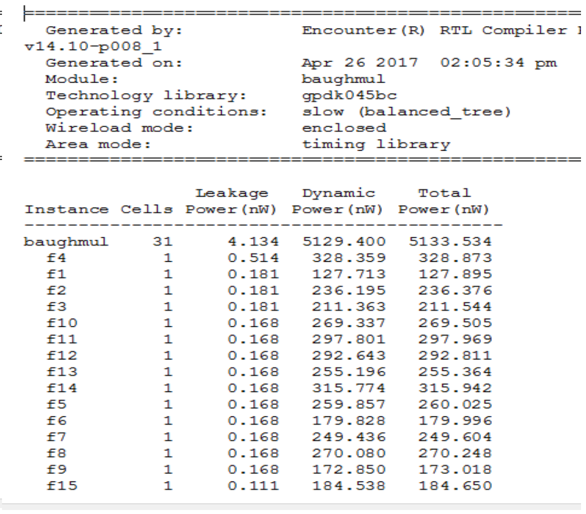
Fast.lib:



```
========================================================================
  Generated by:              Encounter(R) RTL Compiler RC14.1(
v14.10-p008_1
  Generated on:              Apr 26 2017  01:56:06 pm
  Module:                    baughmul
  Technology library:        gpdk045wc
  Operating conditions:      fast (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
----------------------------------------
baughmul    31    13.218  7574.549  7587.767
  f4         1     1.637   520.766   522.403
  f10        1     0.675   419.436   420.111
  f11        1     0.675   462.335   463.010
  f12        1     0.675   453.692   454.367
  f13        1     0.675   396.970   397.645
  f14        1     0.675   489.422   490.098
  f5         1     0.675   409.588   410.263
  f6         1     0.675   282.172   282.847
  f7         1     0.675   393.278   393.953
  f8         1     0.675   423.214   423.889
  f9         1     0.675   270.646   271.321
  f1         1     0.566   193.640   194.207
  f2         1     0.566   366.471   367.038
  f3         1     0.566   329.166   329.732
  f15        1     0.367   279.575   279.942
```

Slow.lib:



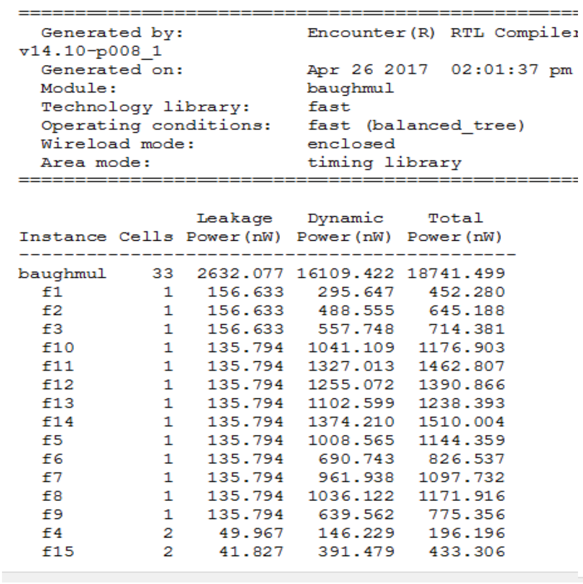
```
========================================================================
  Generated by:              Encounter(R) RTL Compiler I
v14.10-p008_1
  Generated on:              Apr 26 2017  02:05:34 pm
  Module:                    baughmul
  Technology library:        gpdk045bc
  Operating conditions:      slow (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
----------------------------------------
baughmul    31     4.134  5129.400  5133.534
  f4         1     0.514   328.359   328.873
  f1         1     0.181   127.713   127.895
  f2         1     0.181   236.195   236.376
  f3         1     0.181   211.363   211.544
  f10        1     0.168   269.337   269.505
  f11        1     0.168   297.801   297.969
  f12        1     0.168   292.643   292.811
  f13        1     0.168   255.196   255.364
  f14        1     0.168   315.774   315.942
  f5         1     0.168   259.857   260.025
  f6         1     0.168   179.828   179.996
  f7         1     0.168   249.436   249.604
  f8         1     0.168   270.080   270.248
  f9         1     0.168   172.850   173.018
  f15        1     0.111   184.538   184.650
```

Power synthesis report for 4bit Wallace tree multiplier using 45nm

A. Using 90nm Technology

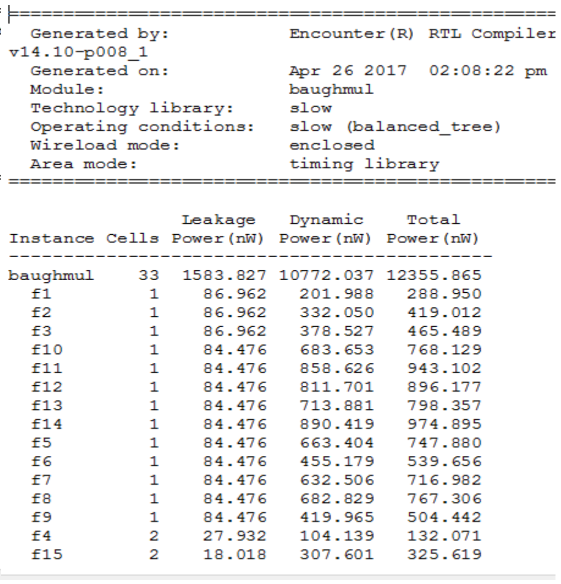
Fast.lib



```
==================================================
  Generated by:           Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:           Apr 26 2017  02:01:37 pm
  Module:                 baughmul
  Technology library:     fast
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
baughmul   33  2632.077 16109.422 18741.499
  f1        1   156.633   295.647   452.280
  f2        1   156.633   488.555   645.188
  f3        1   156.633   557.748   714.381
  f10       1   135.794  1041.109  1176.903
  f11       1   135.794  1327.013  1462.807
  f12       1   135.794  1255.072  1390.866
  f13       1   135.794  1102.599  1238.393
  f14       1   135.794  1374.210  1510.004
  f5        1   135.794  1008.565  1144.359
  f6        1   135.794   690.743   826.537
  f7        1   135.794   961.938  1097.732
  f8        1   135.794  1036.122  1171.916
  f9        1   135.794   639.562   775.356
  f4        2    49.967   146.229   196.196
  f15       2    41.827   391.479   433.306
```

Slow.lib



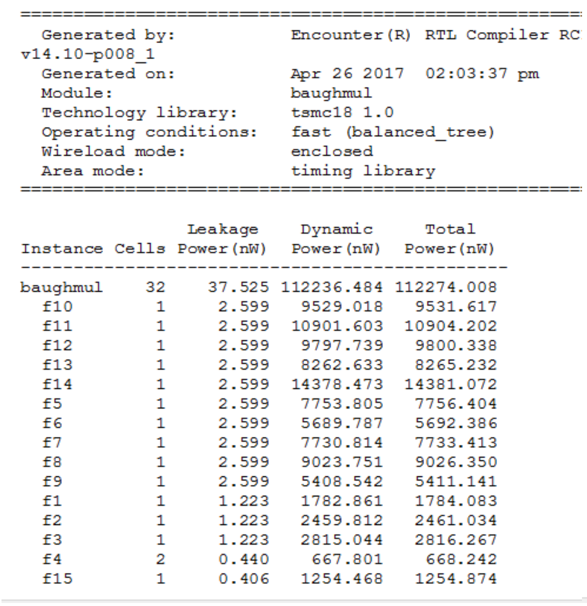
```
==================================================
  Generated by:           Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:           Apr 26 2017  02:08:22 pm
  Module:                 baughmul
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
baughmul   33  1583.827 10772.037 12355.865
  f1        1    86.962   201.988   288.950
  f2        1    86.962   332.050   419.012
  f3        1    86.962   378.527   465.489
  f10       1    84.476   683.653   768.129
  f11       1    84.476   858.626   943.102
  f12       1    84.476   811.701   896.177
  f13       1    84.476   713.881   798.357
  f14       1    84.476   890.419   974.895
  f5        1    84.476   663.404   747.880
  f6        1    84.476   455.179   539.656
  f7        1    84.476   632.506   716.982
  f8        1    84.476   682.829   767.306
  f9        1    84.476   419.965   504.442
  f4        2    27.932   104.139   132.071
  f15       2    18.018   307.601   325.619
```

Power synthesis report for 4bit Wallace tree multiplier using 90nm

A. Using 180nm Technology:

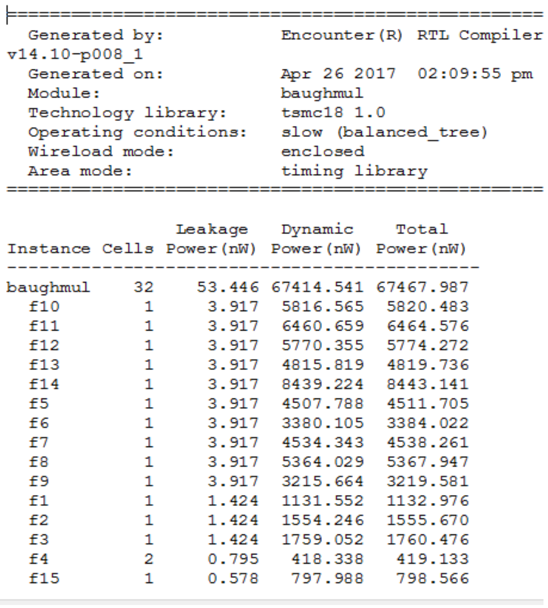
Fast.lib:



```
==================================================
  Generated by:           Encounter(R) RTL Compiler RC
v14.10-p008_1
  Generated on:           Apr 26 2017  02:03:37 pm
  Module:                 baughmul
  Technology library:     tsmc18 1.0
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==================================================

                 Leakage    Dynamic     Total
Instance Cells Power(nW)  Power(nW)   Power(nW)
-----------------------------------------------
baughmul   32    37.525 112236.484 112274.008
  f10       1     2.599   9529.018   9531.617
  f11       1     2.599  10901.603  10904.202
  f12       1     2.599   9797.739   9800.338
  f13       1     2.599   8262.633   8265.232
  f14       1     2.599  14378.473  14381.072
  f5        1     2.599   7753.805   7756.404
  f6        1     2.599   5689.787   5692.386
  f7        1     2.599   7730.814   7733.413
  f8        1     2.599   9023.751   9026.350
  f9        1     2.599   5408.542   5411.141
  f1        1     1.223   1782.861   1784.083
  f2        1     1.223   2459.812   2461.034
  f3        1     1.223   2815.044   2816.267
  f4        2     0.440    667.801    668.242
  f15       1     0.406   1254.468   1254.874
```

Slow.lib:



```
==================================================
  Generated by:           Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:           Apr 26 2017  02:09:55 pm
  Module:                 baughmul
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------
baughmul   32    53.446 67414.541 67467.987
  f10       1     3.917  5816.565  5820.483
  f11       1     3.917  6460.659  6464.576
  f12       1     3.917  5770.355  5774.272
  f13       1     3.917  4815.819  4819.736
  f14       1     3.917  8439.224  8443.141
  f5        1     3.917  4507.788  4511.705
  f6        1     3.917  3380.105  3384.022
  f7        1     3.917  4534.343  4538.261
  f8        1     3.917  5364.029  5367.947
  f9        1     3.917  3215.664  3219.581
  f1        1     1.424  1131.552  1132.976
  f2        1     1.424  1554.246  1555.670
  f3        1     1.424  1759.052  1760.476
  f4        2     0.795   418.338   419.133
  f15       1     0.578   797.988   798.566
```

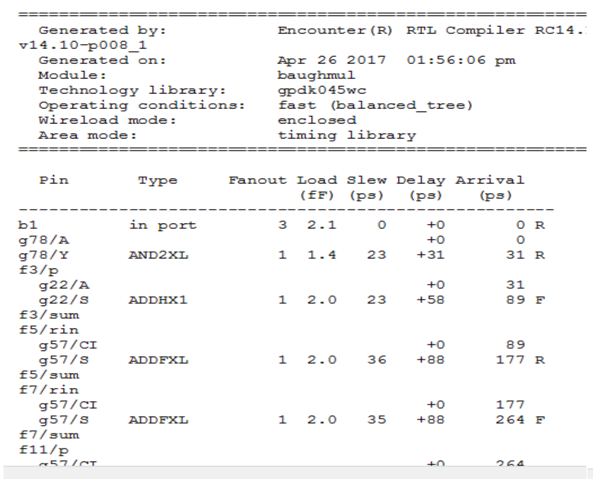
Power synthesis report for 4bit Wallace tree multiplier using 180nm

6.2.2 Delay and Timing Synthesis Results:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

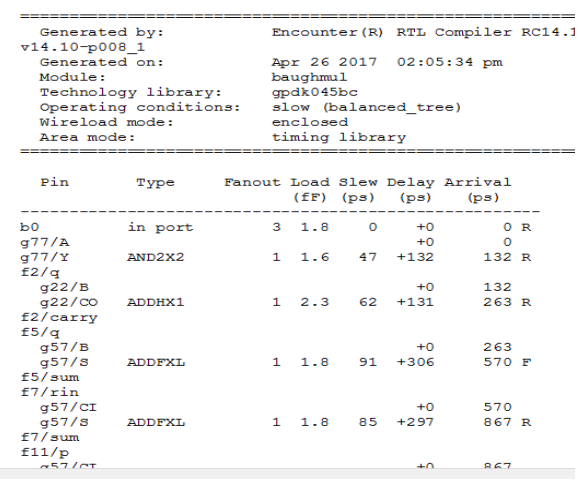
Fast.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:            Apr 26 2017  01:56:06 pm
  Module:                  baughmul
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
--------------------------------------------------------
b1          in port          3  2.1    0    +0       0 R
g78/A                                       +0       0
g78/Y       AND2XL           1  1.4   23   +31      31 R
f3/p
  g22/A                                     +0      31
  g22/S     ADDHX1           1  2.0   23   +58      89 F
f3/sum
f5/rin
  g57/CI                                    +0      89
  g57/S     ADDFXL           1  2.0   36   +88     177 R
f5/sum
f7/rin
  g57/CI                                    +0     177
  g57/S     ADDFXL           1  2.0   35   +88     264 F
f7/sum
f11/p
  g57/CI                                    +0     264
```

Slow.lib



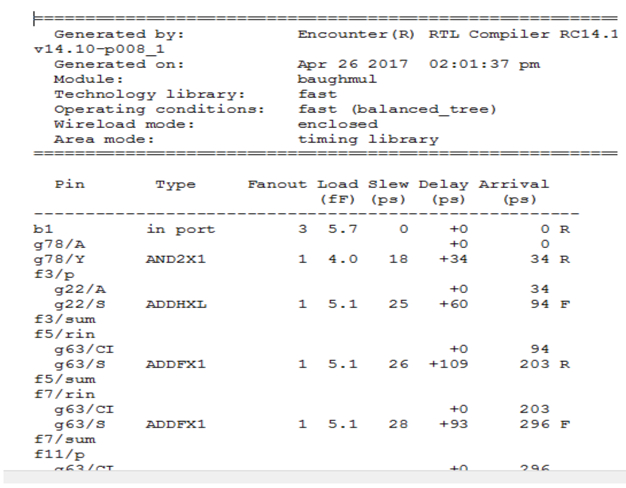
```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:            Apr 26 2017  02:05:34 pm
  Module:                  baughmul
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
--------------------------------------------------------
b0          in port          3  1.8    0    +0       0 R
g77/A                                       +0       0
g77/Y       AND2X2           1  1.6   47  +132     132 R
f2/q
  g22/B                                     +0     132
  g22/CO    ADDHX1           1  2.3   62  +131     263 R
f2/carry
f5/q
  g57/B                                     +0     263
  g57/S     ADDFXL           1  1.8   91  +306     570 F
f5/sum
f7/rin
  g57/CI                                    +0     570
  g57/S     ADDFXL           1  1.8   85  +297     867 R
f7/sum
f11/p
  g57/CI                                    +0     867
```

Timing Synthesis Report for 4bit Wallace tree multiplier using 45nm

B. Using 90nm Technology

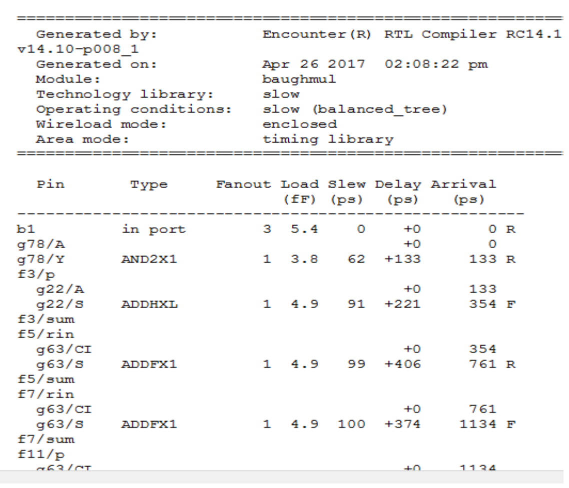
Fast.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:            Apr 26 2017  02:01:37 pm
  Module:                  baughmul
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
--------------------------------------------------------
b1          in port          3  5.7    0    +0       0 R
g78/A                                       +0       0
g78/Y       AND2X1           1  4.0   18   +34      34 R
f3/p
  g22/A                                     +0      34
  g22/S     ADDHXL           1  5.1   25   +60      94 F
f3/sum
f5/rin
  g63/CI                                    +0      94
  g63/S     ADDFX1           1  5.1   26  +109     203 R
f5/sum
f7/rin
  g63/CI                                    +0     203
  g63/S     ADDFX1           1  5.1   28   +93     296 F
f7/sum
f11/p
  g63/CI                                    +0     296
```

