DESIGN AND ANALYSIS OF DIFFERENT ADDERS USING REVERSIBLE LOGIC GATES

A PROJECT REPORT

Submitted by

E NIKITHA	**21781A0459**
G HEMA SUNDAR REDDDY	**21781A0463**
G VINAY	**21781A0466**
I VEERA MANOHAR REDDY	**21781A0481**
J MAHESWARA REDDY	**21781A0482**

in partial fulfillment for the Award of the degree of

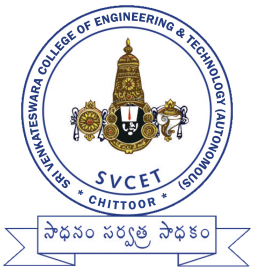
BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Under the guidance of
Mr. S. NAGARAJ, M. Tech, MBA, (Ph.D)
Associate Professor
At

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY(AUTONOMOUS)
R. V. S. NAGAR, CHITTOOR - 517127. (A.P).
(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapuramu)
(Accredited by NBA, NEW Delhi AND NAAC, Bengaluru)
(An ISO 9001:2000 Certified Institution)

APRIL 2025

SRI VENKATESWARA COLLEGE OF ENGINEERING TECHNOLOGY(AUTONOMOUS)
R.V.S. NAGAR, CHITTOOR - 517127. (A.P).
(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapuramu)
(Accredited by NBA, NEW Delhi AND NAAC, Bengaluru)
(An ISO 9001:2000 Certified Institution)



CERTIFICATE

This is to certify that the project entitled **DESIGN AND ANALYSIS OF DIFFERENT ADDERS USING REVERSIBLE LOGIC GATES** is the bonafied work carried out by **"E NIKITHA(21781A0459),G HEMA SUNDAR REDDY(21781A0463), G VINAY(21781A0466),I VEERA MANOHAR REDDY(21781A0481),J MAHESWARA REDDY(21781 A0482)"** students of B.Tech,ECE,SVCET,during the academic year 2021-2025 in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in ELECTRONICS AND COMMUNICATION ENGINEERING.

SIGNATURE OF THE GUIDE **SIGNATURE OF THE HOD**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Viva Voce Conducted on:

ACKNOWLEDGEMENTS

A Grateful thanks to **Dr.R.Venkataswamy**,Chairman,Sri Venkateswara College of Engineering and Technology for providing education in their esteemed institution.

We, wish to record our deep sense of gratitude and profound thanks to our beloved Vice Chairman, **Sri. R.V. Srinivas** for his valuable support throughout the course.

We, express our sincere thanks to **Dr. M. Mohan Babu**, our beloved principal for his encouragement and suggestions during the course of study.

We, wish to convey our gratitude and express our sincere thanks to **Dr.D.Srihari**,M.Tech,Ph.D, Head of the Department, Electronics and Communication Engineering, for giving us his inspiring guidance in undertaking our project report.

We, express our sincere thanks to the Project Guide **Mr. S. Nagaraj** ,M.Tech, MBA, (Ph.D) Associate Professor, Department of Electronics and Communication Engineering, for his keen interest, stimulating guidance, constant encouragement with our work during all stages, to bring this project into fruition.

We, wish to convey our gratitude and express our sincere thanks to all Project Review Committee members for their support and cooperation rendered for successful submission of our project work.

Finally, we would like to express our sincere thanks to all teaching, non-teaching faculty members, our parents, and friends and for all those who have supported us to complete the project work successfully.

E NIKITHA	**21781A0459**
G HEMA SUNDAR REDDDY	**21781A0463**
G VINAY	**21781A0466**
I VEERA MANOHAR REDDY	**21781A0481**
J MAHESWARA REDDY	**21781A0482**

ABSTRACT

The design and analysis of efficient adders are crucial for enhancing the performance of digital systems, particularly in arithmetic and signal processing applications. This paper presents the comparison of five different types of adders Ripple Carry Adder (RCA), Carry Increment Adder (CIA), Carry Skip Adder(CSKA),Carry Select Adder (CSLA), and Carry Save Adder (CSA) using Basic logic gates and Reversible logic gates. Reversible logic is an emerging computational model that offers reduced power dissipation, which is a significant advantage in low-power digital circuit design. Each adder type is analyzed in terms of its logical structure, delay, area, and power consumption, with the aim of identifying how reversible gates can optimize these parameters. The implementation of basic reversible gates such as the Toffoli gate,Peres gate and Feynman gate in constructing the adders and compares their performance with traditional adders with basic logic gates. Through simulation and performance metrics, this paper highlights the potential of reversible logic in reducing energy consumption and enhancing speed in arithmetic operations. The results suggest that reversible logic can provide a promising avenue for the development of energy efficient and high-performance adders in future digital systems.

Index Terms—Ripple Carry Adder, Carry Save Adder, Carry Select Adder, Carry Skip Adder, Carry Increment Adder.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

The design and analysis of adders using reversible logic gates has gained significant attention in recent years due to the growing demand for energy Efficient computting systems. The traditional adder designs Like Ripple carry Adder(RCA), Carry increment-Adder (CIA), Carry Skip Adder (CSKA), Carry Select Adder(CSLA) and Carry Save Adder (CSA) are Evaluated and analyzed to determine their suitability for implementation using reversible logic gates. We aim to explore the potential of reversible computing for reducing power consumption and heat dissipation in modern digital system.

This project focuses on the design and analysis of different adder circuits using reversible logic gates. Adders are fundamental building blocks in digital systems, widely used in arithmetic and logic units (ALUs), signal processing, and data path architectures. By implementing various types of adders such as half adders, full adders, ripple carry adders, and carry look-ahead adders using reversible logic, we aim to evaluate their performance in terms of gate count, quantum cost, garbage outputs, and delay.

The objective of this project is to compare traditional logic implementations with reversible counterparts and highlight the advantages of reversible logic in achieving low-power, high-efficiency computing systems. This work lays the groundwork for developing future computing devices, especially in quantum computing and nanotechnology, where energy efficiency and information preservation are critical.

Chapter 2

LITERATURE SURVEY

2.1 Literature Review

a. **S. Nagaraj; G.M. Sreerama Reddy and S. Aruna Mastani "Analysis of different Adders using CMOS, CPL and DPL logic",Publish ed in: 2017 14th IEEE India Council International Conference (INDICON).**
In this paper we design and analyse different types of adders using CMOS, Complementary Pass Transistor Logic(CPL), Double Pass Tra nsistor Logic(DPL) logics. Ripple Carry Adder, Carry Look Ahead Adder, Carry Save Adder, Carry Incremental Adder, Carry Skip Adder, Carry Select Adder, Conditional Sum Adder are designed using CMOS, Complementary Pass Transistor Logic(CPL), Double pass transistor logic(DPL) logics for 16-bit, 32-bit and their speed, area and power are compared.

b. **M. Bala Murugesh; S. Nagaraj; J. Jayasree; G. Vijay Kumar Reddy,"Modified High Speed 32-bit Vedic Multiplier Design and Implementation",Published in: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)**
The proposed research work specifies the modified version of binary vedic multiplier using vedic sutras of ancient vedic mathematics. It provides modification in prelimi-narilry implemented vedic multiplier. The modified binary vedic multiplier is preferable has shown improvement in the terms of the time delay and also device utilization. The proposed technique was designed and implemented in Verilog HDL. For HDL simulation, modelsim tool is used and for circuit synthesis, Xilinx is used. The simulation has been done for 4 bit, 8 bit, 16 bit, 32 bit multiplication operation. Only for 32 bit binary vedic multiplier technique the simulation results are shown. This modified multipli-

cation technique is extended for larger sizes. The outcomes of this multiplication technique is compared with existing vedic multiplier techniques.

c. **S Nagaraj; K Srinivasul Reddy; K Deekshith Varma; K Rupa Sri; K Likhitha; K Vamsi,"Performance Analysis of Vedic Multiplier using Different Adders",Published in: 2024 5th International Conference on Smart Electronics and Communication (ICO SEC)**

In this paper, we present a comprehensive analysis of the performance of a Vedic multiplier utilizing various types of adders. The Vedic multiplier, inspired by ancient Indian mathematics, offers potential advantages in terms of speed and area efficiency compared to conventional multipliers. We explore the impact of different types of adders, such as ripple-carry adder, carry-look-ahead adder, carry-increament adder, carryskip adder and, carry-select adder on the overall performance of the Vedic multiplier. Our analysis involves simulation studies using modelsim and synthesis using xlinix 14.7 to evaluate critical metrics such as pin-to-pin delay, power consumption, total delay and area utilization. Through detailed experimentation and analysis, we aim to provide insights into the optimal configuration of adders for achieving enhanced performance in Vedic multiplier implementations. This research contributes to the understanding of efficient arithmetic unit design methodologies, with potential applications in high-speed digital signal processing, scientific computing, and other computation-intensive domains.

d. **Pakkiraiah .C, Dr. R.V.S. Satyanarayana Research Scholar, Professor, Department of ECE, SVU College of Engineering, S.V. University, Tirupati. "Design and FPGA Realization of Energy Efficient Reversible Full Adder for Digital Computing Application",25 September ,2022**

This research focuses on the reversible digital full adder circuit, which is a key element in establishing the Energy Delay Product (EDP) for various computer applications. Here, a new reversible binary full adder is designed using the switching activity concept and the logic decomposition method. The internal blocks for reversible full adders such as Feynman Gate, Toffoli Gate, and New Gate are designed first, then a new reversible binary full adder is developed using the proposed method. In this paper, conventional and proposed reversible

full adders are synthesized using the Xilinx Vivado design suite for the Zynq-7000 family of device configuration. According to the implementation results, the proposed reversible full adder circuit consumes less dynamic power dissipation than the existing method in comparison. Furthermore, a formulae-based evaluation is conducted on the implementation results to estimate the EDP of the design. The proposed reversible full adder design can achieve a 32.3% EDP improvement compared to the Proposed Full Adder Keywords: Dynamic Power, EDP, Feynman Gate, New Gate, Toffoli Gate.

e. **"Design of logic gates using reversible gates with reduced quantum cost", S.Saniya Samrin, Rachamma Patil, Sumangala Itagi, Smita C Chetti, Afiya Tasneem H. K. E. Society's S. L. N. College of Engineering, Raichur 584135, India, Global Transitions Proceedings, Volume 3, Issue 1, June 2022, Pages 136-141Available online 2 April 2022, Version of Record 9 June 2022.**
Reversible logic is also called information lossless logic, since the information embedded in the circuits can be recovered, if lost. Research carried out by Lindauer and Bennett proved that the energy dissipation would not occur if computation is made reversible. With this aim a number of reversible gates were designed and invented. As examples likethe Fredkin gate, the Toffoli gate, the Peres gate, and the Feynman gate. Reversible logic has extensive applications and is considered as one of the futuristic technologies. But the logic circuit designing is based on logic gates, which are non-reversible. This paper presents design of logic gates using reversible gates. These logic gates help in future implementation of higher end circuits. In this paper an attempt is made to design logic gates using reversible gates and some of the higher end circuits are also designed such as Binary-to-Grey, grey-to-Binary, Adder, Subtractor etc. this paper represents a brief review of the performance obtained with such chips. Reversible Logic circuits have promising applications in Quantum Computing, Low Power VLSI Design. This paper represents the implementation of Ripple Carry Adder (RCA) circuits using reversible logic gates are discussed.

f. **Gowthami P, RVS Satyanarayana, Research Scholar, Department of ECE, S V University College of Engineering, Tirupathi, AP, India. "Design Of Digital Adder Using Reversible Logic". February 2016.**

Reversible logic circuits have promising applications in Quantum computing, Low power VLSI design, the implementation of Ripple Carry Adder (RCA) circuits using reversible logic gates are discussed. Keywords - Reversible logic, Reversible logic circuits, ultra high speed, Power dissipation, Ripple Carry Adder.

g. **Neelam Soman, Chitrita Chaudhary, Sharad Yadav ,VLSI-design, VIT University, Vellore, India. "Reversible adder design for ripple carry and carry look ahead (4, 8, 16, 32-bit)" ,2016 International Conference on Computing, Communication and Automation (ICCCA), 29-30 April 2016**

The paper presents efficient adder circuits using Peres gate, New fault Tolerant gate and Double Feynman gate. The complexity, simulated outputs and the speed parameters for the adder circuit have been indicated using the Quartus II 9.1 edition and ModelSim tool. Moreover, the quantum algorithms can potentially solve NP-complete problems. Furthermore, doing high performance functions beyond the limit of deterministic computer systems is possible by only reversible logic. Quantum operations are unitary in nature which is reversible and hence the arithmetic operations like adders can be implemented using the reversible logic.