Slow.lib



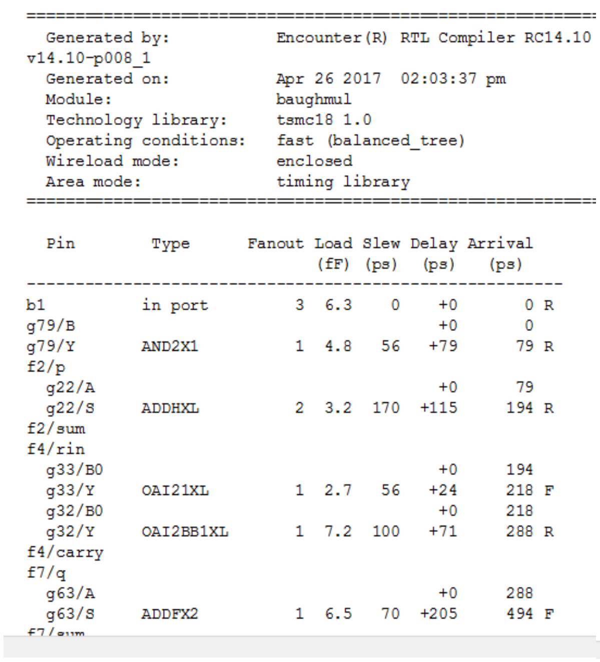
```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:            Apr 26 2017  02:08:22 pm
  Module:                  baughmul
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)    (ps)
--------------------------------------------------------
b1          in port          3  5.4    0    +0       0 R
g78/A                                       +0       0
g78/Y       AND2X1           1  3.8   62  +133     133 R
f3/p
  g22/A                                     +0     133
  g22/S     ADDHXL           1  4.9   91  +221     354 F
f3/sum
f5/rin
  g63/CI                                    +0     354
  g63/S     ADDFX1           1  4.9   99  +406     761 R
f5/sum
f7/rin
  g63/CI                                    +0     761
  g63/S     ADDFX1           1  4.9  100  +374    1134 F
f7/sum
f11/p
  g63/CI                                    +0    1134
```

Timing Synthesis Report for 4bit Wallace tree multiplier using 90nm

C. Using 180nm Technology:

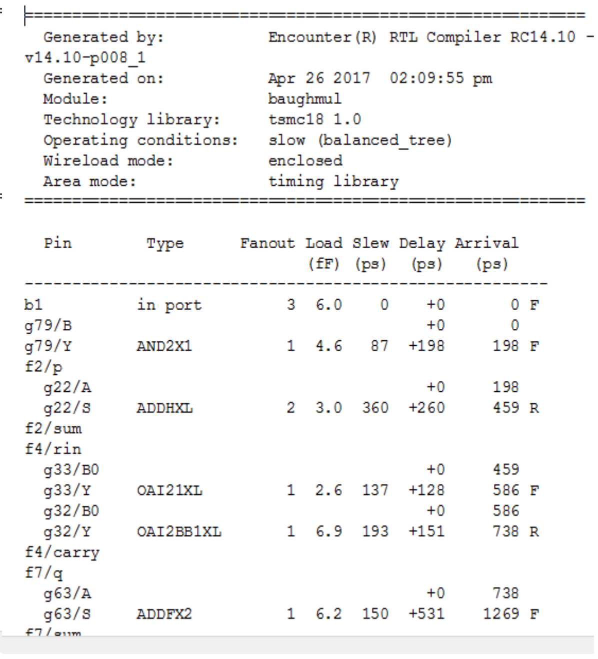
Fast.lib



```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:            Apr 26 2017  02:03:37 pm
  Module:                  baughmul
  Technology library:      tsmc18 1.0
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type       Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
-------------------------------------------------------
b1          in port          3  6.3    0   +0      0 R
g79/B                                      +0      0
g79/Y       AND2X1           1  4.8   56  +79     79 R
f2/p
  g22/A                                    +0     79
  g22/S     ADDHXL           2  3.2  170 +115    194 R
f2/sum
f4/rin
  g33/B0                                   +0    194
  g33/Y     OAI21XL          1  2.7   56  +24    218 F
  g32/B0                                   +0    218
  g32/Y     OAI2BB1XL        1  7.2  100  +71    288 R
f4/carry
f7/q
  g63/A                                    +0    288
  g63/S     ADDFX2           1  6.5   70 +205    494 F
f7/sum
```

Slow.lib



```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  02:09:55 pm
  Module:                  baughmul
  Technology library:      tsmc18 1.0
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type       Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
-------------------------------------------------------
b1          in port          3  6.0    0   +0      0 F
g79/B                                      +0      0
g79/Y       AND2X1           1  4.6   87 +198    198 F
f2/p
  g22/A                                    +0    198
  g22/S     ADDHXL           2  3.0  360 +260    459 R
f2/sum
f4/rin
  g33/B0                                   +0    459
  g33/Y     OAI21XL          1  2.6  137 +128    586 F
  g32/B0                                   +0    586
  g32/Y     OAI2BB1XL        1  6.9  193 +151    738 R
f4/carry
f7/q
  g63/A                                    +0    738
  g63/S     ADDFX2           1  6.2  150 +531   1269 F
f7/sum
```

Timing Synthesis Report for 4bit Wallace tree multiplier using 180nm

6.2.3 8BIT WALLACE TREE:-

The multiplier has been designed in Xilinx and NCsim using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

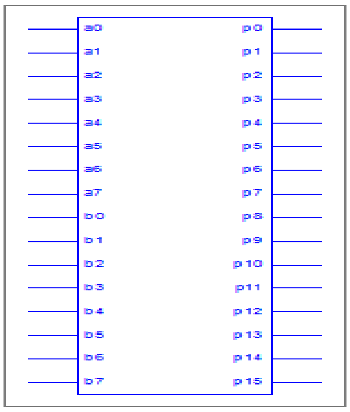


Fig.6.10. 8bit Wallace tree multiplier

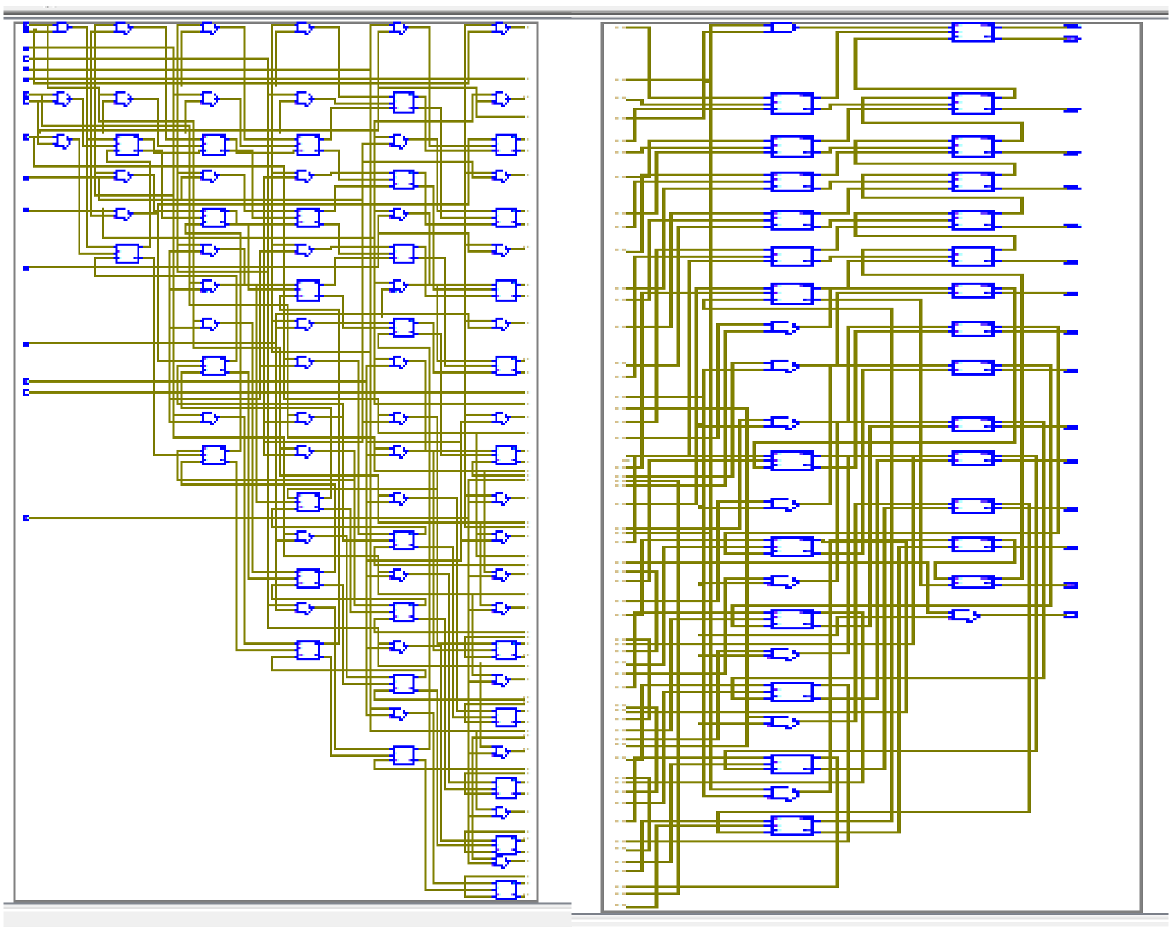


Fig.6.11. RTL view of 8bit Wallace tree multiplier

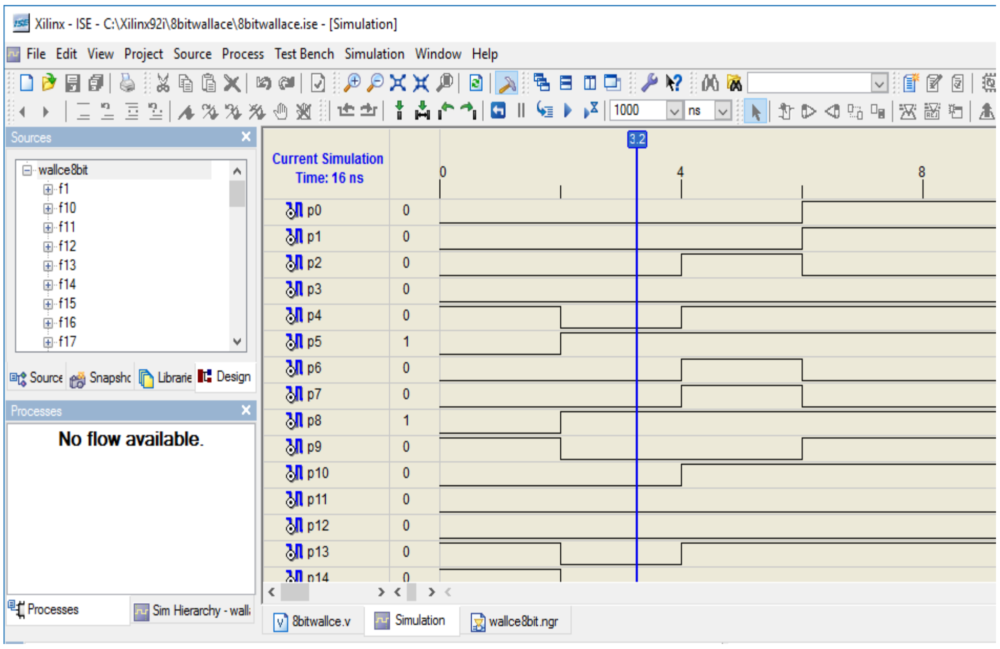


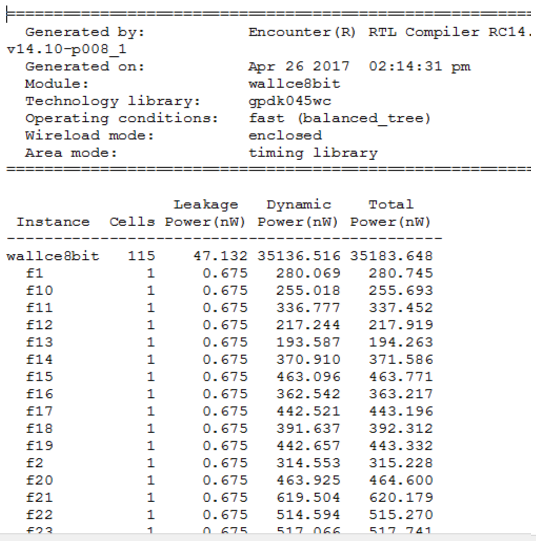
Fig.6.12. Output waveforms of the 8bit Wallace tree multiplier

6.2.4. Power Synthesis Result:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

C. Using 45nm Technology

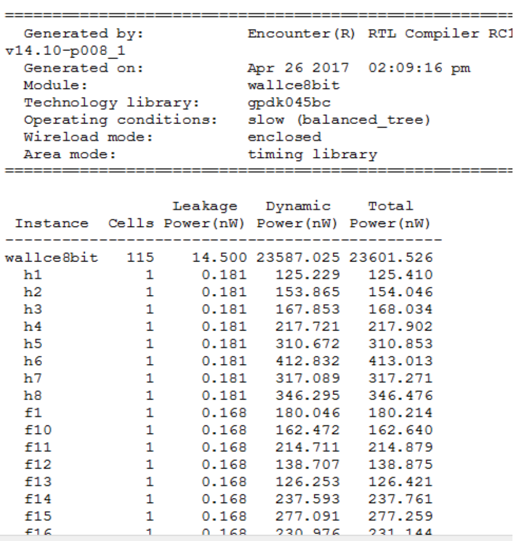
Fast.lib



```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:            Apr 26 2017  02:14:31 pm
  Module:                  wallce8bit
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

                  Leakage   Dynamic    Total
 Instance   Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------------
wallce8bit   115    47.132 35136.516 35183.648
  f1           1     0.675   280.069   280.745
  f10          1     0.675   255.018   255.693
  f11          1     0.675   336.777   337.452
  f12          1     0.675   217.244   217.919
  f13          1     0.675   193.587   194.263
  f14          1     0.675   370.910   371.586
  f15          1     0.675   463.096   463.771
  f16          1     0.675   362.542   363.217
  f17          1     0.675   442.521   443.196
  f18          1     0.675   391.637   392.312
  f19          1     0.675   442.657   443.332
  f2           1     0.675   314.553   315.228
  f20          1     0.675   463.925   464.600
  f21          1     0.675   619.504   620.179
  f22          1     0.675   514.594   515.270
  f23          1     0.675   517.066   517.741
```

Slow.lib



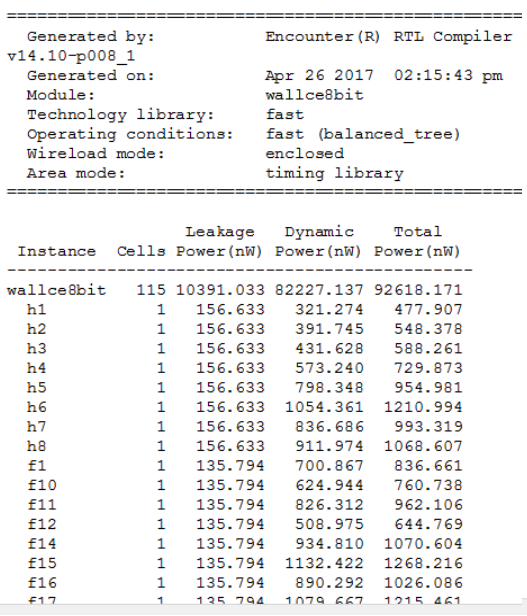
```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC1
v14.10-p008_1
  Generated on:            Apr 26 2017  02:09:16 pm
  Module:                  wallce8bit
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

                  Leakage   Dynamic    Total
 Instance   Cells Power(nW) Power(nW) Power(nW)
--------------------------------------------------
wallce8bit   115    14.500 23587.025 23601.526
  h1           1     0.181   125.229   125.410
  h2           1     0.181   153.865   154.046
  h3           1     0.181   167.853   168.034
  h4           1     0.181   217.721   217.902
  h5           1     0.181   310.672   310.853
  h6           1     0.181   412.832   413.013
  h7           1     0.181   317.089   317.271
  h8           1     0.181   346.295   346.476
  f1           1     0.168   180.046   180.214
  f10          1     0.168   162.472   162.640
  f11          1     0.168   214.711   214.879
  f12          1     0.168   138.707   138.875
  f13          1     0.168   126.253   126.421
  f14          1     0.168   237.593   237.761
  f15          1     0.168   277.091   277.259
  f16          1     0.168   230.976   231.144
```

Power synthesis report for 8bit Wallace multiplier using 45nm

C. Using 90nm Technology

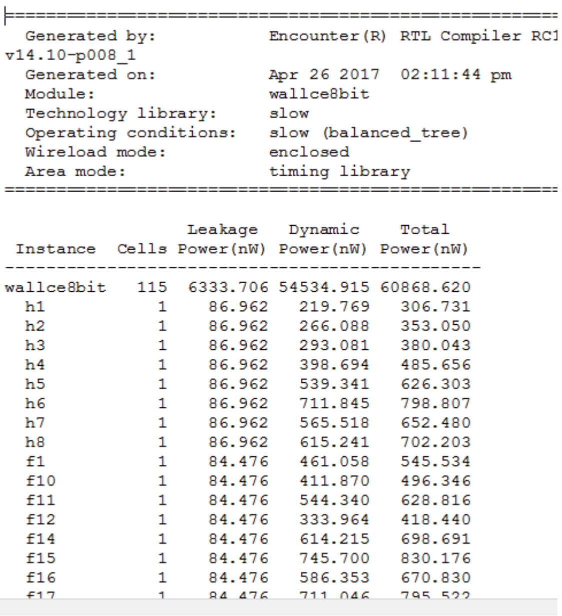
Fast.lib



```
==============================================
  Generated by:            Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:            Apr 26 2017  02:15:43 pm
  Module:                  wallce8bit
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==============================================

                Leakage   Dynamic    Total
 Instance  Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------------
wallce8bit   115 10391.033 82227.137 92618.171
  h1           1   156.633   321.274   477.907
  h2           1   156.633   391.745   548.378
  h3           1   156.633   431.628   588.261
  h4           1   156.633   573.240   729.873
  h5           1   156.633   798.348   954.981
  h6           1   156.633  1054.361  1210.994
  h7           1   156.633   836.686   993.319
  h8           1   156.633   911.974  1068.607
  f1           1   135.794   700.867   836.661
  f10          1   135.794   624.944   760.738
  f11          1   135.794   826.312   962.106
  f12          1   135.794   508.975   644.769
  f14          1   135.794   934.810  1070.604
  f15          1   135.794  1132.422  1268.216
  f16          1   135.794   890.292  1026.086
  f17          1   135.794  1079.667  1215.461
```

Slow.lib



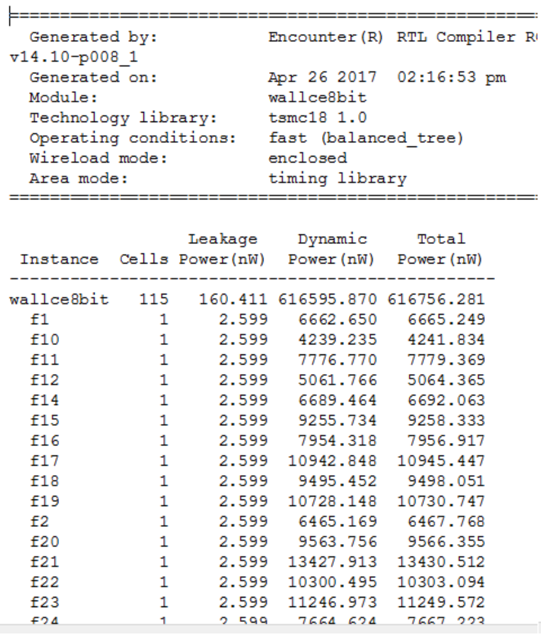
```
==============================================
  Generated by:            Encounter(R) RTL Compiler RC1
v14.10-p008_1
  Generated on:            Apr 26 2017  02:11:44 pm
  Module:                  wallce8bit
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==============================================

                Leakage   Dynamic    Total
 Instance  Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------------
wallce8bit   115  6333.706 54534.915 60868.620
  h1           1    86.962   219.769   306.731
  h2           1    86.962   266.088   353.050
  h3           1    86.962   293.081   380.043
  h4           1    86.962   398.694   485.656
  h5           1    86.962   539.341   626.303
  h6           1    86.962   711.845   798.807
  h7           1    86.962   565.518   652.480
  h8           1    86.962   615.241   702.203
  f1           1    84.476   461.058   545.534
  f10          1    84.476   411.870   496.346
  f11          1    84.476   544.340   628.816
  f12          1    84.476   333.964   418.440
  f14          1    84.476   614.215   698.691
  f15          1    84.476   745.700   830.176
  f16          1    84.476   586.353   670.830
  f17          1    84.476   711.046   795.522
```

Power synthesis report for 8bit Wallace multiplier using 90nm

C. Using 180nm Technology:

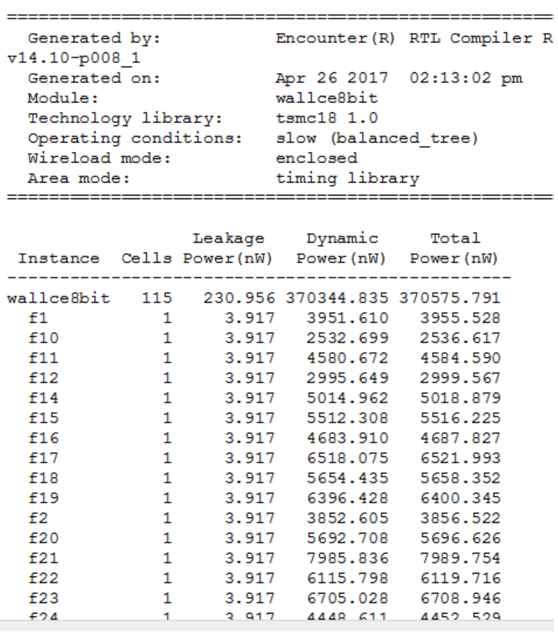
Fast.lib:



```
==============================================
  Generated by:            Encounter(R) RTL Compiler R
v14.10-p008_1
  Generated on:            Apr 26 2017  02:16:53 pm
  Module:                  wallce8bit
  Technology library:      tsmc18 1.0
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==============================================

                Leakage    Dynamic     Total
 Instance  Cells Power(nW)  Power(nW)  Power(nW)
-------------------------------------------------
wallce8bit   115   160.411 616595.870 616756.281
  f1           1     2.599   6662.650   6665.249
  f10          1     2.599   4239.235   4241.834
  f11          1     2.599   7776.770   7779.369
  f12          1     2.599   5061.766   5064.365
  f14          1     2.599   6689.464   6692.063
  f15          1     2.599   9255.734   9258.333
  f16          1     2.599   7954.318   7956.917
  f17          1     2.599  10942.848  10945.447
  f18          1     2.599   9495.452   9498.051
  f19          1     2.599  10728.148  10730.747
  f2           1     2.599   6465.169   6467.768
  f20          1     2.599   9563.756   9566.355
  f21          1     2.599  13427.913  13430.512
  f22          1     2.599  10300.495  10303.094
  f23          1     2.599  11246.973  11249.572
  f24          1     2.599   7664.624   7667.223
```

Slow.lib:



```
==============================================
  Generated by:            Encounter(R) RTL Compiler R
v14.10-p008_1
  Generated on:            Apr 26 2017  02:13:02 pm
  Module:                  wallce8bit
  Technology library:      tsmc18 1.0
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==============================================

                Leakage    Dynamic     Total
 Instance  Cells Power(nW)  Power(nW)  Power(nW)
-------------------------------------------------
wallce8bit   115   230.956 370344.835 370575.791
  f1           1     3.917   3951.610   3955.528
  f10          1     3.917   2532.699   2536.617
  f11          1     3.917   4580.672   4584.590
  f12          1     3.917   2995.649   2999.567
  f14          1     3.917   5014.962   5018.879
  f15          1     3.917   5512.308   5516.225
  f16          1     3.917   4683.910   4687.827
  f17          1     3.917   6518.075   6521.993
  f18          1     3.917   5654.435   5658.352
  f19          1     3.917   6396.428   6400.345
  f2           1     3.917   3852.605   3856.522
  f20          1     3.917   5692.708   5696.626
  f21          1     3.917   7985.836   7989.754
  f22          1     3.917   6115.798   6119.716
  f23          1     3.917   6705.028   6708.946
  f24          1     3.917   4448.611   4452.529
```

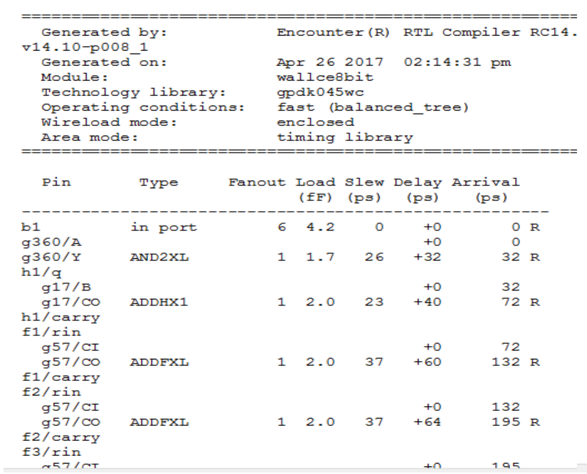
Power synthesis report for 8bit Wallace multiplier using 180nm

6.2.5 Delay and Timing Synthesis Results:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

C. Using 45nm Technology

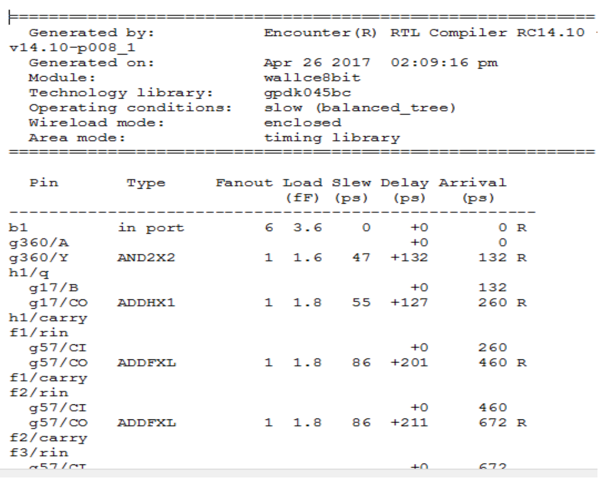
Fast.lib



```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:            Apr 26 2017  02:14:31 pm
  Module:                  wallce8bit
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type     Fanout Load Slew Delay Arrival
                             (fF) (ps)  (ps)    (ps)
-------------------------------------------------------
b1          in port        6  4.2    0    +0       0 R
g360/A                                    +0       0
g360/Y      AND2XL         1  1.7   26   +32      32 R
h1/q
  g17/B                                   +0      32
  g17/CO    ADDHX1         1  2.0   23   +40      72 R
h1/carry
f1/rin
  g57/CI                                  +0      72
  g57/CO    ADDFXL         1  2.0   37   +60     132 R
f1/carry
f2/rin
  g57/CI                                  +0     132
  g57/CO    ADDFXL         1  2.0   37   +64     195 R
f2/carry
f3/rin
  g57/CI                                  +0     195
```

Slow.lib



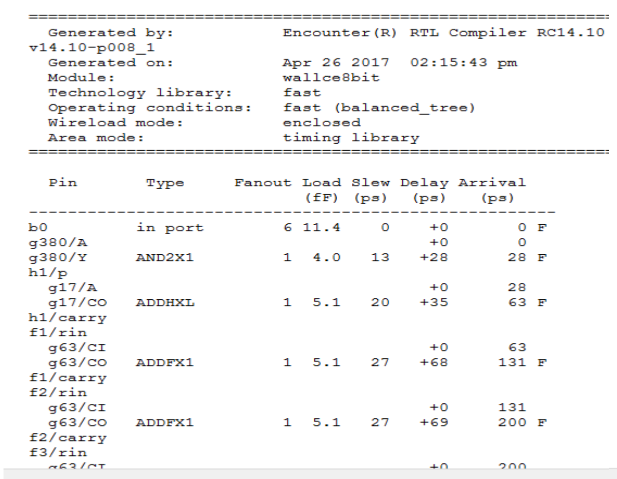
```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 
v14.10-p008_1
  Generated on:            Apr 26 2017  02:09:16 pm
  Module:                  wallce8bit
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type     Fanout Load Slew Delay Arrival
                             (fF) (ps)  (ps)    (ps)
-------------------------------------------------------
b1          in port        6  3.6    0    +0       0 R
g360/A                                    +0       0
g360/Y      AND2X2         1  1.6   47  +132     132 R
h1/q
  g17/B                                   +0     132
  g17/CO    ADDHX1         1  1.8   55  +127     260 R
h1/carry
f1/rin
  g57/CI                                  +0     260
  g57/CO    ADDFXL         1  1.8   86  +201     460 R
f1/carry
f2/rin
  g57/CI                                  +0     460
  g57/CO    ADDFXL         1  1.8   86  +211     672 R
f2/carry
f3/rin
  g57/CI                                  +0     672
```