Chapter 3

REVERSIBLE LOGIC GATES

3.1 Introduction

Reversible logic gates are digital circuits where the input can be unique ly reconstructed from the output, meaning there is a one-to-one mapping between input and output vectors. Unlike conventional logic gates, which dissipate energy in the form of heat for every bit of information lost, reversible gates are designed to be energy-efficient by preserving information. The concept of reversible computing was first introduced by Rolf Landauer in 1961, who proposed that each bit of information loss results in a minimum amount of energy dissipation, known as Landauer's principle. Building on this, Charles H. Bennett in 1973 demonstrated that reversible computation could, in theory, be carried out without energy dissipation, provided no information is lost during processing. This groundbreaking idea laid the foundation for the development of reversible logic gates. Over time, various reversible gates such as the Feynman gate, Toffoli gate, Fredkin gate, and Peres gate have been designed, forming the backbone of reversible circuits. These gates are now extensively explored in low-power VLSI design, quantum computing, optical computing, and nanotechnology due to their potential to minimize power consumption and support backward computation.

The Toffoli gate, or the Controlled-Controlled-NOT (CCNOT) gate, is a 3x3 reversible gate. It takes three inputs and produces three outputs. The first two outputs remain unchanged, while the third output flips only if the first two inputs are both 1. This gate is universal for reversible computation, meaning any reversible logic circuit can be

built using Toffoli gates alone, making it fundamental in the design of complex reversible and quantum circuits.

Another important reversible gate is the Peres gate, which combines the functionalities of the Feynman and Toffoli gates into a single 3x3 gate. It is efficient in terms of quantum cost and gate count, which makes it suitable for implementing arithmetic and logical operations in reversible systems. The Peres gate generates outputs where the first is the same as the first input, the second is the XOR of the first two inputs, and the third is a conditional flip like in the Toffoli gate. Its optimized design has made it popular in designing low-power digital systems.

There are three fundamental Reversible logic gates
1.Toffoli Gate
2.Feynman Gate
3.Peres Gate

3.2 Toffoli Gate

The Toffoli gate, also known as the controlled-controlled-NOT (CC-NOT) gate, is a universal reversible logic gate used in quantum computing and classical reversible computing. It operates on three input bits: two control bits and one target bit. If both control bits are set to 1, the gate flips (negates) the target bit; otherwise, the target bit remains unchanged.

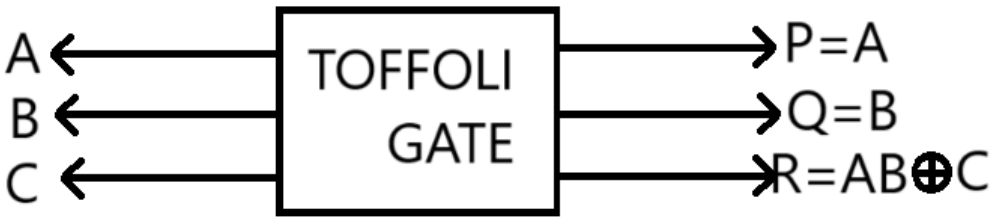


Figure 3.1: Toffoli Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Table 3.1: Toffoli Gate Truth table

The truth table of the Toffoli gate preserves input-output mapping, making it reversible. It is a key component in constructing quantum circuits for fault-tolerant computing and classical circuits with zero energy dissipation. The gate can be used to simulate any Boolean function and is essential for quantum error correction and entanglement operations. In quantum computing, the Toffoli gate is often decomposed into simpler gates like CNOT and single-qubit gates for practical implementation. Its universality stems from its ability to emulate other logic gates like AND, OR, and XOR.

3.3 Feynman Gate

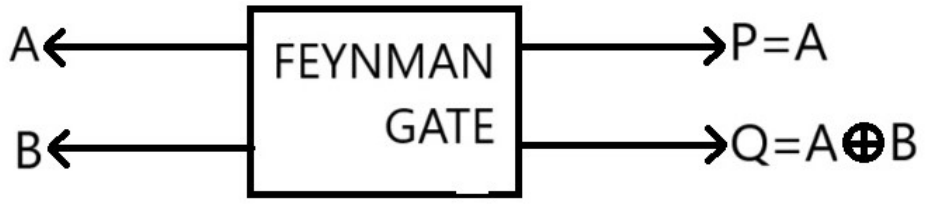


Figure 3.2: Feyman Gate

It makes use the Feynman gate is a fundamental building block in quantum computer. Some stock market swearing that two qubit gate

A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 3.2: Feyman Gate Truth Table

just is not true at all! The original input state can be recovered from the output state The gate is named after renowned physicist Richard Feynman, whose contributions to quantum t the first one; and so on along with you Browne

In the matrix representation, this means: [1 0] [1 1] should be:

- The first input qubit is copied to the first output qubit, while the second output qubit is the XOR of the two input qubits.
- The Feynman gate is a universal gate. That is any quantum circuit can be constructed using only Feynman gates It is also reversible because any computations performed with this gate will release less energy than classical counterparts non proprietary compiler efficiency
- The Feynman gate is a strong tool in quantum computing world. And it is expected to play an important role in the future quantum computer.

3.4 Peres Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Table 3.3: Peres Gate Truth table

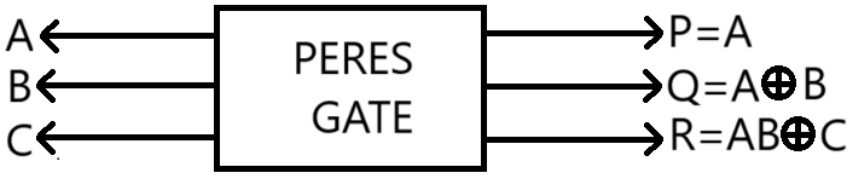


Figure 3.3: Peres Gate

The Peres Gate, also known as a Peres logic gate, is a quantum reversible logic gate widely used in quantum computing and reversible computing. It is a three-input, three-output gate combining the functions of the classical AND gate and XOR gate. The Peres Gate is efficient in terms of minimizing the number of garbage outputs and gate operations, making it suitable for reversible logic circuit design. Its inputs are typically labeled as and , and it produces three outputs: , (XOR operation), and . It is often used in applications like adders, multipliers, and error-correcting circuits in quantum systems. Its design ensures the preservation of information, which is essential in quantum computing to avoid energy loss. The Peres Gate is named after Asher Peres, a pioneer in quantum mechanics.

Chapter 4

ADDERS

4.1 Ripple Carry Adder

The Ripple Carry Adder is primarily used in digital electronics and computing systems for binary addition. It is commonly implemented in arithmetic logic units (ALUs), calculators, and digital signal processors. Due to its simple design, it is often used in educational tools for teaching digital logic. It is also found in embedded systems and small-scale processors where speed is less critical. Though slower than other adders, it is valued for its low hardware complexity and ease of implementation.

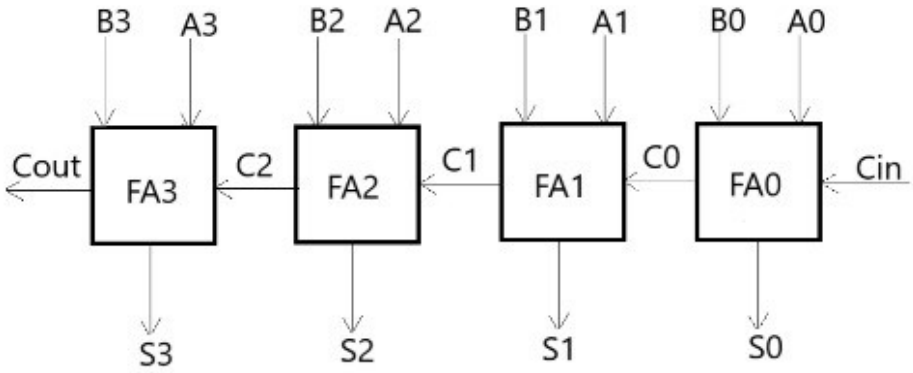


Figure 4.1: Ripple Carry Adder

The image shows a 4-bit Ripple Carry Adder, which adds two 4-bit binary numbers (A3-A0 and B3-B0) using four Full Adders (FA0 to FA3). The process begins with FA0, which takes the least significant bits A0 and B0, along with an input carry (Cin), and produces sum

bit S0 and carry C0. FA1 then adds A1, B1, and carry C0 to generate S1 and carry C1. This carry propagates sequentially through FA2 and FA3, each adding corresponding bits (A2-B2 and A3-B3) and the previous carry to produce sum outputs S2 and S3. The final carry out (Cout) is generated by FA3 and may indicate overflow. The term "ripple" comes from how each carry output must propagate through to the next adder, causing delays. This structure is simple and easy to implement but slower for large bit-width additions due to the carry propagation delay across all full adders.

4.2 Carry Increment Adder

The Carry Increment Adder is used in digital systems requiring faster arithmetic operations, such as in ALUs of processors, signal processing units, and embedded systems. It provides improved performance over ripple carry adders by reducing propagation delay, making it suitable for medium-speed applications. Its structure is efficient for hardware implementation with better speed-to-area ratio, ideal for systems balancing performance and complexity. Additionally, it is used in custom hardware designs where moderate speed enhancement is required without significantly increasing circuit complexity.

The image shows a Carry Increment Adder, which improves delay performance over the Ripple Carry Adder. The addition is divided into two 4-bit blocks. The lower 4-bit block (A[3:0] and B[3:0]) is added using a Ripple Carry Adder (RCA) with the actual input carry (Cin), producing outputs S0 to S3. The upper 4-bit block (A[7:4] and B[7:4]) is first added using a second RCA assuming Cin = 0, producing preliminary sums and a carry-out. If the carry-out from the first block is 1, the result of the second block must be incremented. This is done using a chain of Half Adders (HA) that increment the sum output bits S4 to S7. A final OR gate determines the final carry-out (Cout) based on whether the increment path was used. This design avoids recomputing the upper block for different carry-in values and reduces the overall delay compared to a standard ripple carry design.