Timing Synthesis Report for 8bit Wallace multiplier using 45nm

C. Using 90nm Technology

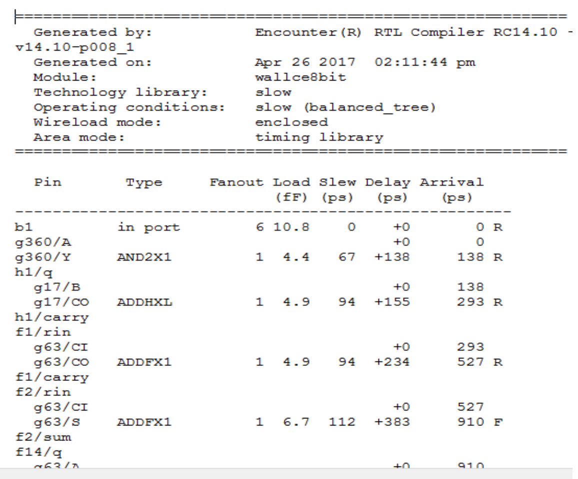
Fast.lib



```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10
v14.10-p008_1
  Generated on:            Apr 26 2017  02:15:43 pm
  Module:                  wallce8bit
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type     Fanout Load Slew Delay Arrival
                             (fF) (ps)  (ps)    (ps)
-------------------------------------------------------
b0          in port        6 11.4    0    +0       0 F
g380/A                                    +0       0
g380/Y      AND2X1         1  4.0   13   +28      28 F
h1/p
  g17/A                                   +0      28
  g17/CO    ADDHXL         1  5.1   20   +35      63 F
h1/carry
f1/rin
  g63/CI                                  +0      63
  g63/CO    ADDFX1         1  5.1   27   +68     131 F
f1/carry
f2/rin
  g63/CI                                  +0     131
  g63/CO    ADDFX1         1  5.1   27   +69     200 F
f2/carry
f3/rin
  g63/CI                                  +0     200
```

Slow.lib



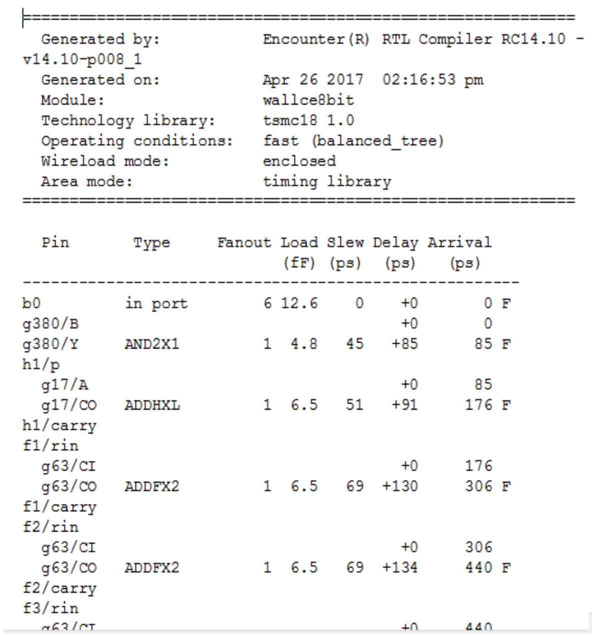
```
==========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  02:11:44 pm
  Module:                  wallce8bit
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
==========================================================

  Pin        Type     Fanout Load Slew Delay Arrival
                             (fF) (ps)  (ps)    (ps)
-------------------------------------------------------
b1          in port        6 10.8    0    +0       0 R
g360/A                                    +0       0
g360/Y      AND2X1         1  4.4   67  +138     138 R
h1/q
  g17/B                                   +0     138
  g17/CO    ADDHXL         1  4.9   94  +155     293 R
h1/carry
f1/rin
  g63/CI                                  +0     293
  g63/CO    ADDFX1         1  4.9   94  +234     527 R
f1/carry
f2/rin
  g63/CI                                  +0     527
  g63/S     ADDFX1         1  6.7  112  +383     910 F
f2/sum
f14/q
  g63/A                                   +0     910
```

Timing Synthesis Report for 8bit Wallace multiplier using 90nm

C. Using 180nm Technology:

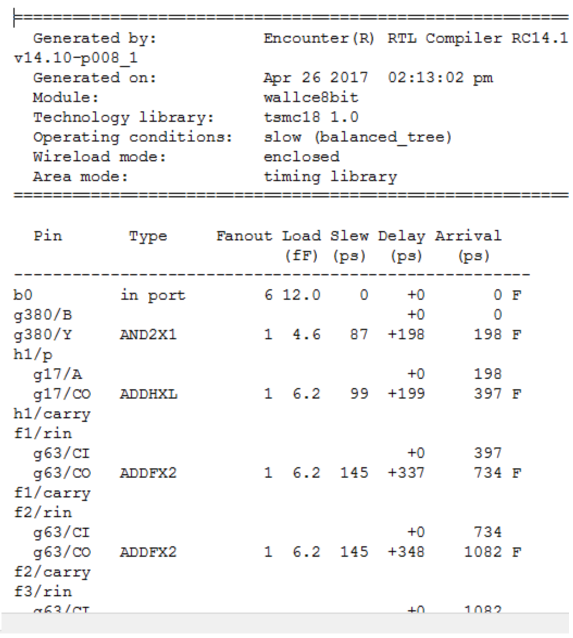
Fast.lib



```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:          Apr 26 2017  02:16:53 pm
  Module:                wallce8bit
  Technology library:    tsmc18 1.0
  Operating conditions:  fast (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------
b0          in port          6 12.6    0    +0      0 F
g380/B                                      +0      0
g380/Y      AND2X1           1  4.8   45   +85     85 F
h1/p
  g17/A                                     +0     85
  g17/CO    ADDHXL           1  6.5   51   +91    176 F
h1/carry
f1/rin
  g63/CI                                    +0    176
  g63/CO    ADDFX2           1  6.5   69  +130    306 F
f1/carry
f2/rin
  g63/CI                                    +0    306
  g63/CO    ADDFX2           1  6.5   69  +134    440 F
f2/carry
f3/rin
  g63/CI                                    +0    440
```

Slow.lib



```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:          Apr 26 2017  02:13:02 pm
  Module:                wallce8bit
  Technology library:    tsmc18 1.0
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------
b0          in port          6 12.0    0    +0      0 F
g380/B                                      +0      0
g380/Y      AND2X1           1  4.6   87  +198    198 F
h1/p
  g17/A                                     +0    198
  g17/CO    ADDHXL           1  6.2   99  +199    397 F
h1/carry
f1/rin
  g63/CI                                    +0    397
  g63/CO    ADDFX2           1  6.2  145  +337    734 F
f1/carry
f2/rin
  g63/CI                                    +0    734
  g63/CO    ADDFX2           1  6.2  145  +348   1082 F
f2/carry
f3/rin
  g63/CI                                    +0   1082
```

Timing Synthesis Report for 8bit Wallace multiplier using 180nm

6.3 BAUGH WOOLEY MULTIPLIER

6.3.1. Baugh wooley 4bit multiplier

The multiplier has been designed in the Xilinx, Ncsim using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

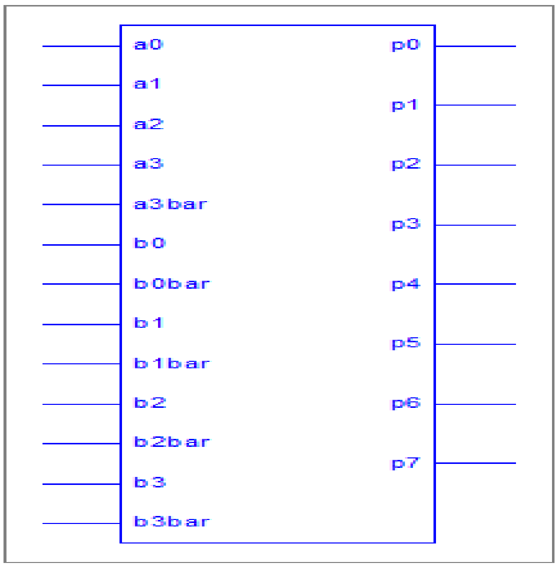


Fig6.13. Baugh wooley 4bit multiplier

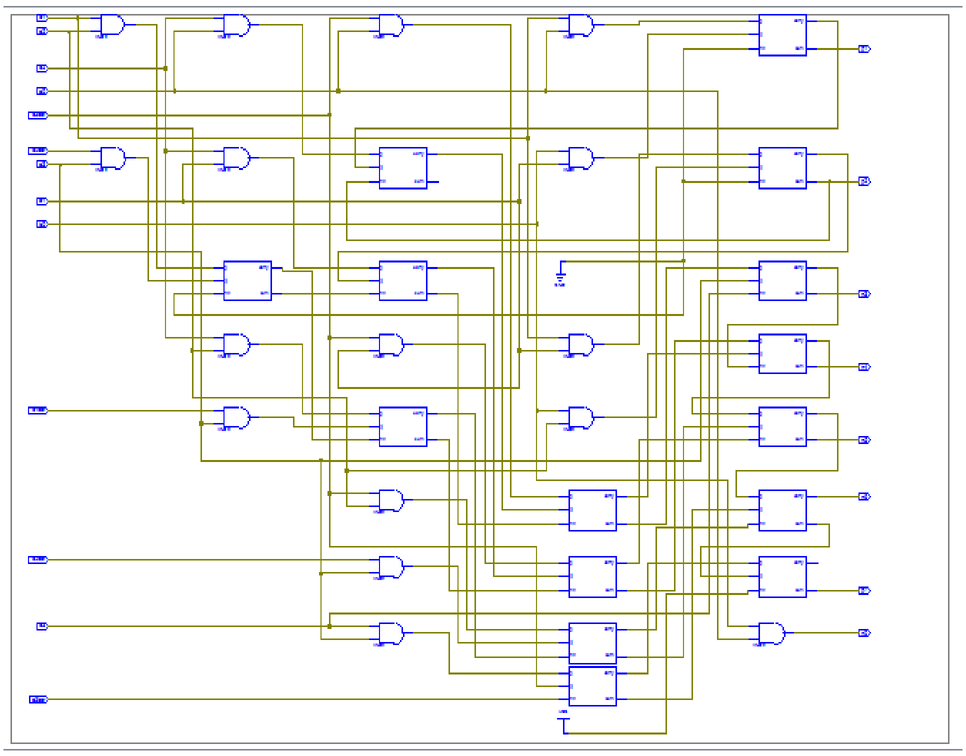


Fig6.14. RTL View of Baugh wooley 4bit multiplier

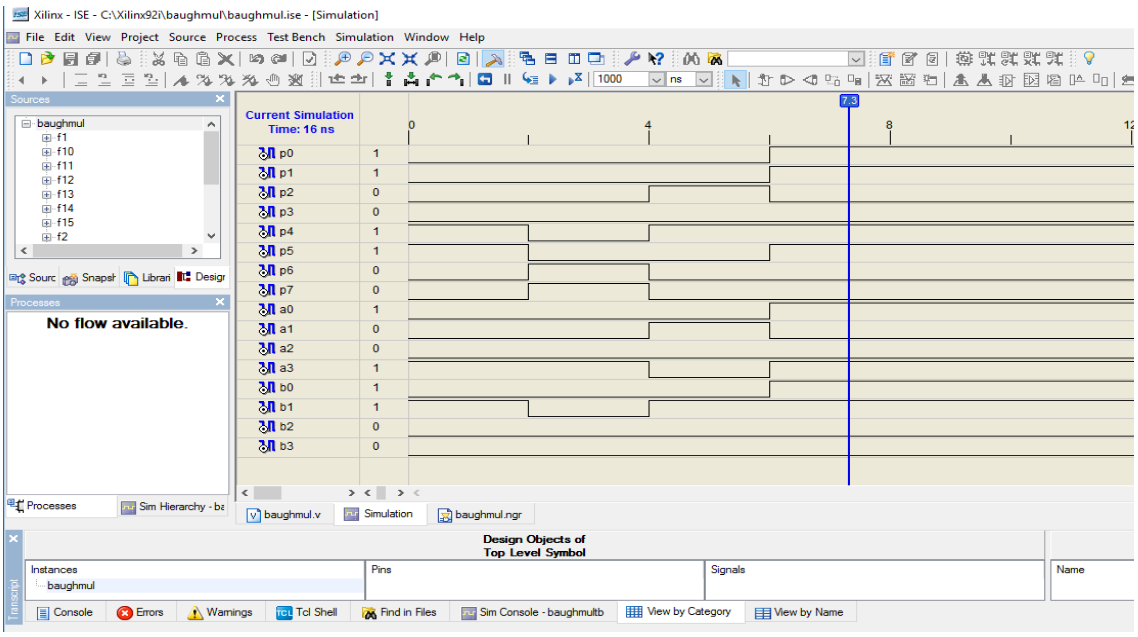


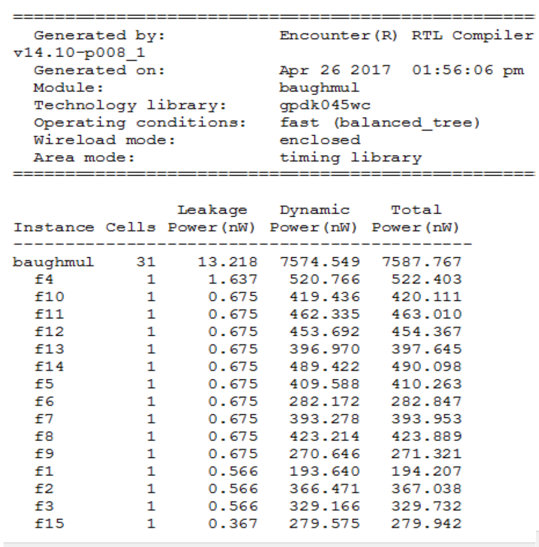
Fig6.15. Output waveform of Baugh wooley 4bit multiplier

6.3.2 Power Synthesis Result:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

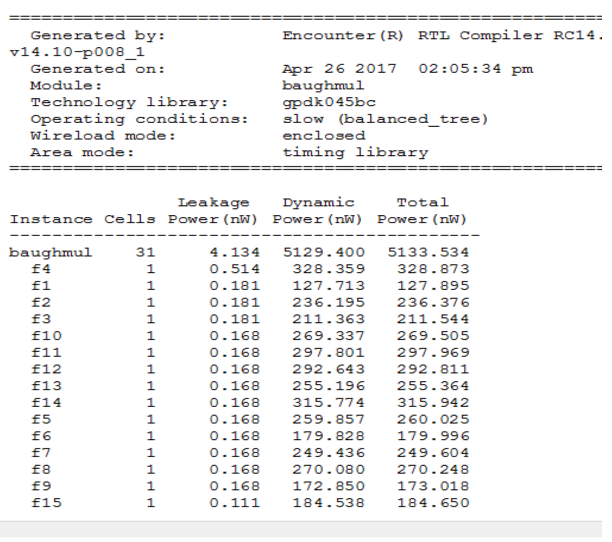
Fast.lib



```
======================================================
  Generated by:            Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:            Apr 26 2017  01:56:06 pm
  Module:                  baughmul
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
======================================================

                  Leakage   Dynamic    Total
Instance Cells  Power(nW) Power(nW) Power(nW)
----------------------------------------------
baughmul    31     13.218  7574.549  7587.767
  f4         1      1.637   520.766   522.403
  f10        1      0.675   419.436   420.111
  f11        1      0.675   462.335   463.010
  f12        1      0.675   453.692   454.367
  f13        1      0.675   396.970   397.645
  f14        1      0.675   489.422   490.098
  f5         1      0.675   409.588   410.263
  f6         1      0.675   282.172   282.847
  f7         1      0.675   393.278   393.953
  f8         1      0.675   423.214   423.889
  f9         1      0.675   270.646   271.321
  f1         1      0.566   193.640   194.207
  f2         1      0.566   366.471   367.038
  f3         1      0.566   329.166   329.732
  f15        1      0.367   279.575   279.942
```

Slow.lib



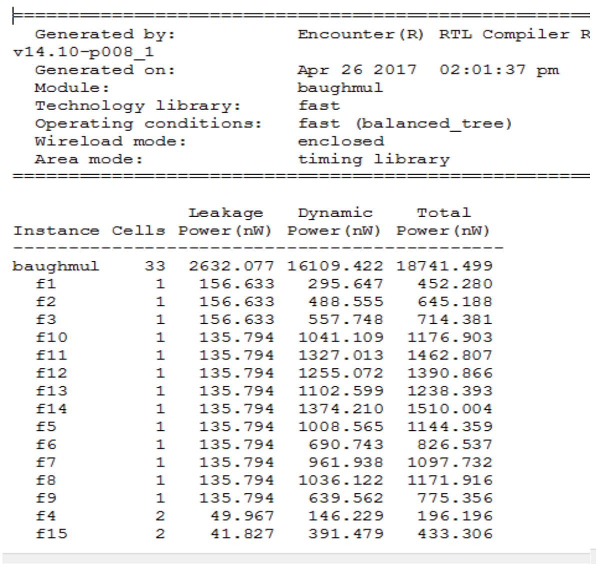
```
===========================================================
  Generated by:            Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:            Apr 26 2017  02:05:34 pm
  Module:                  baughmul
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
===========================================================

                  Leakage   Dynamic    Total
Instance Cells  Power(nW) Power(nW) Power(nW)
----------------------------------------------
baughmul    31      4.134  5129.400  5133.534
  f4         1      0.514   328.359   328.873
  f1         1      0.181   127.713   127.895
  f2         1      0.181   236.195   236.376
  f3         1      0.181   211.363   211.544
  f10        1      0.168   269.337   269.505
  f11        1      0.168   297.801   297.969
  f12        1      0.168   292.643   292.811
  f13        1      0.168   255.196   255.364
  f14        1      0.168   315.774   315.942
  f5         1      0.168   259.857   260.025
  f6         1      0.168   179.828   179.996
  f7         1      0.168   249.436   249.604
  f8         1      0.168   270.080   270.248
  f9         1      0.168   172.850   173.018
  f15        1      0.111   184.538   184.650
```

Power synthesis report for 4bit Baugh wooley multiplier using 45nm

B. Using 90nm Technology

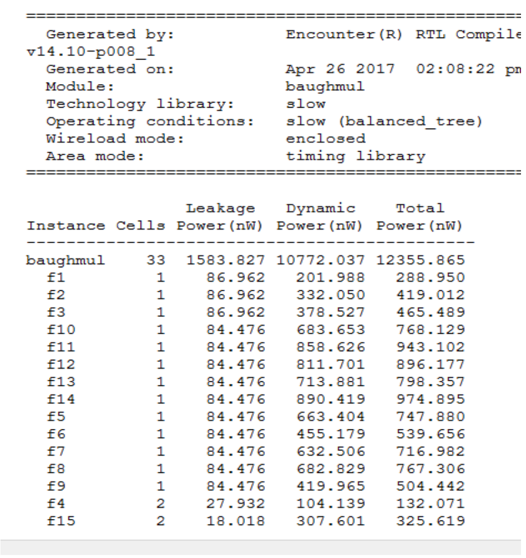
Fast.lib



```
===================================================================
  Generated by:             Encounter(R) RTL Compiler R
v14.10-p008_1
  Generated on:             Apr 26 2017  02:01:37 pm
  Module:                   baughmul
  Technology library:       fast
  Operating conditions:     fast (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
===================================================================

                   Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------
baughmul    33   2632.077 16109.422 18741.499
  f1         1    156.633   295.647   452.280
  f2         1    156.633   488.555   645.188
  f3         1    156.633   557.748   714.381
  f10        1    135.794  1041.109  1176.903
  f11        1    135.794  1327.013  1462.807
  f12        1    135.794  1255.072  1390.866
  f13        1    135.794  1102.599  1238.393
  f14        1    135.794  1374.210  1510.004
  f5         1    135.794  1008.565  1144.359
  f6         1    135.794   690.743   826.537
  f7         1    135.794   961.938  1097.732
  f8         1    135.794  1036.122  1171.916
  f9         1    135.794   639.562   775.356
  f4         2     49.967   146.229   196.196
  f15        2     41.827   391.479   433.306
```

Slow.lib



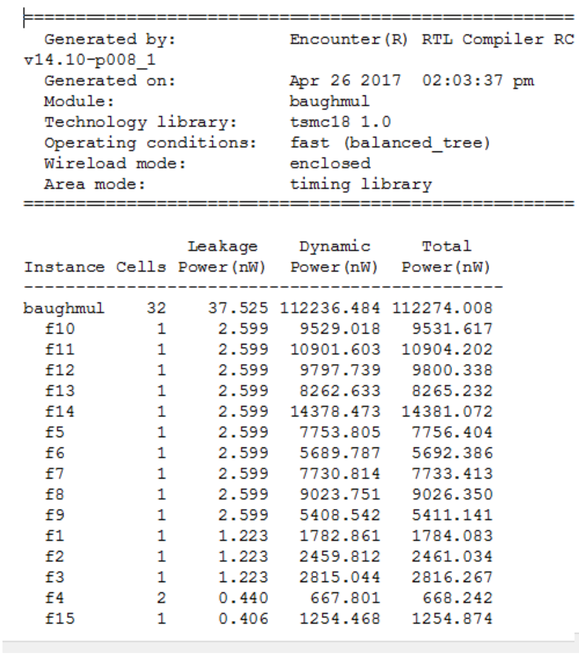
```
===================================================================
  Generated by:             Encounter(R) RTL Compile
v14.10-p008_1
  Generated on:             Apr 26 2017  02:08:22 pm
  Module:                   baughmul
  Technology library:       slow
  Operating conditions:     slow (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
===================================================================

                   Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------
baughmul    33   1583.827 10772.037 12355.865
  f1         1     86.962   201.988   288.950
  f2         1     86.962   332.050   419.012
  f3         1     86.962   378.527   465.489
  f10        1     84.476   683.653   768.129
  f11        1     84.476   858.626   943.102
  f12        1     84.476   811.701   896.177
  f13        1     84.476   713.881   798.357
  f14        1     84.476   890.419   974.895
  f5         1     84.476   663.404   747.880
  f6         1     84.476   455.179   539.656
  f7         1     84.476   632.506   716.982
  f8         1     84.476   682.829   767.306
  f9         1     84.476   419.965   504.442
  f4         2     27.932   104.139   132.071
  f15        2     18.018   307.601   325.619
```

Power synthesis report for 4bit Baugh wooley multiplier using 90nm

C. Using 180nm Technology:

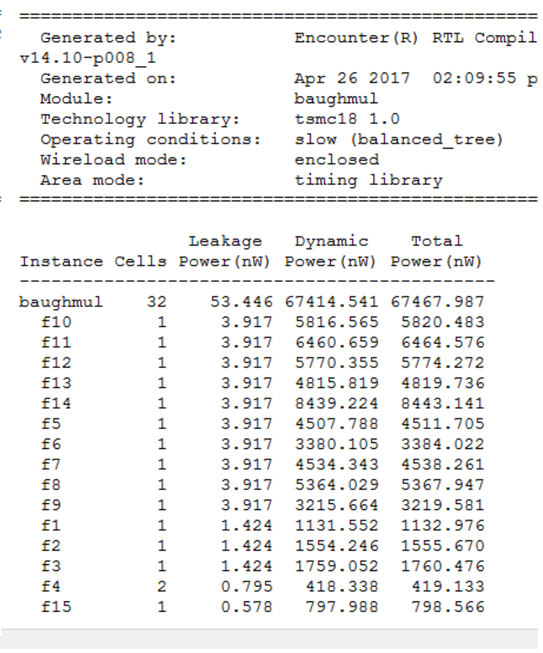
Fast.lib:



```
===================================================================
  Generated by:             Encounter(R) RTL Compiler RC
v14.10-p008_1
  Generated on:             Apr 26 2017  02:03:37 pm
  Module:                   baughmul
  Technology library:       tsmc18 1.0
  Operating conditions:     fast (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
===================================================================

                   Leakage   Dynamic      Total
Instance Cells Power(nW) Power(nW)   Power(nW)
-------------------------------------------------
baughmul    32     37.525 112236.484 112274.008
  f10        1      2.599   9529.018   9531.617
  f11        1      2.599  10901.603  10904.202
  f12        1      2.599   9797.739   9800.338
  f13        1      2.599   8262.633   8265.232
  f14        1      2.599  14378.473  14381.072
  f5         1      2.599   7753.805   7756.404
  f6         1      2.599   5689.787   5692.386
  f7         1      2.599   7730.814   7733.413
  f8         1      2.599   9023.751   9026.350
  f9         1      2.599   5408.542   5411.141
  f1         1      1.223   1782.861   1784.083
  f2         1      1.223   2459.812   2461.034
  f3         1      1.223   2815.044   2816.267
  f4         2      0.440    667.801    668.242
  f15        1      0.406   1254.468   1254.874
```

Slow.lib:



```
===================================================================
  Generated by:             Encounter(R) RTL Compil
v14.10-p008_1
  Generated on:             Apr 26 2017  02:09:55 p
  Module:                   baughmul
  Technology library:       tsmc18 1.0
  Operating conditions:     slow (balanced_tree)
  Wireload mode:            enclosed
  Area mode:                timing library
===================================================================

                   Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------
baughmul    32     53.446 67414.541 67467.987
  f10        1      3.917  5816.565  5820.483
  f11        1      3.917  6460.659  6464.576
  f12        1      3.917  5770.355  5774.272
  f13        1      3.917  4815.819  4819.736
  f14        1      3.917  8439.224  8443.141
  f5         1      3.917  4507.788  4511.705
  f6         1      3.917  3380.105  3384.022
  f7         1      3.917  4534.343  4538.261
  f8         1      3.917  5364.029  5367.947
  f9         1      3.917  3215.664  3219.581
  f1         1      1.424  1131.552  1132.976
  f2         1      1.424  1554.246  1555.670
  f3         1      1.424  1759.052  1760.476
  f4         2      0.795   418.338   419.133
  f15        1      0.578   797.988   798.566
```

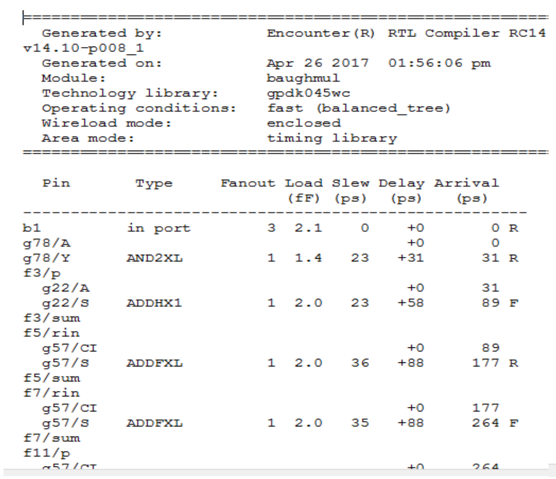
Power synthesis report for 4bit Baugh wooley multiplier using 180nm

6.3.3 Delay and Timing Synthesis Results:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

A. Using 45nm Technology

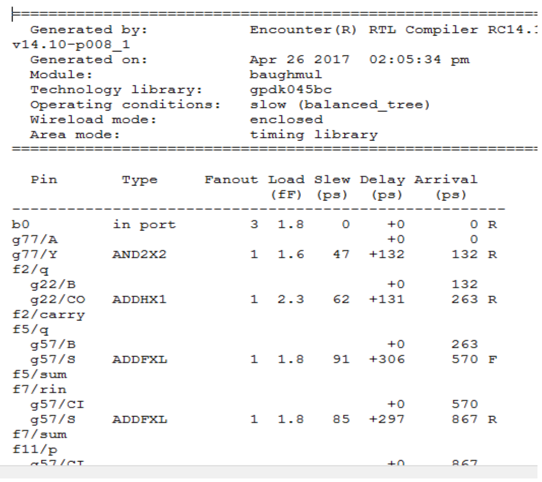
Fast.lib



```
========================================================================
  Generated by:              Encounter(R) RTL Compiler RC14
v14.10-p008_1
  Generated on:              Apr 26 2017  01:56:06 pm
  Module:                    baughmul
  Technology library:        gpdk045wc
  Operating conditions:      fast (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------
b1           in port          3  2.1    0    +0       0 R
g78/A                                         +0       0
g78/Y        AND2XL           1  1.4   23   +31      31 R
f3/p
  g22/A                                       +0      31
  g22/S      ADDHX1           1  2.0   23   +58      89 F
f3/sum
f5/rin
  g57/CI                                      +0      89
  g57/S      ADDFXL           1  2.0   36   +88     177 R
f5/sum
f7/rin
  g57/CI                                      +0     177
  g57/S      ADDFXL           1  2.0   35   +88     264 F
f7/sum
f11/p
  g57/CI                                      +0     264
```

Slow.lib



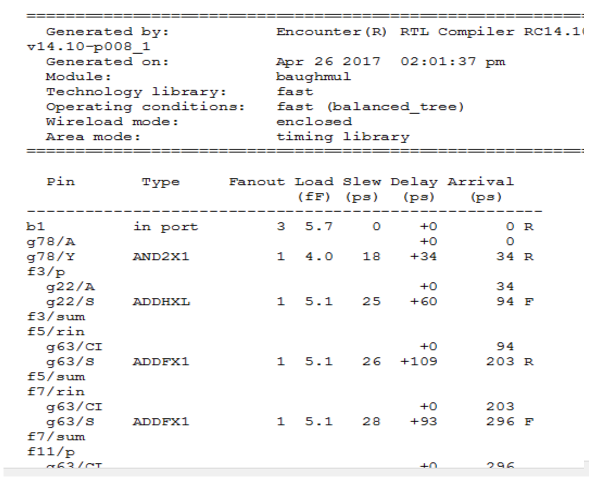
```
========================================================================
  Generated by:              Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:              Apr 26 2017  02:05:34 pm
  Module:                    baughmul
  Technology library:        gpdk045bc
  Operating conditions:      slow (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------
b0           in port          3  1.8    0    +0       0 R
g77/A                                         +0       0
g77/Y        AND2X2           1  1.6   47  +132     132 R
f2/q
  g22/B                                       +0     132
  g22/CO     ADDHX1           1  2.3   62  +131     263 R
f2/carry
f5/q
  g57/B                                       +0     263
  g57/S      ADDFXL           1  1.8   91  +306     570 F
f5/sum
f7/rin
  g57/CI                                      +0     570
  g57/S      ADDFXL           1  1.8   85  +297     867 R
f7/sum
f11/p
  g57/CI                                      +0     867
```