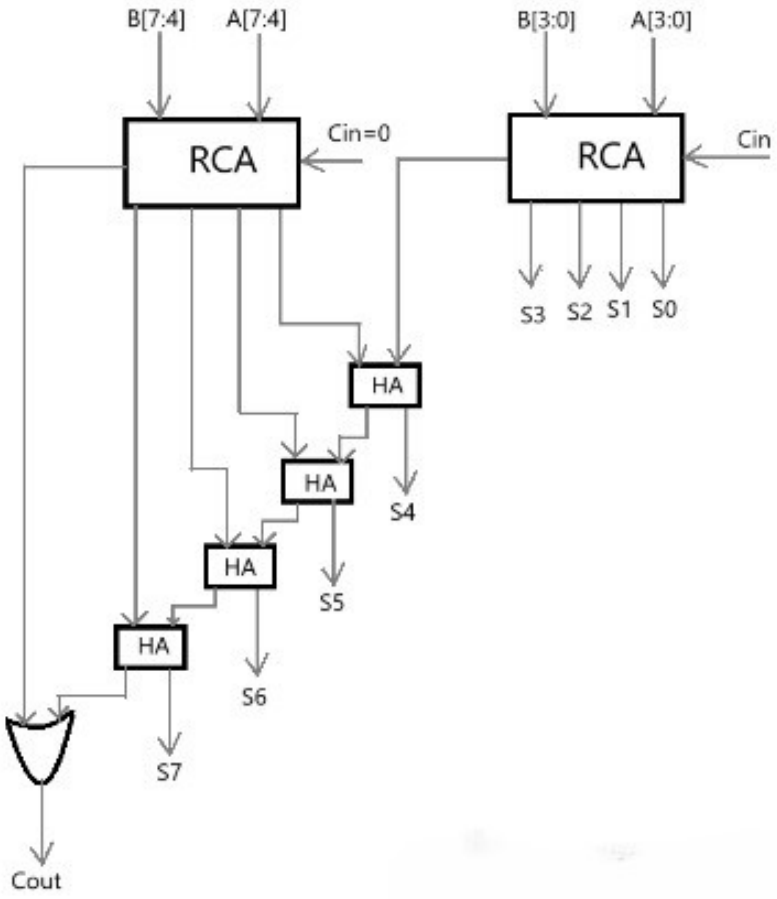


Figure 4.2: Carry Increment Adder

4.3 Carry Select Adder

The carry select adder is used in high-speed arithmetic circuits where fast addition is crucial, such as in digital signal processing, microprocessors, and ALUs (Arithmetic Logic Units). It significantly reduces delay compared to ripple carry adders by computing sum outputs in parallel. It is ideal for applications requiring quick computations and efficient performance. Due to its balanced trade-off between speed and area, it is often implemented in hardware design for processors and embedded systems to enhance execution speed.

The carry select adder in the image enhances speed by computing sum and carry outputs in parallel. It divides an 8-bit input into two 4-bit blocks. The least significant 4 bits (A3–A0 and B3–B0) are

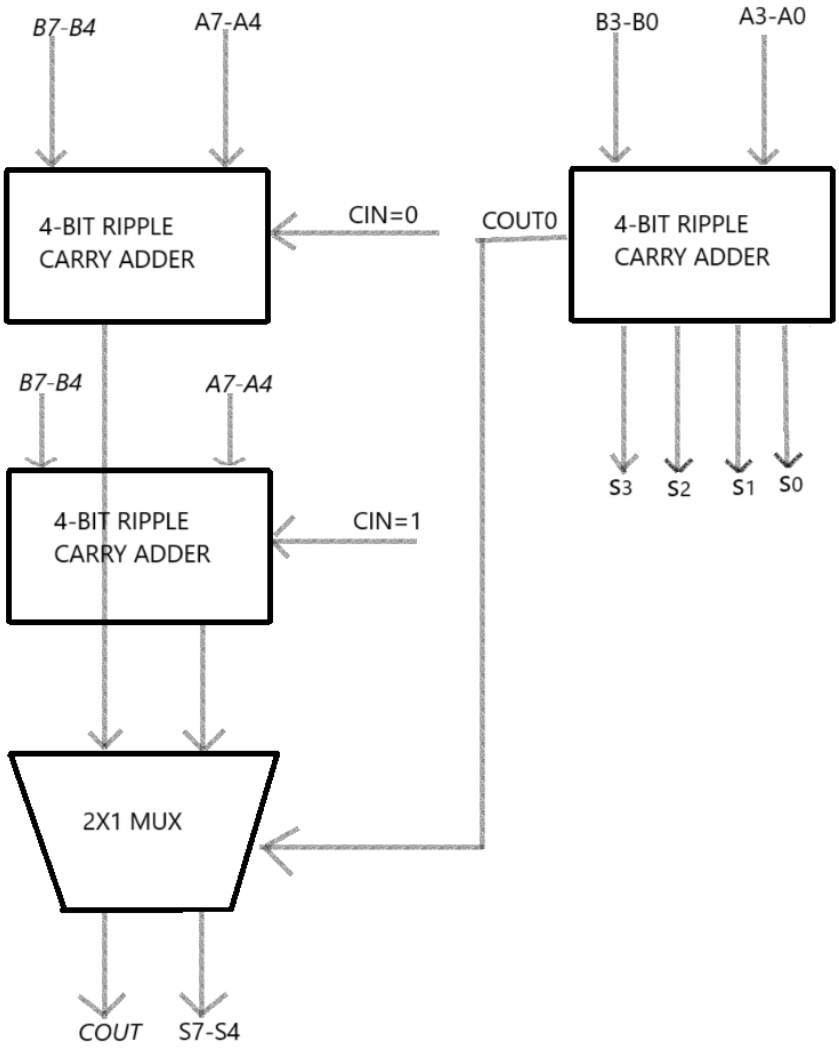


Figure 4.3: Carry Select Adder

added using a 4-bit ripple carry adder, producing sum bits S0–S3 and a carry-out (COUT0). The most significant 4-bit inputs (A7–A4 and B7–B4) are fed into two separate 4-bit ripple carry adders—one assumes carry-in (CIN) as 0 and the other as 1. These adders generate two potential outputs: one for CIN = 0 and the other for CIN = 1. A 2x1 multiplexer selects the correct sum (S4–S7) and carry-out based on the actual carry-out (COUT0) from the first stage. If COUT0 is 0, the mux selects the result from the adder with CIN = 0; if it is 1, it selects the output from CIN = 1 adder. This parallelism reduces delay, optimizing performance.

4.4 Carry Skip Adder

A Carry Skip Adder is used in digital circuits to speed up binary addition by reducing the delay caused by carry propagation. It is commonly used in arithmetic logic units (ALUs), digital signal processors (DSPs), and microprocessors where fast computation is essential. By allowing the carry to skip over blocks of full adders when all propagate conditions are true, it achieves better performance than ripple carry adders while maintaining simpler hardware than carry lookahead adders, making it suitable for medium-speed applications.

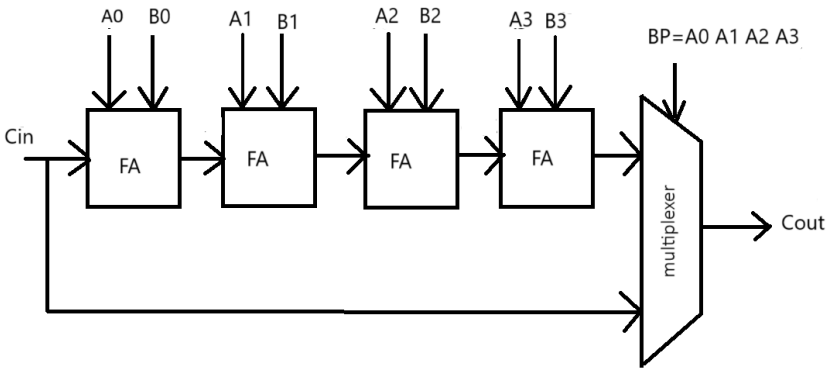


Figure 4.4: Carry Skip Adder

The image depicts a Carry Skip Adder, which enhances addition speed by reducing carry propagation delay. It consists of four Full

Adders (FAs) that compute sum and carry for each bit position of two 4-bit binary numbers (A and B) and an input carry (Cin).

Step-wise:

1) Each FA takes two input bits (e.g., A0, B0 for the first FA) and a carry-in to produce a sum and carry-out.

2) Carries ripple through each FA unless bypassed.

3) A block propagate signal (BP = A0·B0 · A1·B1 · A2·B2 · A3·B3) is generated true only if all bits will propagate a carry.

4) A multiplexer uses BP to select between the carry-out from the last FA (normal ripple) and the initial carry-in Cin (bypassed carry).

5) If BP is 1, the carry skips the block; otherwise, it ripples through. This mechanism speeds up addition by skipping carry propagation when possible.

4.5 Carry Save Adder

Carry Save Adders (CSAs) are widely used in high-speed arithmetic operations, especially in digital signal processing and computer architecture. They are essential in multiplication circuits, such as Wallace trees and Booth multipliers, where multiple binary numbers are added simultaneously. CSAs significantly reduce propagation delay by saving intermediate carry values instead of immediately propagating them. This makes them ideal for implementing fast and efficient adders in ALUs, floating-point units, and cryptographic hardware, where speed and parallelism are critical for performance.

This image shows the working of a Carry Save Adder (CSA) used for efficient multi-operand binary addition. The inputs A and B are two 4-bit binary numbers, and FA0 to FA7 are full adders. In the first stage, FA0–FA3 add the bits A0–A3 and B0–B3 with an initial carry-in of 0, producing sum outputs (S0–S3) and carry outputs (C0–C2). These carry outputs are passed to the next stage along with sum outputs and a constant zero as the third input for each adder in the second row (FA4–FA7). FA4–FA6 add sum and carry values from the first stage to produce the final sum bits (S1–S3) and carry bits. The final carry-out is generated by FA7. This structure avoids the delay caused by carry propagation in traditional ripple-carry adders,

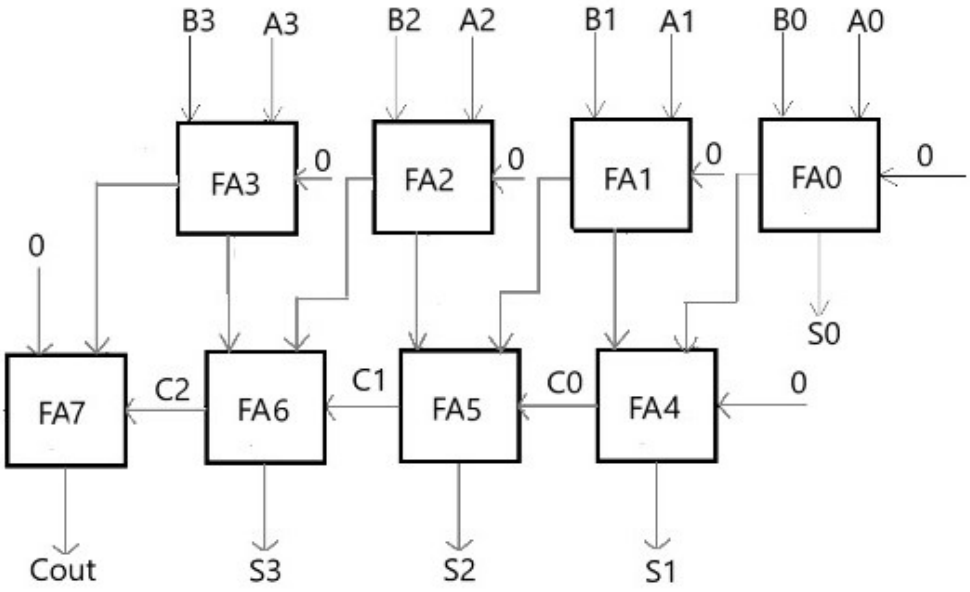


Figure 4.5: Carry Save Adder

improving speed by computing partial sums and deferring carry propagation to the final stage, making CSA efficient for arithmetic circuits like multipliers.