Timing Synthesis Report for 4bit Baugh wooley multiplier using 45nm

B. Using 90nm Technology

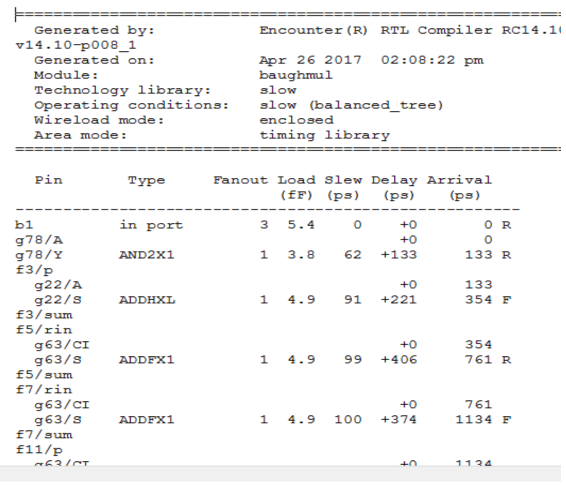
Fast.lib



```
========================================================================
  Generated by:              Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:              Apr 26 2017  02:01:37 pm
  Module:                    baughmul
  Technology library:        fast
  Operating conditions:      fast (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------
b1           in port          3  5.7    0    +0       0 R
g78/A                                         +0       0
g78/Y        AND2X1           1  4.0   18   +34      34 R
f3/p
  g22/A                                       +0      34
  g22/S      ADDHXL           1  5.1   25   +60      94 F
f3/sum
f5/rin
  g63/CI                                      +0      94
  g63/S      ADDFX1           1  5.1   26  +109     203 R
f5/sum
f7/rin
  g63/CI                                      +0     203
  g63/S      ADDFX1           1  5.1   28   +93     296 F
f7/sum
f11/p
  g63/CI                                      +0     296
```

Slow.lib



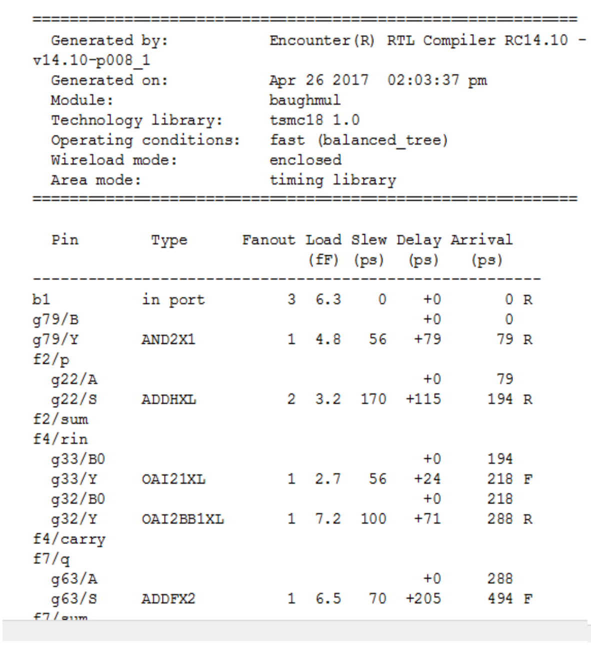
```
========================================================================
  Generated by:              Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:              Apr 26 2017  02:08:22 pm
  Module:                    baughmul
  Technology library:        slow
  Operating conditions:      slow (balanced_tree)
  Wireload mode:             enclosed
  Area mode:                 timing library
========================================================================

  Pin         Type       Fanout Load Slew Delay Arrival
                                (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------
b1           in port          3  5.4    0    +0       0 R
g78/A                                         +0       0
g78/Y        AND2X1           1  3.8   62  +133     133 R
f3/p
  g22/A                                       +0     133
  g22/S      ADDHXL           1  4.9   91  +221     354 F
f3/sum
f5/rin
  g63/CI                                      +0     354
  g63/S      ADDFX1           1  4.9   99  +406     761 R
f5/sum
f7/rin
  g63/CI                                      +0     761
  g63/S      ADDFX1           1  4.9  100  +374    1134 F
f7/sum
f11/p
  g63/CI                                      +0    1134
```

Timing Synthesis Report for 4 bit Baugh wooley multiplier using 90nm

C. Using 180nm Technology:

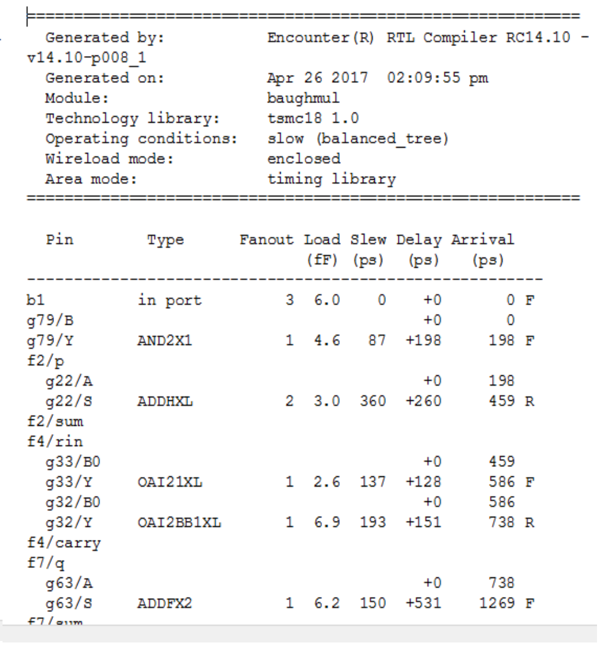
Fast.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  02:03:37 pm
  Module:                  baughmul
  Technology library:      tsmc18 1.0
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
--------------------------------------------------------
b1           in port        3  6.3    0   +0       0 R
g79/B                                     +0       0
g79/Y        AND2X1         1  4.8   56  +79      79 R
f2/p
  g22/A                                   +0      79
  g22/S      ADDHXL         2  3.2  170 +115     194 R
f2/sum
f4/rin
  g33/B0                                  +0     194
  g33/Y      OAI21XL        1  2.7   56  +24     218 F
  g32/B0                                  +0     218
  g32/Y      OAI2BB1XL      1  7.2  100  +71     288 R
f4/carry
f7/q
  g63/A                                   +0     288
  g63/S      ADDFX2         1  6.5   70 +205     494 F
f7/sum
```

Slow.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  02:09:55 pm
  Module:                  baughmul
  Technology library:      tsmc18 1.0
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
--------------------------------------------------------
b1           in port        3  6.0    0   +0       0 F
g79/B                                     +0       0
g79/Y        AND2X1         1  4.6   87 +198     198 F
f2/p
  g22/A                                   +0     198
  g22/S      ADDHXL         2  3.0  360 +260     459 R
f2/sum
f4/rin
  g33/B0                                  +0     459
  g33/Y      OAI21XL        1  2.6  137 +128     586 F
  g32/B0                                  +0     586
  g32/Y      OAI2BB1XL      1  6.9  193 +151     738 R
f4/carry
f7/q
  g63/A                                   +0     738
  g63/S      ADDFX2         1  6.2  150 +531    1269 F
f7/sum
```

Timing Synthesis Report for 4bit Baugh wooley multiplier using 180nm

6.3.4 Baugh wooley 8bit Multiplier:-

The multiplier has been designed in Xilinx and NCsim using gate level modelling for verification we have taken the results and verified with industry standard cadence tools.

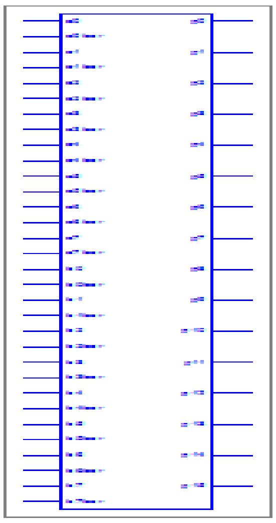


Fig.6.16. 8bit Baugh Wooley multiplier

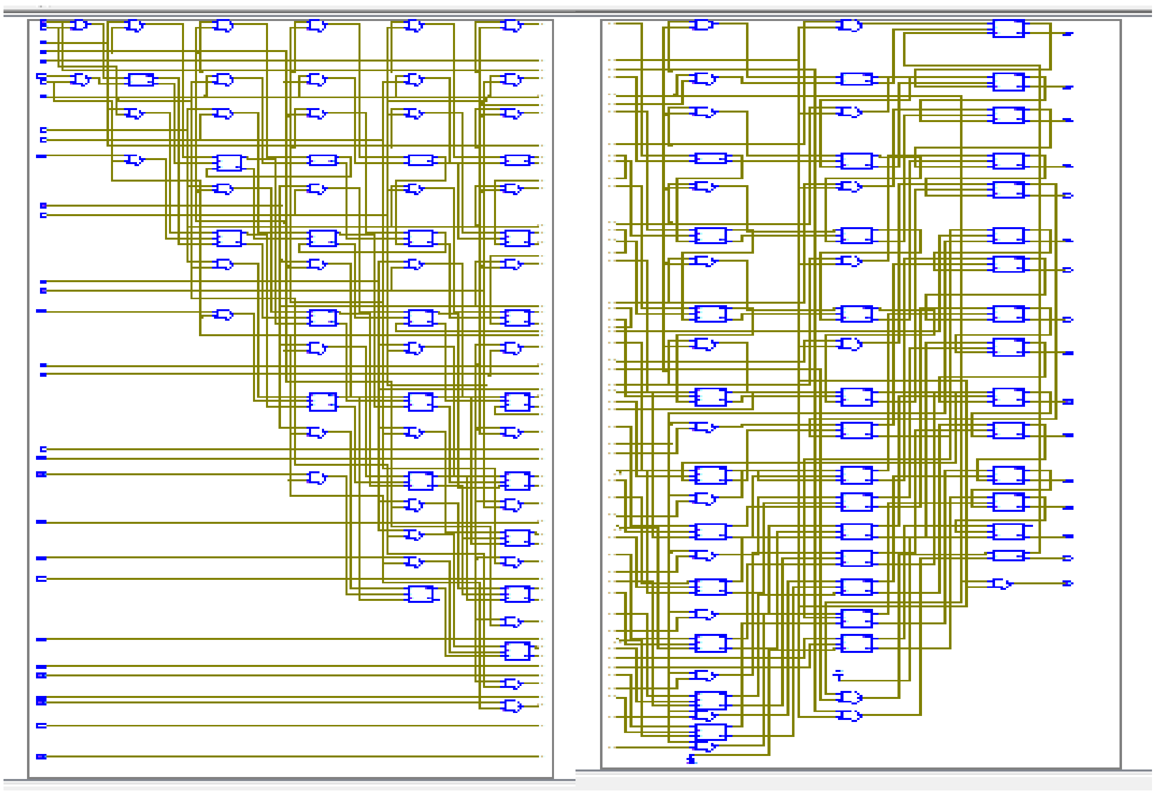


Fig.6.17. RTL view of 8bit Baugh Wooley multiplier

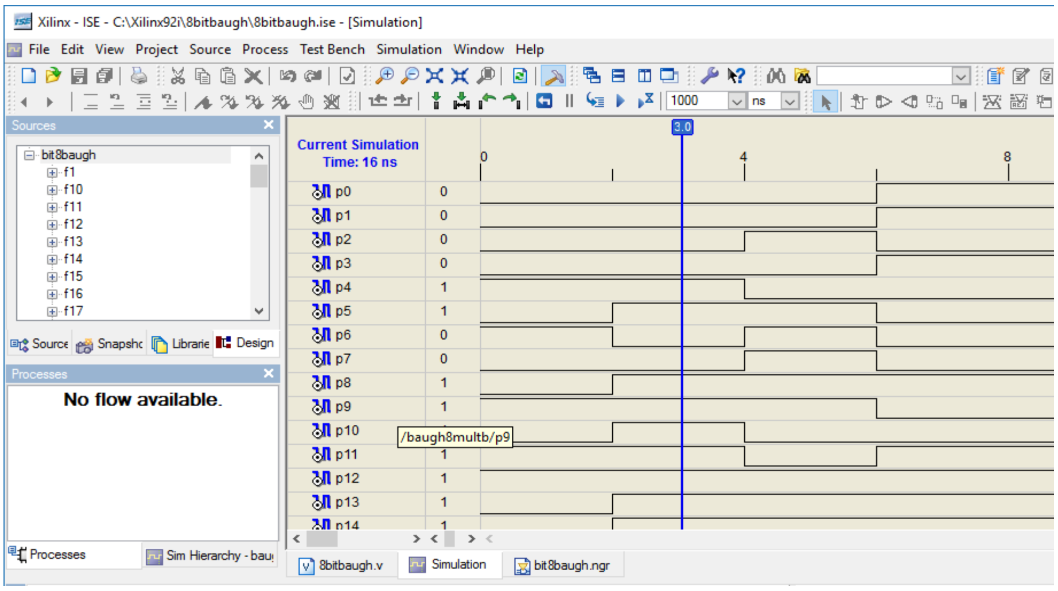


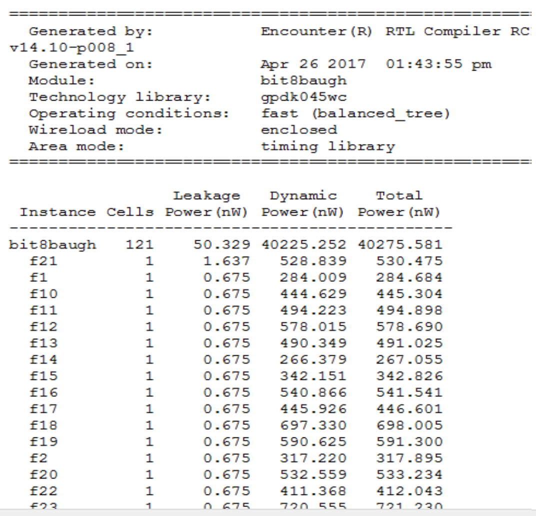
Fig.6.18. Output waveforms of the 8bit Baugh Wooley multiplier

6.3.5 Power Synthesis Result:

For the synthesis, here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

D. Using 45nm Technology

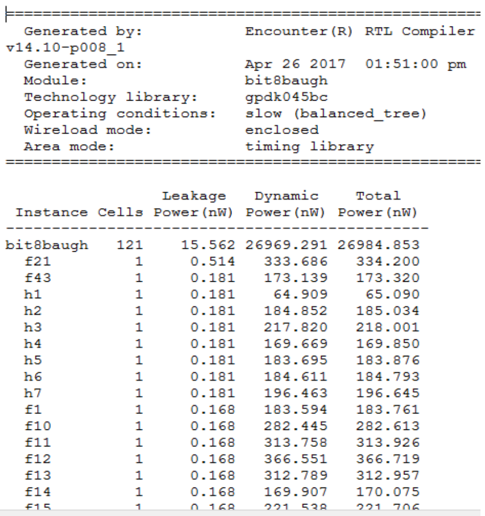
Fast.lib



```
====================================================================
  Generated by:            Encounter(R) RTL Compiler RC
v14.10-p008_1
  Generated on:            Apr 26 2017  01:43:55 pm
  Module:                  bit8baugh
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
====================================================================

                    Leakage   Dynamic    Total
 Instance  Cells Power(nW) Power(nW) Power(nW)
----------------------------------------------
bit8baugh   121    50.329 40225.252 40275.581
  f21         1     1.637   528.839   530.475
  f1          1     0.675   284.009   284.684
  f10         1     0.675   444.629   445.304
  f11         1     0.675   494.223   494.898
  f12         1     0.675   578.015   578.690
  f13         1     0.675   490.349   491.025
  f14         1     0.675   266.379   267.055
  f15         1     0.675   342.151   342.826
  f16         1     0.675   540.866   541.541
  f17         1     0.675   445.926   446.601
  f18         1     0.675   697.330   698.005
  f19         1     0.675   590.625   591.300
  f2          1     0.675   317.220   317.895
  f20         1     0.675   532.559   533.234
  f22         1     0.675   411.368   412.043
  f23         1     0.675   720.555   721.230
```