Chapter 5

ADDERS USING REVERSIBLE LOGIC GATES

5.1 Full Adder Using Reversible Logic Gates

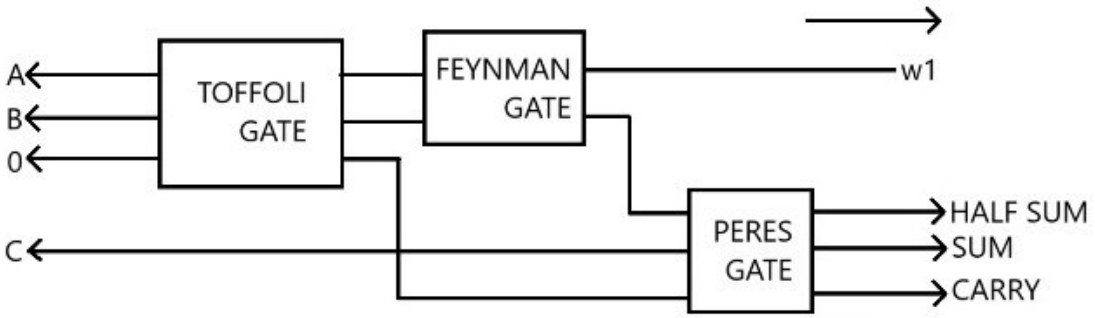


Figure 5.1: Full Adder Using Reversible Logic Gates

A full adder can be implemented using reversible logic gates such as Feynman, Toffoli, and Peres gates. The Feynman gate, also known as the controlled-NOT (CNOT) gate, is a 2x2 reversible gate that outputs (A, A⊕B), useful for copying or XOR operations. The Toffoli gate is a 3x3 reversible gate that maps (A, B, C) to (A, B, C⊕AB), often used to implement AND logic reversibly. The Peres gate combines the functionality of Feynman and Toffoli, producing outputs (A, A⊕B, AB⊕C) from inputs (A, B, C), efficiently computing both XOR and carry. A full adder requires three inputs: A, B, and Cin (carry-in), and two outputs: Sum and Cout (carry-out). The Sum is computed as A⊕B⊕Cin, and Cout as AB + (A⊕B)Cin. Using two Peres gates, we first compute A⊕B and AB, then use the result with Cin to compute the final Sum and intermediate carry. A Toffoli gate can then

be used to OR the two carry terms AB and (A⊕B)Cin. This configuration achieves a full adder with reversible logic, minimal garbage outputs, and efficient gate usage.

5.2 Ripple Carry Adder using Reversible Logic Gates

A 32-bit reversible ripple carry adder is a circuit that adds two 32-bit binary numbers while ensuring the process is reversible, meaning the original numbers can be recovered from the output. It uses special reversible logic gates, like Toffoli or Fredkin gates, instead of regular gates. The adder works by adding each bit of the two numbers, starting from the least significant bit (LSB) to the most significant bit (MSB). For each bit, it calculates the sum and carry, passing the carry

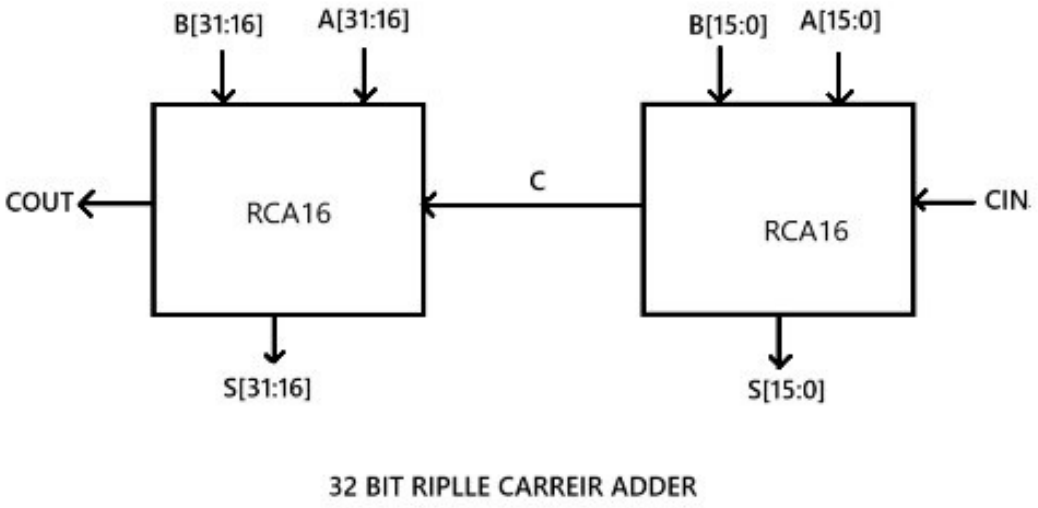


Figure 5.2: 32-bit Ripple Carry Adder Using Reversible Logic Gates

to the next bit. The reversible gates ensure that no information is lost during this process, making it energy-efficient and suitable for applications like quantum computing. The final output is the 32-bit sum and carry-out bit, with the entire operation being energy-conscious and reversible.

5.3 Carry Increment Adder using Reversible Logic Gates

A 32-bit Reversible Carry Increment Adder is a type of digital circuit designed to perform binary addition with reduced power consumption and area, often used in quantum computing or low-power VLSI systems. Unlike traditional adders, a reversible adder ensures that no information is lost during computation, adhering to the principles of reversible logic which allows the circuit to be run backward, thereby minimizing energy dissipation. The adder typically works in two main stages: first, a 4-bit reversible ripple carry adder computes partial sums and carry bits; then, an incrementer circuit adds any necessary carry to the next 4-bit block. This structure is repeated across all 32 bits, effectively breaking the full addition into smaller, manageable chunks and propagating the carry through increment operations. The use of reversible logic gates such as the Toffoli and Fredkin gates ensures that the entire system maintains reversibility. Overall, this type of adder is advantageous in systems where power efficiency and heat reduction are critical.

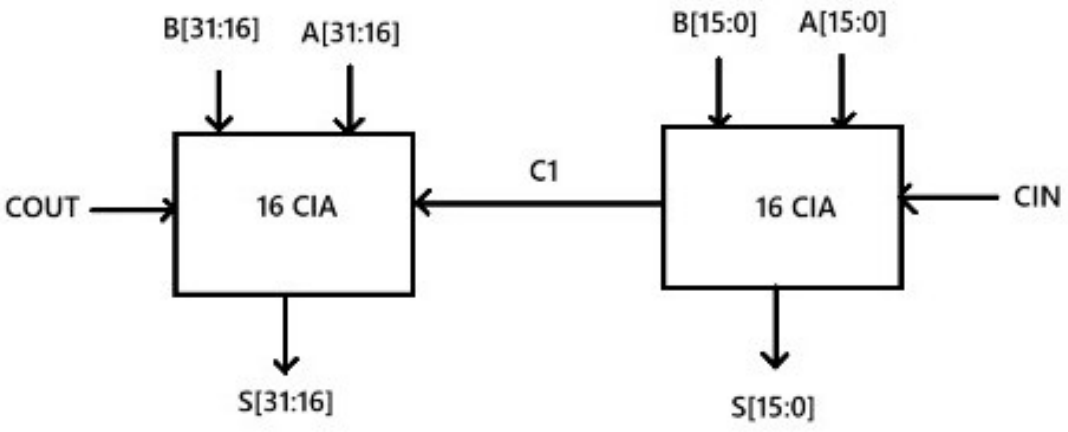


Figure 5.3: 32-bitCarry Increment Adder Using Reversible Logic Gates

5.4 Carry Select Adder using Reversible Logic Gates

A 32-bit Reversible Carry Select Adder is a high-speed adder design that combines reversible logic principles with the traditional carry select architecture to improve performance while minimizing power dissipation. In this design, the 32-bit input is divided into smaller blocks, typically 4 or 8 bits each. For each block (except the first), the adder computes two possible sums in parallel: one assuming the incoming carry is 0 and the other assuming it is 1. Once the actual carry from the previous block is known, a reversible multiplexer selects the correct sum and carry output. Reversible logic gates like Fredkin, Toffoli, and Peres gates are used to ensure that the entire computation is information-lossless, which is crucial for quantum computing and low-power applications. This parallel computation and selection significantly reduce the delay compared to ripple carry adders, making the carry select adder faster while still maintaining reversibility for energy efficiency.

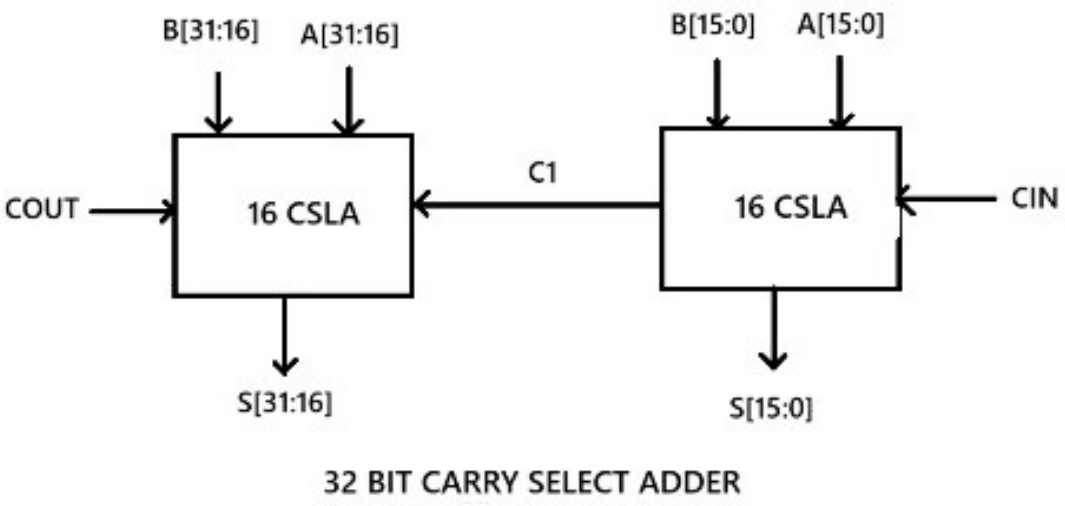


Figure 5.4: 32-bit Carry Select Adder Using Reversible Logic Gates

5.5 Carry Skip Adder using Reversible Logic Gates

A 32-bit Reversible Carry Save Adder is a specialized adder architecture designed for efficient multi-operand addition using reversible logic, which is crucial in quantum computing and low-power VLSI systems. Unlike traditional adders that immediately propagate carries from one bit position to the next, a carry save adder separates the carry and sum outputs, allowing multiple binary numbers to be added simultaneously without waiting for carry propagation. In the reversible version, logic gates such as the Toffoli, Fredkin, and Peres gates are used to ensure that no information is lost during computation, preserving reversibility and minimizing power dissipation. The 32-bit reversible carry save adder processes input operands in parallel, producing a partial sum and carry output, which are then combined in a final stage using a reversible adder (e.g., ripple carry or carry lookahead) to produce the final result. This approach is especially useful in applications requiring the addition of three or more numbers, such as multiplication or digital signal processing, where speed and energy efficiency are critical.

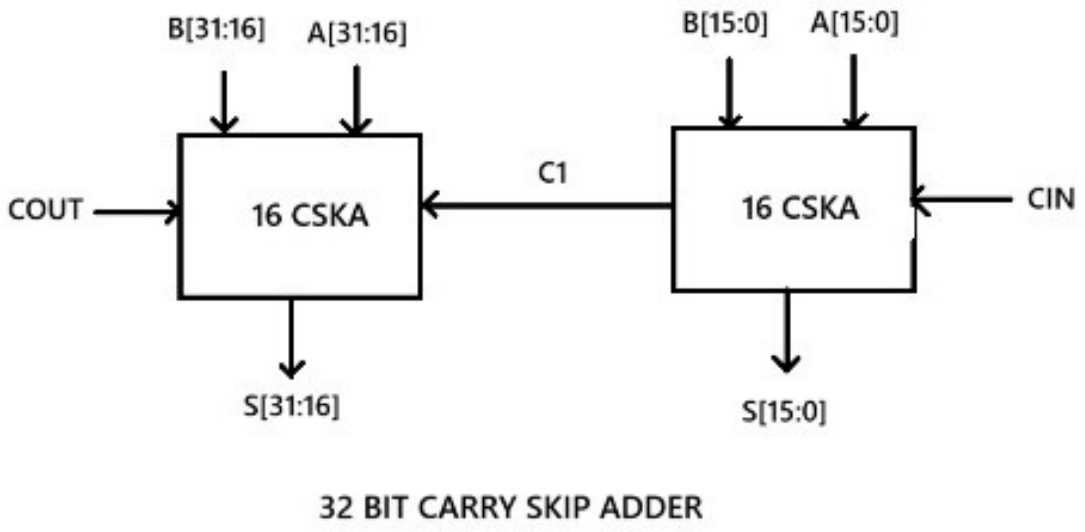


Figure 5.5: 32-bit Carry Skip Adder Using Reversible Logic Gates

5.6 Carry Save Adder using Reversible Logic Gates

A 32-bit Reversible Carry Save Adder is a specialized adder architecture designed for efficient multi-operand addition using reversible logic, which is crucial in quantum computing and low-power VLSI systems. Unlike traditional adders that immediately propagate carries from one bit position to the next, a carry save adder separates the carry and sum outputs, allowing multiple binary numbers to be added simultaneously without waiting for carry propagation. In the reversible version, logic gates such as the Toffoli, Fredkin, and Peres gates are used to ensure that no information is lost during computation, preserving reversibility and minimizing power dissipation. The 32-bit reversible carry save adder processes input operands in parallel, producing a partial sum and carry output, which are then combined in a final stage using a reversible adder (e.g., ripple carry or carry lookahead) to produce the final result. This approach is especially useful in applications requiring the addition of three or more numbers, such as multiplication or digital signal processing, where speed and energy efficiency are critical.

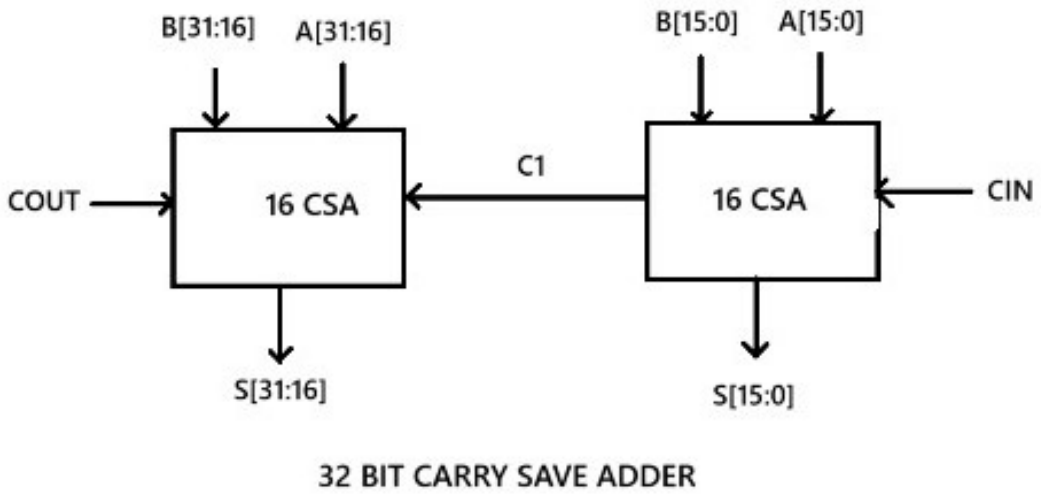


Figure 5.6: 32-bit Carry Save Adder Using Reversible Logic Gates

5.7 Applications

1. Low-Power and Energy-Efficient Computing
2. Quantum Computing
3. Nanotechnology and DNA Computing
4. Optical Computing
5. Cryptographic Hardware
6. Fault-Tolerant and Reliable Systems

5.8 Methodology

Designing adders with reversible gates like Feynman, Peres, and Toffoli gates requires a thoughtful approach that aligns with the principles of reversible computing. Reversible gates play a crucial role in ensuring that computations retain all information, thereby minimizing power dissipation. This process involves grasping how these reversible gates function, creating effective logic for adders, and fine-tuning the design to meet the specific needs of quantum computing.

The first step is to identify the specific adder type to be developed, such as a half adder, full adder, ripple carry adder, carry increment adder,carry save adder,carry select adder,carry skip adder . For instance, a Feynman gate can effectively execute the XOR operation in a half adder to calculate the sum, while the Toffoli gate serves well for the AND operation to produce the carry. In the case of a full adder, Peres gates are favored due to their efficiency in minimizing extraneous outputs and overall quantum cost. For adding multiple bits, ripple carry adders can be created by stringing together several full adders and half adder, basic logic gates can be streamlined by leveraging propagate and generate functions with Toffoli and Feynman gates.

Optimization is a vital aspect of the design journey. It's important to reduce the number of unnecessary outputs and ancilla bits to make the best use of available resources. Furthermore, assessing and minimizing the quantum cost, which indicates the total number of gates employed, is essential. A careful mix of Feynman, Peres, and Toffoli gates can strike a balance between performance and resource effi-

ciency.

Once the design phase is complete, simulation tools like MODELSIM or XILINX can be used to check the circuit's functionality and reversibility. By comparing outputs with the original inputs, we can confirm that the design maintains its reversible nature. Additionally, iterative optimization based on these simulation results can lead to further enhancements in the adder's efficiency.

Chapter 6

VERILOG AND XILINX

6.1 Verilog

Verilog was once begun in the 365 days 1984 by the use of Gateway Design Automation Inc. as a proprietary hardware modeling language. It can be rumored that the usual language used to be designed via utilizing taking elements from essentially the most preferred HDL language of the time, referred to as HiLo, as well as from typical computer languages similar to C. At that time, Verilog used to be no longer standardized and the language modified itself in almost all the revisions that got here out inside of 1984 to 1990.

Verilog simulator first utilized in 1985 and increased appreciably via 1987. The implementation of Verilog simulator supplied via Gateway. The first foremost extension of Verilog is Verilog-XL, which delivered a couple of aspects and implemented the notorious ”XL algorithm” which is an awfully efficient system for doing gate-degree simulation.

Later 1990, Cadence Design system, whose major product on the moment integrated skinny film procedure simulator, decided to accumulate Gateway Automation approach, together with other Gateway merchandise., Cadence now emerge because the owner of the Verilog language, and persevered to market Verilog as each a language and a simulator. Whilst, Synopsys was as soon as promoting the top-down design methodology, utilizing Verilog.

In 1990, Cadence organized the Open Verilog global (OVI), and in 1991 gave it the documentation for the Verilog Hardware Description Language. This was once the occasion which ”opened” the language.

6.2 Basic Concepts

6.2.1 Hardware Description Language

Two matters distinguish an HDL from a linear language like “C”:
Concurrency: The capacity to do a number of matters simultaneously i.e. One of a kind code-blocks can run at the same time.
Timing: Capacity to symbolize the passing of time and sequence routine accordingly

6.2.2 Verilog Introduction

∗ Verilog HDL is a Hardware Description Language (HDL).
∗A Hardware Description Language is a language used to describe a digital procedure; one may describe a digital system at a few levels.
∗ An HDL could describe the design of the wires, resistors and transistors on constructed in Circuit (IC) chip, i.e., the swap level.
∗ It could describe the logical gates and flip flops in a digital system, i.e., the gate level.
∗A first-rate greater measure describes the registers and the transfers of vectors of knowledge between registers. This is called the Register swap degree (RTL).
∗Verilog helps all of those stages.
∗strong operate of the Verilog HDL is that you need to use the equal language for describing, trying out and debugging your procedure.

Verilog Feature

Powerful history: Supported through utilizing OVI, and standardized in 1995 as IEEE std 1364
Industrial help: Speedy simulation and amazing synthesis (eighty five)% were used in ASIC foundries via EE events).
Universal: Makes it possible for entire approach in a single design surroundings (together with evaluation and verification)

Extensibility: Verilog PLI that makes it viable for extension of Verilog capabilities.

6.2.3 Design Flow

The typical design flow is shown in Figure 5.2.7,

6.2.4 Design Specification:

Requirements are written first-Requirement/desires concerning the task.
•Describe the performance total structure of the digital circuit to be designed.
Specification: phrase processor like phrase, Kwriter, AbiWord and for drawing waveform use tools like wave former or scan bencher or phrase.

6.2.5 RTL Description

Conversation of specification in coding format using CAD tools.

6.2.6 Coding Styles

Gate level modeling
Data flow modeling
Behavioral modeling

6.2.7 RTL Coding Editor

Vim, Emacs, Context,HDL, Turbowriter

6.2.8 Functional Verification and Testing

Evaluating the coding with the requisites.
Trying out the procedure of coding with corresponding inputs and outputs.

If trying out fails – as soon as again determine the RTL Description.
Simulation: Modelsim, VCS, Verilog-XL,Xilin

6.2.9 Logic Synthesis

Dialog of RTL description into Gate level -web list kind.
Description of the circuit in phrases of gates and connections.
•Synthesis: Design Compiler, FPGA Compiler, Simplify professional,Leonardo Spectrum, Altera and Xilinx.

6.2.10 Logic Verification and Testing

Sensible Checking of HDL coding by simulation and synthesis. If fails – examine the RTL description.

6.2.11 Flow Planning Automatic Place and Route

• Construction of design with the corresponding gate measure net report.
• Arrange the blocks of the web record on the chip .
• Function & Route: For FPGA use FPGA' companies P&R instrument. ASIC instruments require steeply-priced P&R tools like Apollo. Students can use LASI, Magic

6.2.12 Physical Layout

Bodily design is the system of reworking a circuit description into the bodily design, which describes the role of cells and routes for the interconnections between them.

6.2.13 Layout Verification

• Verifying the bodily layout constitution.
• If any change –as soon as once more investigate ground Planning automated location and Route and RTL Description.

6.2.14 Implementation

Final stage within the design method.
• Implementation of coding and RTL in the form of IC.

6.2.15 Comments

Comments can be inserted in the code for readability and documentation. There are two varieties to introduce comments.
• Single line feedback start with the token // and end with a carriage return
• Multi line feedback start with the token /* and finish with the token */

6.2.16 Identifiers and Keywords

Identifiers are names used to gift an object, similar to a register or perform or a module, a reputation so that it could be referenced from different areas in a description.
Key words are reserved to stipulate the language constructs.
• Identifiers ought to start with an alphabetic personality or the underscore persona (a-z A-Z)
• Identifiers might contain alphabetic characters, numeric characters, the underscore, and the buck sign (a-z, A-Z, 0-9,-)
• Identifiers will also be as a lot as 1024 characters lengthy.
• Keyword phrases are in lowercase.

6.2.17 Examples of Keywords

• always
• begin
• end

6.3 Modules

A module in Verilog contains exotic factors as proven in check. A module definition endlessly starts off evolved with the important thing phrase module. The module title, port record, port declarations, and

no longer obligatory parameters have got to come first in a module definition. Port list and port declarations are present supplied that the module has any ports to engage with the outside atmosphere. The three add-ons inside a module are;
• variable declarations
• dataflow statements
These add-ons can be in any order and at any function within the module definition. The endmodule assertion have to continuously come final in a module definition. All components apart from module, module title, and endmodule are non-obligatory and may also be mixed and matched as per design wants. Verilog makes it possible for a few modules to be outlined in a single file. The modules will also be outlined in any order inside the file.

6.3.1 Example of Module Structure

module$<$ $modulename$ $>$
$(<$ $module_terminals_list$ $>)$;
.....
$<$ $moduleinternals$ $>$
....
endmodule;

6.3.2 Instances

A module presents a template from which that you'd be able to create precise objects. When a module is invoked, Verilog creates an exceptional object from the template. Every object has it's possess establish, variables, parameters and I/O interface. The method of creating objects from a module template is often called instantiation, and the objects are referred to as occasions. In illustration under, the very best-stage block creates four occasions from the T flip- flop (T-FF) template. Every T FF instantiates a D-FF and an inverter gate. Each example have to take delivery of a specific identify.

6.4 Ports

Ports furnish the interface in the course of which a module can preserve up a correspondence with its atmosphere. For illustration, the enter/output pins of an IC chip are its ports. The atmosphere can engage with the module easiest through its ports. The internals of the module customarily will not be seen to the atmosphere. This presents an awfully powerful flexibility to the fashion designer. The internals of the module can also be modified without affecting the atmosphere as long as the interface shouldn't be modified. Ports are additionally referred to as terminals.