Slow.lib



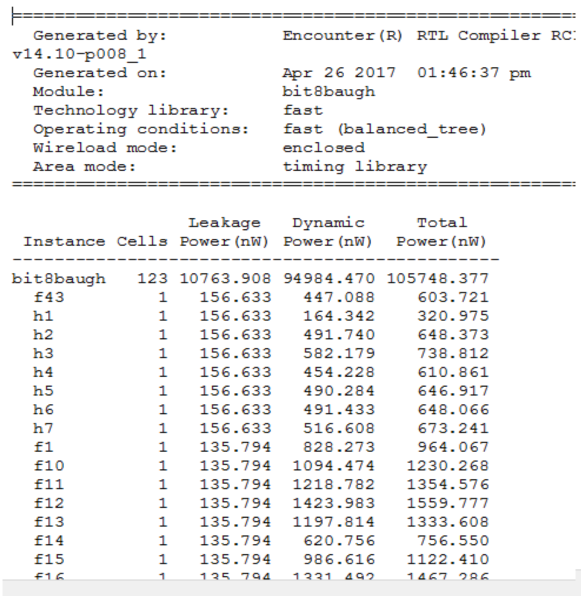
```
====================================================================
  Generated by:            Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:            Apr 26 2017  01:51:00 pm
  Module:                  bit8baugh
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
====================================================================

                    Leakage   Dynamic    Total
 Instance  Cells Power(nW) Power(nW) Power(nW)
----------------------------------------------
bit8baugh   121    15.562 26969.291 26984.853
  f21         1     0.514   333.686   334.200
  f43         1     0.181   173.139   173.320
  h1          1     0.181    64.909    65.090
  h2          1     0.181   184.852   185.034
  h3          1     0.181   217.820   218.001
  h4          1     0.181   169.669   169.850
  h5          1     0.181   183.695   183.876
  h6          1     0.181   184.611   184.793
  h7          1     0.181   196.463   196.645
  f1          1     0.168   183.594   183.761
  f10         1     0.168   282.445   282.613
  f11         1     0.168   313.758   313.926
  f12         1     0.168   366.551   366.719
  f13         1     0.168   312.789   312.957
  f14         1     0.168   169.907   170.075
  f15         1     0.168   221.538   221.706
```

Power synthesis report for 8bit Baugh wooley multiplier using 45nm

D. Using 90nm Technology

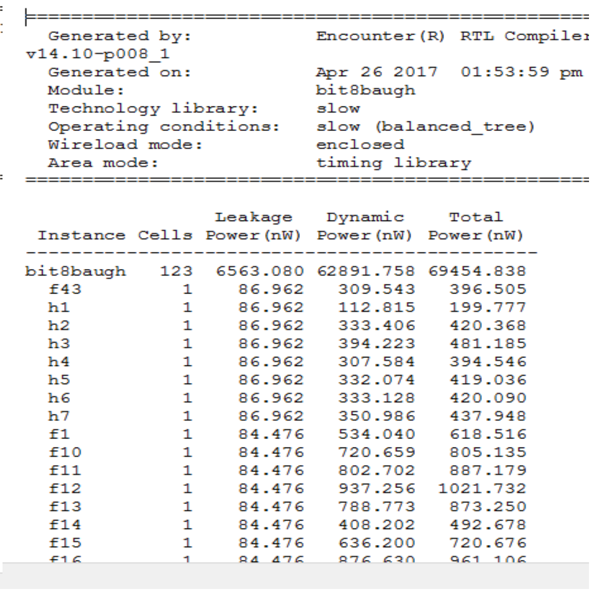
Fast.lib



```
=======================================================
  Generated by:            Encounter(R) RTL Compiler RC
v14.10-p008_1
  Generated on:            Apr 26 2017  01:46:37 pm
  Module:                  bit8baugh
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=======================================================

                   Leakage   Dynamic     Total
 Instance Cells Power(nW) Power(nW)  Power(nW)
-----------------------------------------------
bit8baugh   123 10763.908 94984.470 105748.377
  f43         1   156.633   447.088    603.721
  h1          1   156.633   164.342    320.975
  h2          1   156.633   491.740    648.373
  h3          1   156.633   582.179    738.812
  h4          1   156.633   454.228    610.861
  h5          1   156.633   490.284    646.917
  h6          1   156.633   491.433    648.066
  h7          1   156.633   516.608    673.241
  f1          1   135.794   828.273    964.067
  f10         1   135.794  1094.474   1230.268
  f11         1   135.794  1218.782   1354.576
  f12         1   135.794  1423.983   1559.777
  f13         1   135.794  1197.814   1333.608
  f14         1   135.794   620.756    756.550
  f15         1   135.794   986.616   1122.410
  f16         1   135.794  1331.492   1467.286
```

Slow.lib



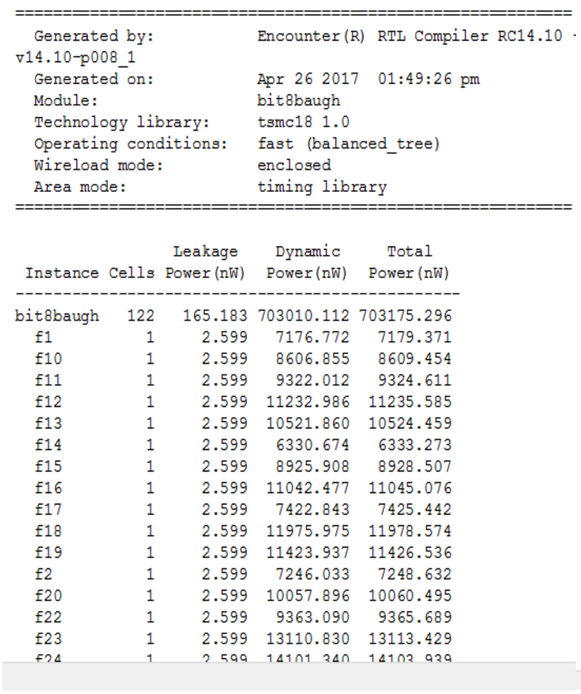
```
=======================================================
  Generated by:            Encounter(R) RTL Compiler
v14.10-p008_1
  Generated on:            Apr 26 2017  01:53:59 pm
  Module:                  bit8baugh
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=======================================================

                   Leakage   Dynamic     Total
 Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
bit8baugh   123  6563.080 62891.758 69454.838
  f43         1    86.962   309.543   396.505
  h1          1    86.962   112.815   199.777
  h2          1    86.962   333.406   420.368
  h3          1    86.962   394.223   481.185
  h4          1    86.962   307.584   394.546
  h5          1    86.962   332.074   419.036
  h6          1    86.962   333.128   420.090
  h7          1    86.962   350.986   437.948
  f1          1    84.476   534.040   618.516
  f10         1    84.476   720.659   805.135
  f11         1    84.476   802.702   887.179
  f12         1    84.476   937.256  1021.732
  f13         1    84.476   788.773   873.250
  f14         1    84.476   408.202   492.678
  f15         1    84.476   636.200   720.676
  f16         1    84.476   876.630   961.106
```

Power synthesis report for 8bit Baugh wooley multiplier using 90nm

D. Using 180nm Technology:

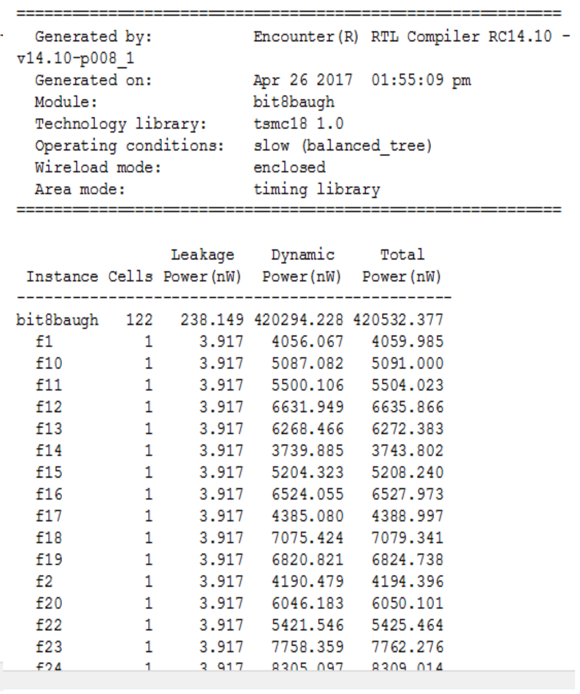
Fast.lib:



```
=======================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  01:49:26 pm
  Module:                  bit8baugh
  Technology library:      tsmc18 1.0
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=======================================================

                   Leakage   Dynamic     Total
 Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
bit8baugh   122   165.183 703010.112 703175.296
  f1          1     2.599   7176.772   7179.371
  f10         1     2.599   8606.855   8609.454
  f11         1     2.599   9322.012   9324.611
  f12         1     2.599  11232.986  11235.585
  f13         1     2.599  10521.860  10524.459
  f14         1     2.599   6330.674   6333.273
  f15         1     2.599   8925.908   8928.507
  f16         1     2.599  11042.477  11045.076
  f17         1     2.599   7422.843   7425.442
  f18         1     2.599  11975.975  11978.574
  f19         1     2.599  11423.937  11426.536
  f2          1     2.599   7246.033   7248.632
  f20         1     2.599  10057.896  10060.495
  f22         1     2.599   9363.090   9365.689
  f23         1     2.599  13110.830  13113.429
  f24         1     2.599  14101.340  14103.939
```

Slow.lib:



```
=======================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  01:55:09 pm
  Module:                  bit8baugh
  Technology library:      tsmc18 1.0
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
=======================================================

                   Leakage   Dynamic     Total
 Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
bit8baugh   122   238.149 420294.228 420532.377
  f1          1     3.917   4056.067   4059.985
  f10         1     3.917   5087.082   5091.000
  f11         1     3.917   5500.106   5504.023
  f12         1     3.917   6631.949   6635.866
  f13         1     3.917   6268.466   6272.383
  f14         1     3.917   3739.885   3743.802
  f15         1     3.917   5204.323   5208.240
  f16         1     3.917   6524.055   6527.973
  f17         1     3.917   4385.080   4388.997
  f18         1     3.917   7075.424   7079.341
  f19         1     3.917   6820.821   6824.738
  f2          1     3.917   4190.479   4194.396
  f20         1     3.917   6046.183   6050.101
  f22         1     3.917   5421.546   5425.464
  f23         1     3.917   7758.359   7762.276
  f24         1     3.917   8305.097   8309.014
```

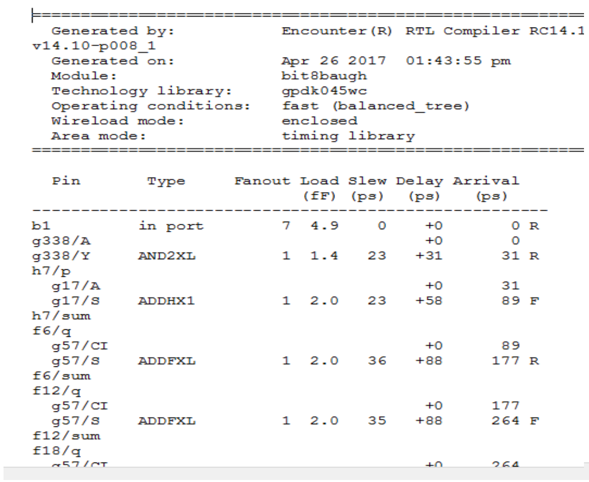
Power synthesis report for 8bit Baugh wooley multiplier using 180nm

6.3.6 Delay and Timing Synthesis Results:

Same as in case of power synthesis, here we are using here we are using fast.lib and slow.lib at supply voltage = 1.8 V for 180nm technology, supply voltage = 1.1 V for 90nm technology and supply voltage = 1.3 V for 45nm technology. As to these supply voltages the results for fast.lib and slow.lib are shown below:

D. Using 45nm Technology

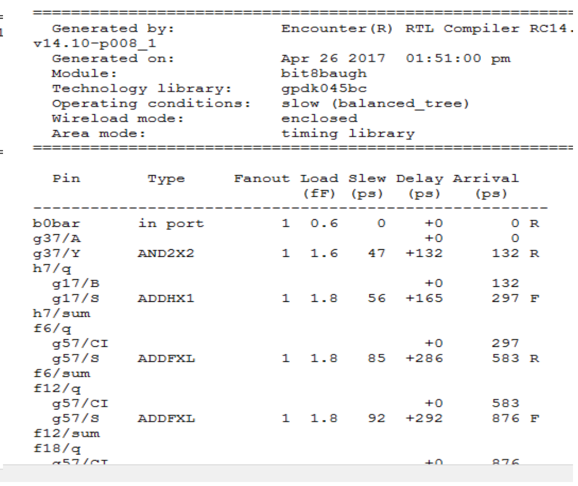
Fast.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.1
v14.10-p008_1
  Generated on:            Apr 26 2017  01:43:55 pm
  Module:                  bit8baugh
  Technology library:      gpdk045wc
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b1          in port        7  4.9    0    +0       0 R
g338/A                                    +0       0
g338/Y      AND2XL         1  1.4   23   +31      31 R
h7/p
  g17/A                                   +0      31
  g17/S     ADDHX1         1  2.0   23   +58      89 F
h7/sum
f6/q
  g57/CI                                  +0      89
  g57/S     ADDFXL         1  2.0   36   +88     177 R
f6/sum
f12/q
  g57/CI                                  +0     177
  g57/S     ADDFXL         1  2.0   35   +88     264 F
f12/sum
f18/q
  g57/CI                                  +0     264
```

Slow.lib



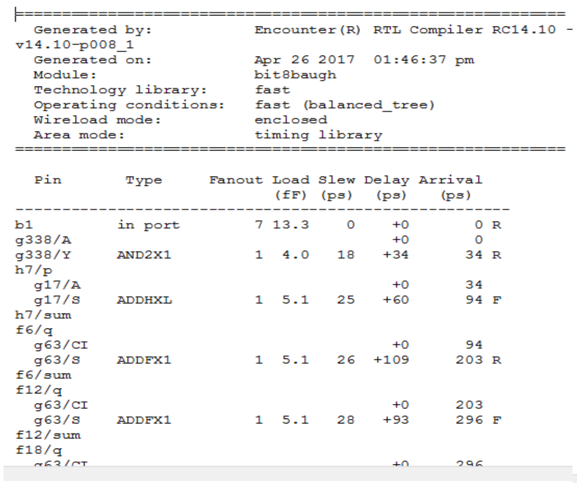
```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.
v14.10-p008_1
  Generated on:            Apr 26 2017  01:51:00 pm
  Module:                  bit8baugh
  Technology library:      gpdk045bc
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b0bar       in port        1  0.6    0    +0       0 R
g37/A                                     +0       0
g37/Y       AND2X2         1  1.6   47  +132     132 R
h7/q
  g17/B                                   +0     132
  g17/S     ADDHX1         1  1.8   56  +165     297 F
h7/sum
f6/q
  g57/CI                                  +0     297
  g57/S     ADDFXL         1  1.8   85  +286     583 R
f6/sum
f12/q
  g57/CI                                  +0     583
  g57/S     ADDFXL         1  1.8   92  +292     876 F
f12/sum
f18/q
  g57/CI                                  +0     876
```

Timing Synthesis Report for 8bit Baugh wooley multiplier using 45nm

D. Using 90nm Technology

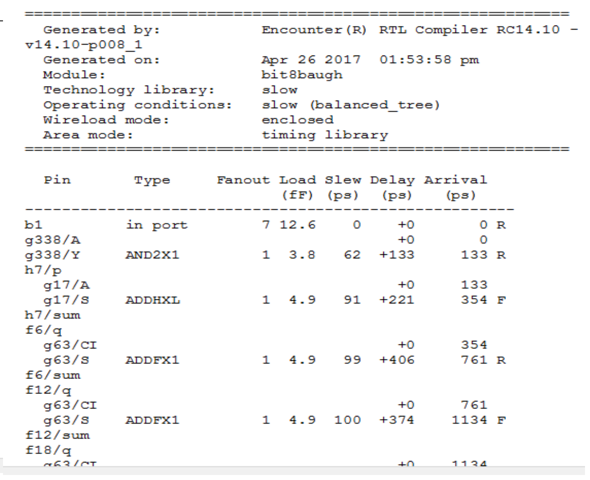
Fast.lib



```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  01:46:37 pm
  Module:                  bit8baugh
  Technology library:      fast
  Operating conditions:    fast (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b1          in port        7 13.3    0    +0       0 R
g338/A                                    +0       0
g338/Y      AND2X1         1  4.0   18   +34      34 R
h7/p
  g17/A                                   +0      34
  g17/S     ADDHXL         1  5.1   25   +60      94 F
h7/sum
f6/q
  g63/CI                                  +0      94
  g63/S     ADDFX1         1  5.1   26  +109     203 R
f6/sum
f12/q
  g63/CI                                  +0     203
  g63/S     ADDFX1         1  5.1   28   +93     296 F
f12/sum
f18/q
  g63/CI                                  +0     296
```

Slow.lib



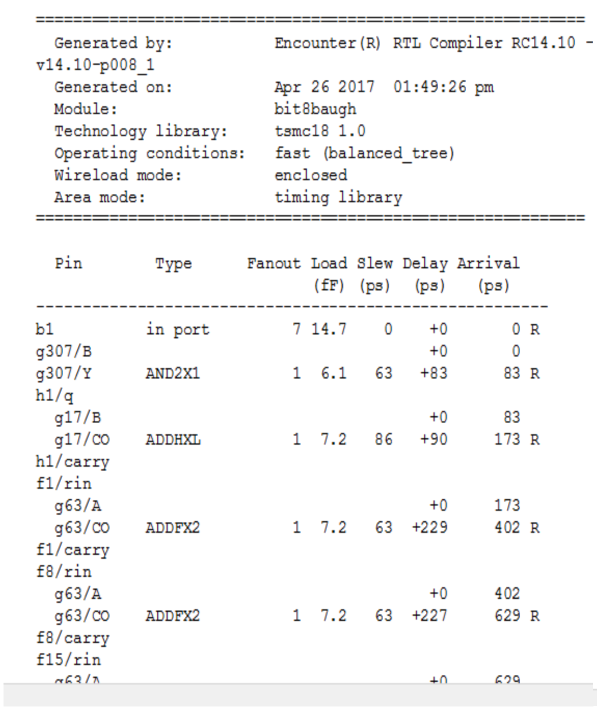
```
============================================================
  Generated by:            Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:            Apr 26 2017  01:53:58 pm
  Module:                  bit8baugh
  Technology library:      slow
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
------------------------------------------------------
b1          in port        7 12.6    0    +0       0 R
g338/A                                    +0       0
g338/Y      AND2X1         1  3.8   62  +133     133 R
h7/p
  g17/A                                   +0     133
  g17/S     ADDHXL         1  4.9   91  +221     354 F
h7/sum
f6/q
  g63/CI                                  +0     354
  g63/S     ADDFX1         1  4.9   99  +406     761 R
f6/sum
f12/q
  g63/CI                                  +0     761
  g63/S     ADDFX1         1  4.9  100  +374    1134 F
f12/sum
f18/q
  g63/CI                                  +0    1134
```

Timing Synthesis Report for 8bit Baugh wooley multiplier using 90nm

D. Using 180nm Technology:

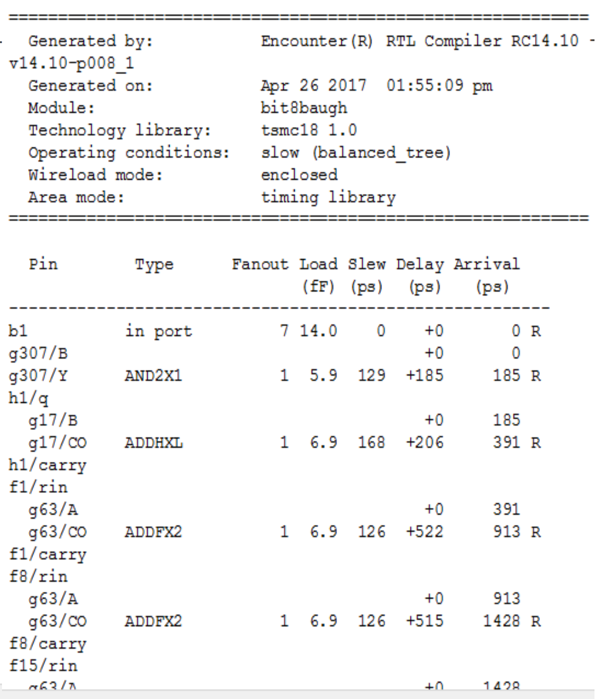
Fast.lib



```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:          Apr 26 2017  01:49:26 pm
  Module:                bit8baugh
  Technology library:    tsmc18 1.0
  Operating conditions:  fast (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
---------------------------------------------------------
b1           in port       7 14.7    0    +0       0 R
g307/B                                    +0       0
g307/Y       AND2X1        1  6.1   63   +83      83 R
h1/q
  g17/B                                   +0      83
  g17/CO     ADDHXL        1  7.2   86   +90     173 R
h1/carry
f1/rin
  g63/A                                   +0     173
  g63/CO     ADDFX2        1  7.2   63  +229     402 R
f1/carry
f8/rin
  g63/A                                   +0     402
  g63/CO     ADDFX2        1  7.2   63  +227     629 R
f8/carry
f15/rin
  g63/A                                   +0     629
```

Slow.lib



```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.10 -
v14.10-p008_1
  Generated on:          Apr 26 2017  01:55:09 pm
  Module:                bit8baugh
  Technology library:    tsmc18 1.0
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

  Pin        Type      Fanout Load Slew Delay Arrival
                              (fF) (ps)  (ps)   (ps)
---------------------------------------------------------
b1           in port       7 14.0    0    +0       0 R
g307/B                                    +0       0
g307/Y       AND2X1        1  5.9  129  +185     185 R
h1/q
  g17/B                                   +0     185
  g17/CO     ADDHXL        1  6.9  168  +206     391 R
h1/carry
f1/rin
  g63/A                                   +0     391
  g63/CO     ADDFX2        1  6.9  126  +522     913 R
f1/carry
f8/rin
  g63/A                                   +0     913
  g63/CO     ADDFX2        1  6.9  126  +515    1428 R
f8/carry
f15/rin
  g63/A                                   +0    1428
```

Timing Synthesis Report for 8bit Baugh wooley multiplier using 180nm

6.4. Performance Evaluation for 4x4 multipliers:-

Here we are comparing the synthesis of the taken multiplier, 4bit Vedic, Wallace tree multiplier and baugh wooley multipliers in terms of area and power and time delay.

Comparative analysis of fast multipliers for 4bit

Fast multipliers	Technology used	Type	Cells	Total Power (nW)	Total Delay(ps)
Vedic multiplier	45nm	Fast	31	6221.449	512
		Slow	31	4314.216	1720
	90nm	Fast	31	16132.59	553
		Slow	31	10800.55	2164
	180nm	Fast	31	87615.51	1094
		Slow	31	53358.82	2658
Wallace tree multiplier	45nm	Fast	31	6076.655	466
		Slow	31	4215.107	1579
	90nm	Fast	31	16127.1	509
		Slow	31	10591.51	1902
	180nm	Fast	31	86606.6	1096
		Slow	31	52815.12	2702
Baugh Wooley multiplier	45nm	Fast	31	7587.767	590
		Slow	31	512.534	1947
	90nm	Fast	33	18741.5	638
		Slow	33	12355.87	2396
	180nm	Fast	32	112274	1116
		Slow	32	67467.99	2818

Table6.1. Comparison between the parameters of 4-bit Vedic, Wallace, Baugh wooley multiplier

6.5. Performance Evaluation for 8x8 multipliers:-

Here we are comparing the synthesis of the taken multiplier, 8bit Vedic, Wallace tree multiplier and baugh wooley multipliers in terms of area and power and time delay.

Comparative analysis of Fast multipliers 8 bit

Fast multipliers	Technology used	Type	Cells	Total Power (nW)	Total Delay(ps)
Vedic multiplier	45nm	Fast	143	43023.68	1036
		Slow	142	28914.86	3457
	90nm	Fast	145	109978.4	1122
		Slow	145	71957.79	4194
	180nm	Fast	143	666989.5	2419
		Slow	143	398013.8	5813
Wallace tree multiplier	45nm	Fast	115	35183.65	1471
		Slow	115	23601.53	4876
	90nm	Fast	115	92618.17	1574
		Slow	115	60868.62	5843
	180nm	Fast	115	616756.3	3068
		Slow	115	370575.8	7690
Baugh Wooley multiplier	45nm	Fast	121	40275.58	1214
		Slow	121	26984.85	3944
	90nm	Fast	123	102748.4	1297
		Slow	123	69454.84	4854
	180nm	Fast	122	703175.3	2675
		Slow	122	420532.4	6494

Table6.2. Comparison between the parameters of 8-bit Vedic, Wallace, Baugh wooley multiplier

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Vedic, Wallace tree, Baugh wooley multipliers has been implemented using Verilog in cadence. For simulation, cadence NCsim and for synthesis Encounter RTL compiler has been used. As a result, it has been concluded that 4-bit and 8-bit multipliers gives result in all three Technologies 180nm, 90nm, and 45nm. As per analysis of three multipliers conclusion can be drawn and suggestion can be made with reference to area, power and speed. By adopting these types of like Vedic, Wallace, Baugh-wooley multiplier accurate performance can be achieved when compared to the existing techniques. In terms power dissipation Wallace tree multiplier is better, in terms of time delay and area Vedic multiplier is better. And combination of these all (power, time delay, and area) Baughwooley is better comparing with Vedic and Wallace tree multiplier. Multipliers have proven effective in DSP, other applications, to increase speed of the DSP, Microprocessor chips and digital communication.

In future, we will further compare with these Vedic, Wallace tree, Baugh wooley multipliers with other multipliers for different parameters such as area, power and delay as well as total number of cells used to give the reduced number of output bit steam as compared to these three multiplies. So these three multipliers which will be more efficient in terms of area, power And delay should give a reduced bit stream outputs as compared to these three multiplies will be the efficient multipliers Techniques.

Depending upon the parametric analysis these fast multipliers have utilized in MAC unit in DSP application

REFERENCES

[1] R.Raju, S.Veerakumar, "Design and Implementation of Low Power and High Performance Vedic Multiplier" *International Conference on Communication and Signal Processing,978-1-5090-0396-9/16/$31.00 © April 6-8, 2016 IEEE.*

[2] Rakesh Kumar, Pradeep Kumar, "An Efficient Baugh-Wooley Multiplication Algorithm for 32-bit Synchronous Multiplication" *International Journal of Advanced Engineering Research and Science (IJAERS) [Vol-1, Issue-2, July 2014] ISSN: 2349-6495*

[3] Kokila Bharti Jaiswal, Nithish Kumar V, Pavithra Seshadri, "Low Power Wallace Tree Multiplier Using Modified Full Adder" *3rd International Conference on Signal Processing, Communication and Networking (ICSCN),978-1-4673-6823-0/15/$31c.2015 IEEE.*

[4] Indrayani Patle, Akansha Bhargav, Prashant Wanjari, "Implementation of Baugh-Wooley Multiplier Based on Soft-Core Processor" *IOSR Journal of Engineering (IOSRJEN) e-ISSN: 2250-3021, p-ISSN: 2278-8719 Vol. 3, Issue 10 (October. 2013), ||V3|| PP 01-07*

[5] Ms. G. R. Gokhale, Mr. S. R. Gokhale, "Design of Area and Delay Efficient Vedic Multiplier Using Carry Select Adder" *International Conference on Information Processing (ICIP) Vishwakarma Institute of Technology. 978-1-4673-7758-4/15/$31.00 © Dec 16-19, 2015 IEEE*

[6] Soniya, Suresh Kumar, "A Review of Different Type of Multipliers and Multiplier-Accumulator Unit", *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 4, July – August 2013.*

[7] Abhishek Mukherjee, Abhijit Asati, "Generic Modified Baugh Wooley Multiplier" *International Conference on Circuits, Power and Computing Technologies [ICCPCT-2013] 978-1-4673-4922-2/13/$31.00 ©2013 IEEE*

[8] Sumit Vaidya, Deepak Dandekar, "Delay-Power Performance Comparison of Multipliers in Vlsi Circuit Design" *International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, July 2010*

[9] Pramodini Mohanty, "An Efficient Baugh-Wooley Architecture for Signed & Unsigned Fast Multiplication*" NIET Journal of Engineering & Technology, Vol. 1, Issue 2, 2013*

[10] Gaurav Sharma, Arjun Singh Chauhan, Himanshu Joshi, Satish Kumar Alaria,"Delay Comparison of 4 by 4 Vedic Multiplier based on Different Adder Architectures using VHDL" *International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 3 Issue 3, March - 2014*

[11] Taye Girma, (2013) "Designing and Synthesizing a Wallace Tree Multiplier for High Speed Performance" *International Journal of Artificial Intelligence and Mechatronics Volume 2, Issue 3, ISSN 2320 – 5121*

[12] Amrita Nanda, Shreetam Behera, (2014) “Design and Implementation of UrdhvaTiryakbhyam Based Fast 8×8 Vedic Binary Multiplier” *International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181 Vol. 3 Issue 3, March – 2014*

[13] M Pradhan, R Panda, S K Sahu, " MAC Implementation using Vedic Multiplication Algorithm," *International Journal of Computer Applications (0975 – 8887), Vol- 21, No.7, May 2011*

[14] Premananda B.S, Samarth S. Pai, Shashank B, Shashank S.Bhat, “ Design And Implementation of 8-bit Vedic Multiplier*”, IJAREEIE,Vol.2,Issue 12,ISSN:2320-3765,Dec-2013.*

[15] Taye Girma, “Designing and Synthesizing a Wallace Tree Multiplier for High Speed Performance”, *International Journal of Artificial Intelligence and Mechatronics Volume 2, Issue 3, ISSN 2320 – 5121,2013.*

[16] Pramod S. Aswale, Mukesh P. Mahajan, Manjul V. Nikumbh, Omkar S. Vaidya, (2015), “Implementation of Baugh-Wooely Multiplier and Modified Baugh Wooely Multiplier Using Cadence (Encounter) RTL” *International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 2, February 2015 ,293 ISSN: 2278 – 7798*