6.4.1 Port Declaration

All ports in the list of ports must be declared in the module. Ports can be declared as follows each port in the port list is defined as input, output, or inout, based on the direction of the port signal.

6.4.2 Port Connection Rules

You'll be able to visualize a port as which includes two models, one unit that's inside to the module yet another that is outside to the module. The inner and outside models are linked. There are ideas governing port connections when modules are instantiated inside exceptional modules. The Verilog simulator complains if any port connection rules are violated.

Inputs:

• Internally must be of net data type (e.g. wire).

• Externally the inputs may be connected to a reg or net data type.

Outputs:

• Internally may be of net or reg data type.

• Externally must be connected to a net data type .

Inouts:

• Internally must be of net data type (tri recommended).

• Externally must be connected to a net data type (tri recommended)

6.5 Modeling Concepts

Verilog is each and every a behavioral and a structural language. Internals of each module to be outlined at four stages of abstraction, relying on the wishes of the design. The module behaves identically with the external atmosphere without reference to the level of abstraction at which the module is described. The internals of the module hidden from the environment. As a consequence, the extent of abstraction to explain a module can also be transformed with none trade inside the surroundings. The phases are defined under

Behavioral or algorithmic level This is the superb measure of abstraction offered by utilizing Verilog HDL. A module may also be carried out in phrases of the preferred design algorithm without situation for the hardware implementation small print. Designing at this degree is similar to C programming

Dataflow level At this stage the module is designed via specifying the data go with the flow. The designer is conscious of how knowledge flows between hardware registers and the best way the information is processed within the design.

Gate level The module is implemented in phrases of customary feel gates and interconnections between these gates.

Switch level That's the bottom stage of abstraction offered via Verilog. A module can be applied in phrases of switches, storage nodes, and the interconnections between them. Design at this measure requires expertise of switch-degree implementation fundamental facets. Verilog allows the fashion dressmaker to mix 'n match all 4 phases of abstractions in a design. Inside the digital design group, the time period register swap measure (RTL) is most likely used for a Verilog description that makes use of a blend of behavioral and dataflow constructs and is right to original sense synthesis devices. If a design involves four modules, Verilog makes it possible for every of the modules to be written at a different degree of abstraction. Because the design matures, most modules are changed with gatestage implementations.

Probably, the higher the extent of abstraction, the additional bendy and science impartial the design. As one goes decrease towards tradestage design, the design becomes science centered and rigid. A small modification can purpose an enormous quantity of changes within the design. Evaluating the analogy with C programming and meeting

language programming. It can be less difficult to application in bigger stage language akin to C. The applying can even be quite simply ported to any laptop. On the other hand, if the design on the assembly stage, the application is designated for that pc and aren't equipped to be conveniently ported to an additional computing device.

6.5.1 Gate Level Modeling

Verilog has developed in primitives like gates, transmission gates, and switches. These are rarely utilized in design (RTL Coding), however are used in submit synthesis world for modeling.
ASIC/FPGA cells; these cells are then used for gate degree simulation. Also the output net list structure from the synthesis tool, which is imported into the place and route instrument, can also be in Verilog gate degree primitives.
Gate Types A usual experience circuit can also be designed with the support of use of common experience gates. Verilog helps excellent judgment gates as predefined primitives. Theses primitives are instantiated like modules besides that they're predefined in Verilog and should not have a module definition. All circuit may also be designed by way of utilizing common gates. There are two classes of typical gates: and/or gates and buff/now not gates.

6.5.2 Behavioral and RTL modeling

Verilog supplies designers the potential to explain design effectively in an algorithmic method. In certain words, the trend clothier describes the habits of the circuit. Thus, behavioral modeling represents the circuit at an extraordinarily immoderate stage of abstraction. Design at this measure resembles C programming higher than it resembles digital circuit design. Behavioral Verilog constructs are similar to C program language constructs in plenty of tactics. Verilog is rich in behavioral constructs that furnish the clothier with a high exceptional amount of flexibility.

6.6 Operators

Verilog provided many different operators types. Operators can be,
- Arithmetic Operators
- Relational Operators
- Bit-wise Operators
-Logical Operators
- Reduction Operators
- Shift Operators
- Concatenation Operator
- Replication Operator
- Conditional Operator
- Equality Operator

6.6.1 Arithmetic Operators

These perform arithmetic operations. The + and - can be utilized as each unary (-z) or binary (x-y) operators.
Binary: +, -, *, /,% (the modulus operator)
Unary: +, - (that's used to specify the sign)
Integer division truncates any fractional phase.
The effect of a modulus operation takes the signal of the major operand.
If any operand bit worth is the unknown rate x, then the entire have an effect on price is x.
• Register know-how forms are used as unsigned values (poor numbers are saved in two's complement form)

6.6.2 Relational Operators

Operator	Description
a<b	a less than b
a>b	a greater than b
a<=b	a less than or equal to b
a>=b	a greater than or equal to b

Relational operators evaluate two operands and return a single bit 1or zero. These operators synthesize into comparators. Wire and reg variables are positive therefore (- three'b001) = = 3'b111 and (- 3d001)¿3d1 10, nonetheless for integers -1¡ 6.
The outcome is a scalar worth (illustration a ¡ b)
Zero if the relation is fake (a is higher than b)
1 if the relation is true (a is smaller than b) X if any of the operands has unknown x bits (if a or b includes X) be aware: If any operand is x or z, then the result of that experiment is handled as false (zero)

6.6.3 Bitwise Operators

Bitwise operators perform just a little of intelligent operation on two operands. This take every bit in a single operand and participate within the operation with the corresponding bit inside the different operand. If one operand is shorter than the other, it's going to be accelerated on the left phase with zeroes to verify the size of the longer operand.

Operator	Description
~	Negation
&	And
|	inclusive or
^	exclusive or

Computations include unknown bits, in the following way:
$-> x = x$
$-> 0\&x = 0$
$-> 1\&x = x\&x = x$
$-> 1|x = 1$
$-> 0|x = x|x = x$
When operands are of unequal bit length, the shorter operand is zero-filled in the most significant bit positions.

6.6.4 Logical Operators

Legitimate administrators give back a solitary piece 1 or zero. They're the equivalent as bit canny administrators only for single piece operands. They can take a shot at expressions, whole numbers or organizations of bits, and deal with all qualities which may likewise be nonzero as "1". Coherent administrators are without uncertainty utilized as a part of contingent (if ... Else) explanations seeing that they work with expressions

Operator	Description
!	logic negation
&&	logical and
\|\| | logical or |

Expressions connected with the backing of are assessed from left to appropriate

• Assessment stops as fast when you consider that the impact is noted The impact is a scalar esteemed at:

$-$ >zero if the connection is false

$-$ > 1 if the connection is correct

$-$ > x if any of the operands has x (obscure) bits

6.6.5 Reduction Operators

Rebate administrators work on the greater part of the bits of an operand vector and return a solitary piece cost. These are the unary (one contention) kind of the bit-astute administrators. Reduction administrators are unary.

Operator	Description
&	And
~&	Nand
|	Or
~|	Nor
^	Xor
^~ or ~^	Xnor

They play out somewhat insightful operation on a solitary operand to deliver a solitary piece result.
Reduction unary NAND and NOR administrators work as AND as well as separately, yet with their yields nullified.
$-$ > Unknown bits are dealt with as portrayed some time recently

6.6.6 Shift Operators

Shift administrators move the primary operand by the quantity of bits determined by the second operand. Emptied positions are loaded with zeros for both left and right moves (There is no sign augmentation)

Operator	Description
<<	Left shift
>>	Right shift

The left operand is moved by the quantity of bit positions given by the right operand. The cleared piece positions are loaded with zeroes

6.6.7 Concatenation Operators

The link administrator joins two or more operands to shape a bigger vector. Concatenations are communicated utilizing the prop charac-

ters and , with commas isolating the expressions inside.
− >Example: + a, b[3:0], c, 4'b1001/if an and c are 8-bit numbers, the outcomes has 24 bits Unsized consistent numbers are not permitted in connections.

6.7 Operator Precedence

6.7.1 Procedural Blocks

Verilog behavioral code is inside strategy pieces, yet there is a special case: some behavioral code additionally exist outside method squares. We can see this in subtle element as we gain ground. There are two sorts of procedural squares in Verilog: Introductory: beginning squares execute just once at time zero (begin execution at time zero).
Dependably: dependably squares circle to execute again and again; at the end of the day, as the name proposes, it executes dependably.
In a dependably square, when the trigger occasion happens, the code inside start and end is executed; then by and by the dependably piece sits tight for next occasion activating. This procedure of holding up and executing on occasion is rehashed till reproduction stops.

6.8 Modelsim

6.8.1 Introduction

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartus Prime, Xilinx ISE or Xilinx Vivado. Simulation is performed using the graphical user interface (GUI), or automatically using scripts. Mentor HDL simulation products are offered in multiple editions, such as ModelSim PE and Questa Sim.

Editions Questa Sim offers high-performance and advanced debugging capabilities, while ModelSim PE is the entry-level simulator for

hobbyists and students. Questa Sim is used in large multi-million gate designs, and is supported on Microsoft Windows and Linux, in 32-bit and 64-bit architectures.

ModelSim can also be used with MATLAB/Simulink, using Link for ModelSim. Link for ModelSim is a fast bidirectional co-simulation interface between Simulink and ModelSim. For such designs, MATLAB provides a numerical simulation toolset, while ModelSim provides tools to verify the hardware implementation & timing characteristics of the design.

Language support ModelSim uses a unified kernel for simulation of all supported languages, and the method of debugging embedded C code is the same as VHDL or Verilog.

ModelSim and Questa Sim products enable simulation, verification and debugging for the following languages:
-VHDL
-Verilog
-Verilog 2001
-SystemVerilog
-PSL
-SystemC

6.8.2 Working

1. Open Modelsim

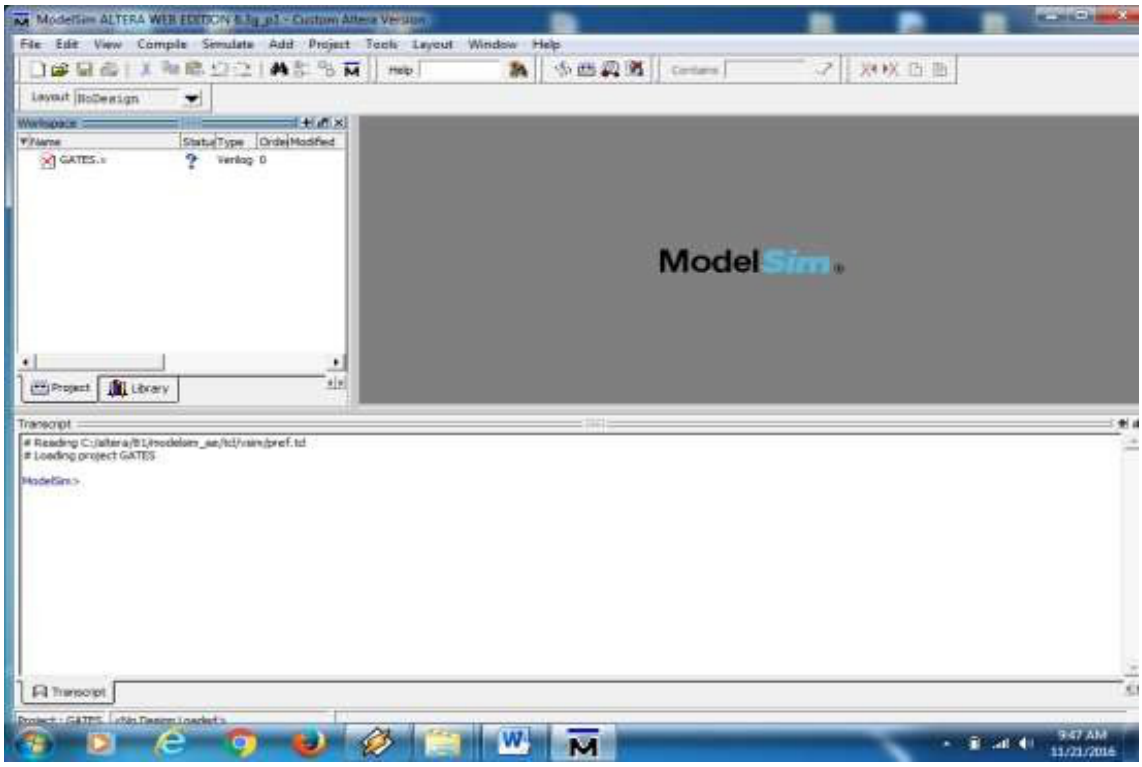


Figure 6.1: Open Modelsim

2.Go to File menu and open new project

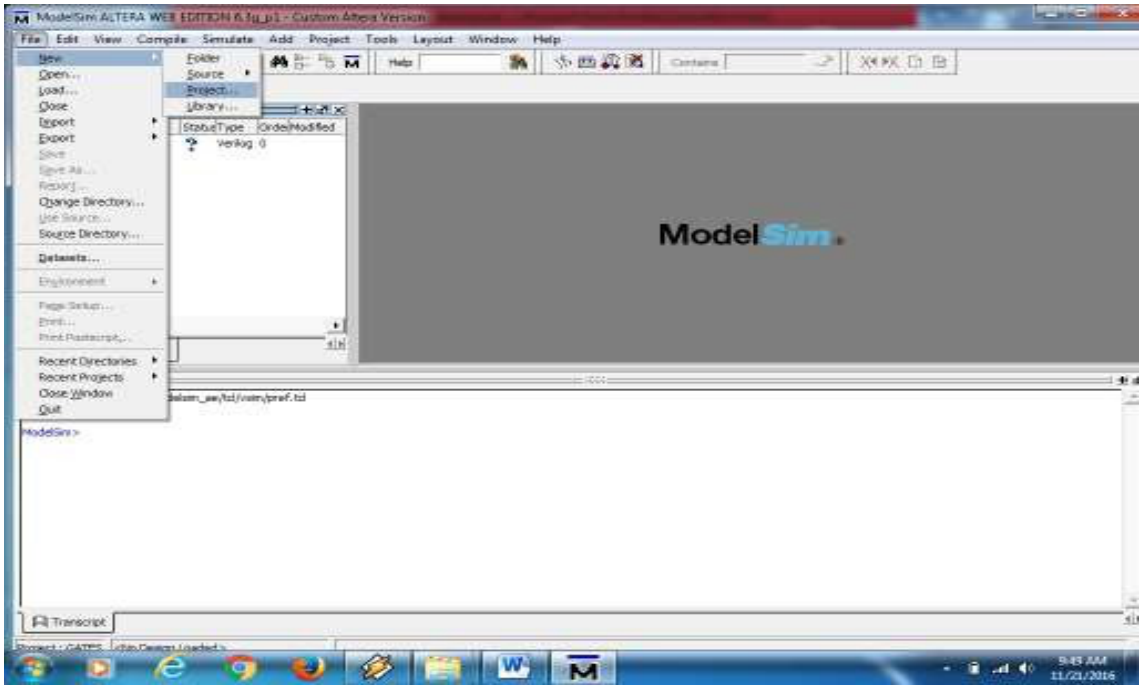


Figure 6.2: Go to File menu and open new project

3.Create new project and name project

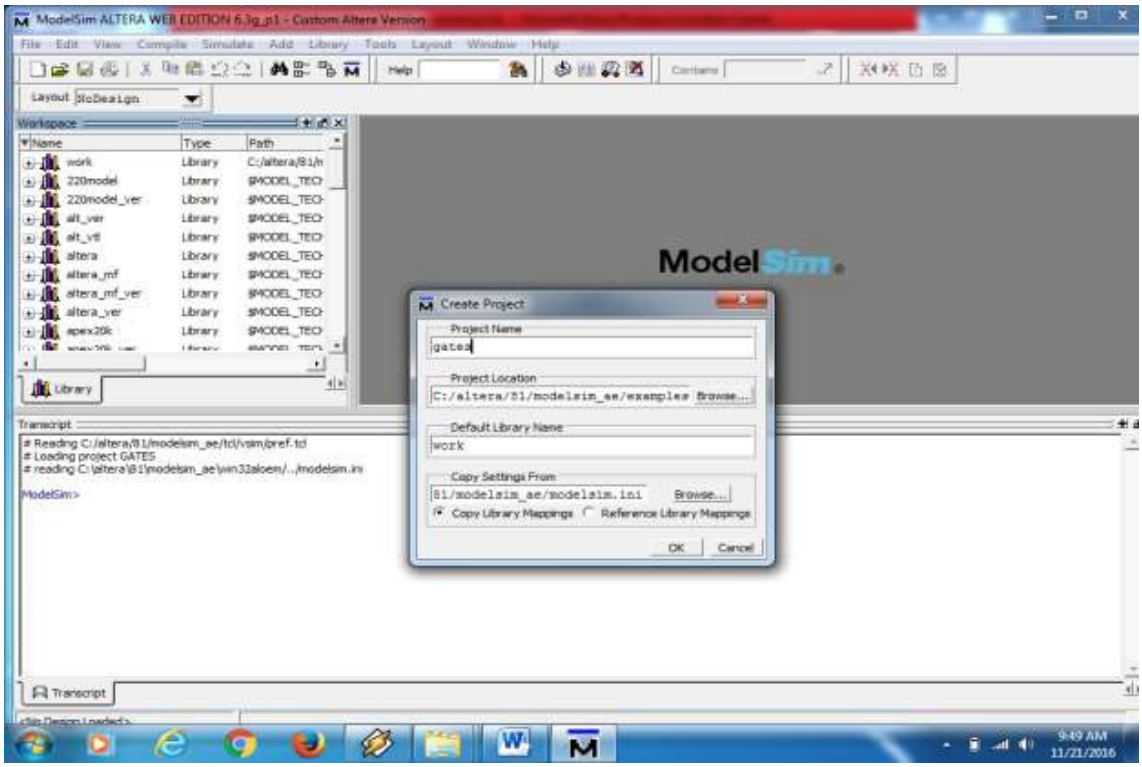


Figure 6.3: Create new project and name project

4.Create new file under the project

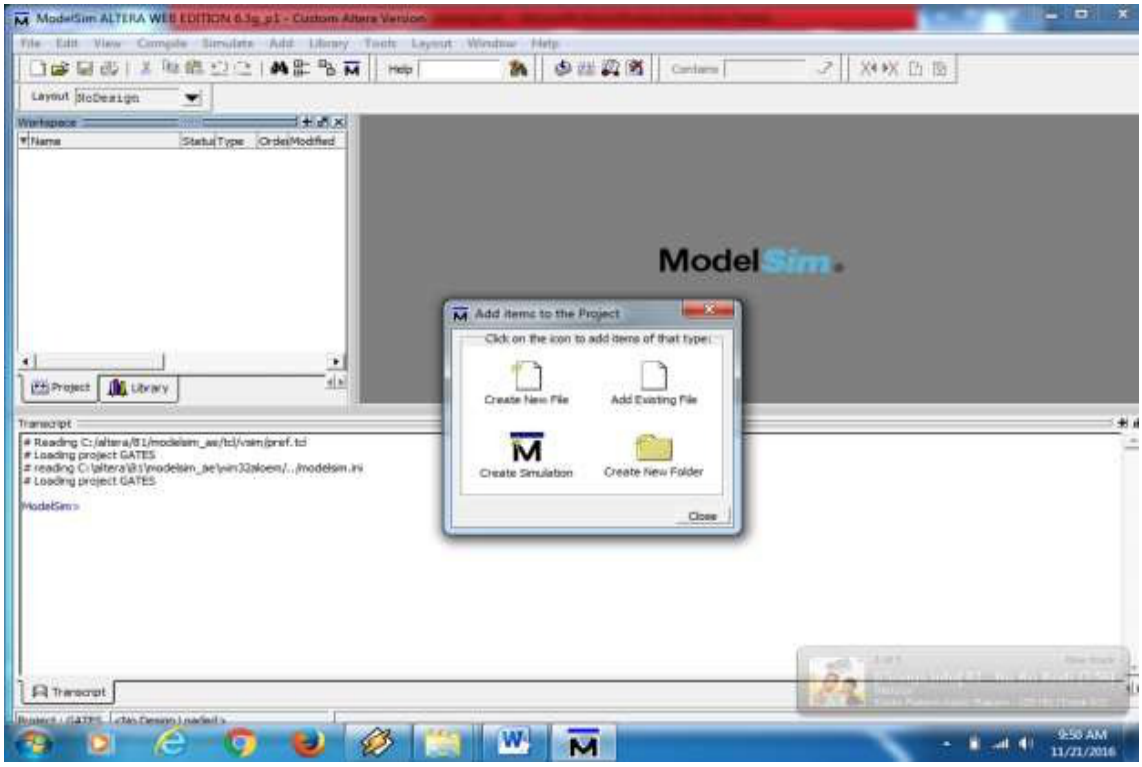


Figure 6.4: Create new file under the project

5.Name the file and select Verilog type

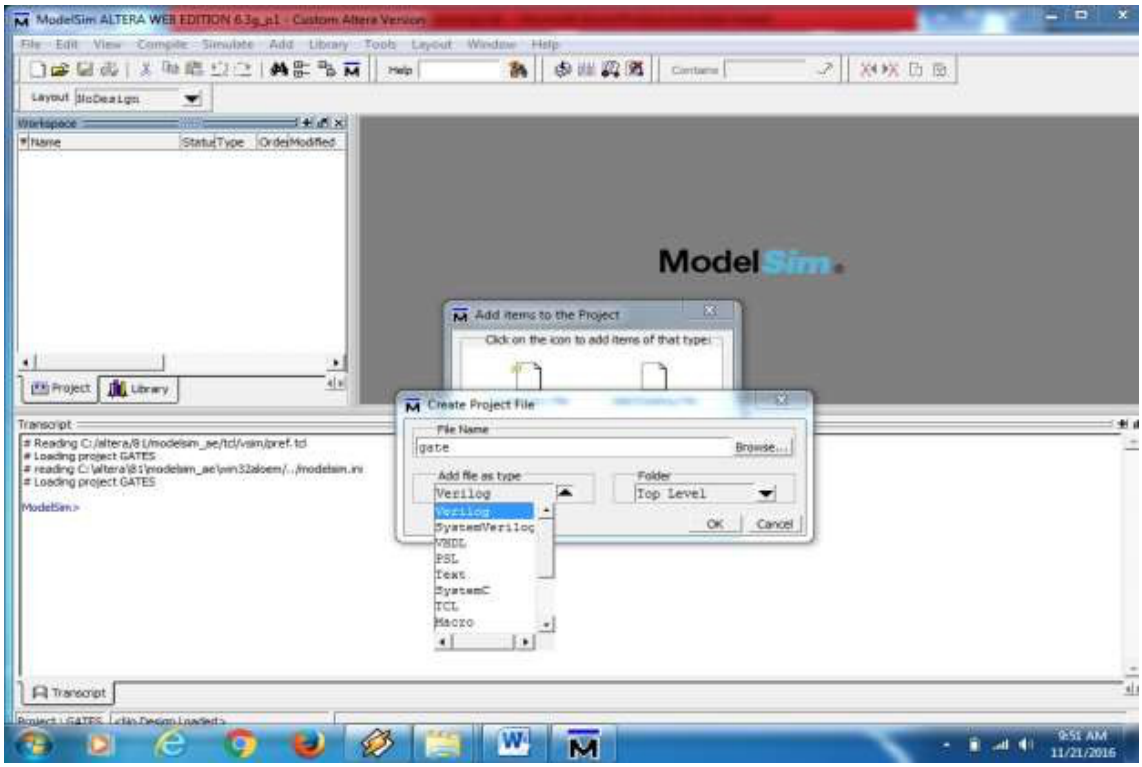


Figure 6.5: Name the file and select Verilog type

6.Double click on file name in left window the file will open

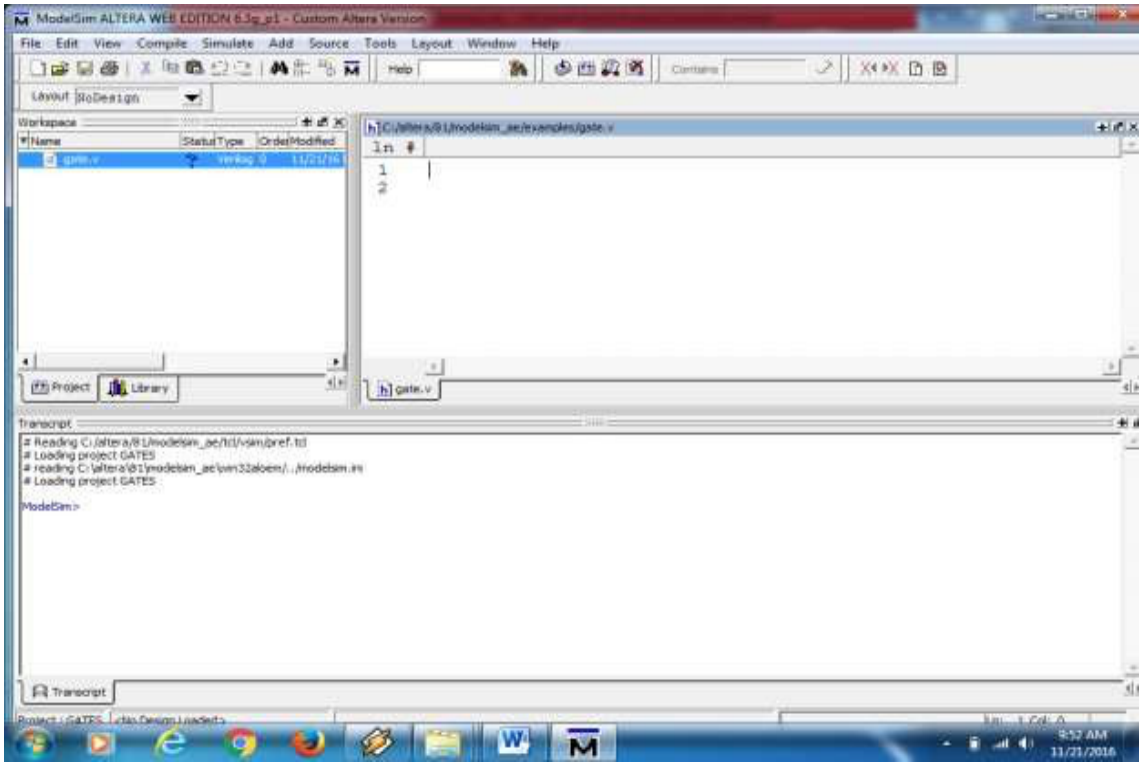


Figure 6.6: Double click on file name in left window the file will open

7.Type the Verilog code in the file

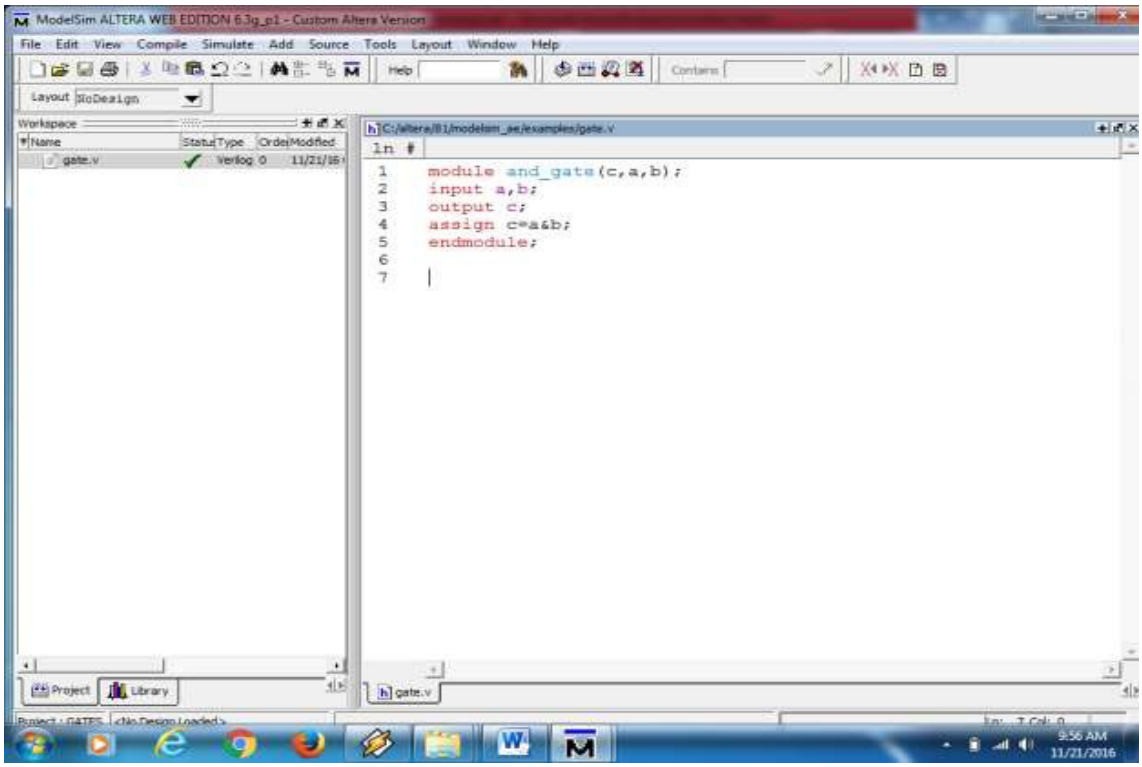


Figure 6.7: Type the Verilog code in the file

8.compile the code

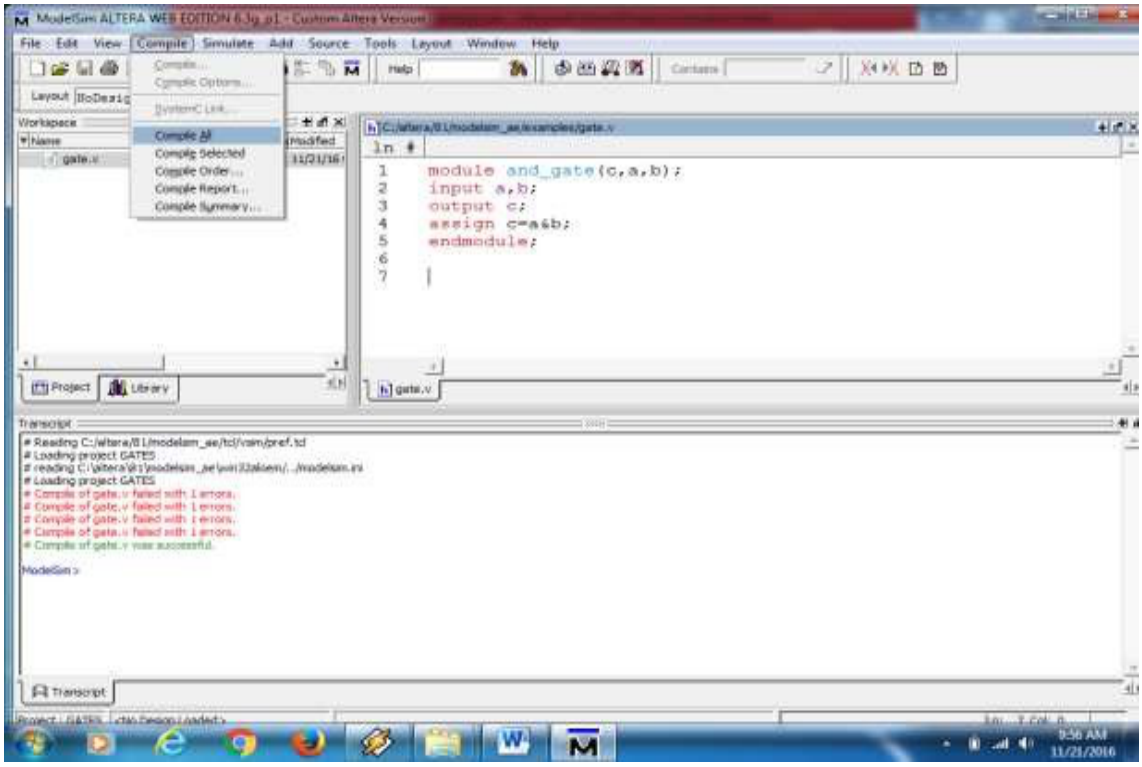


Figure 6.8: Compile the code

9.once compiled with no errors then simulate

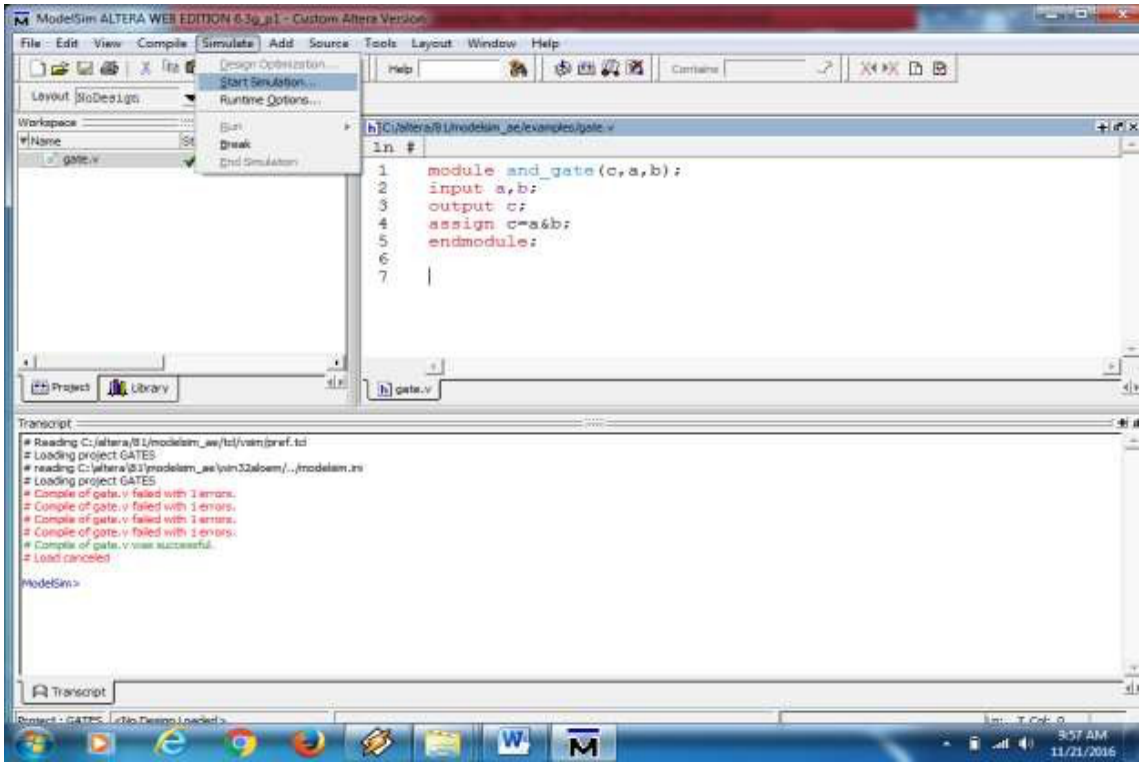


Figure 6.9: Once compiled with no errors then simulate

10.click start simulation

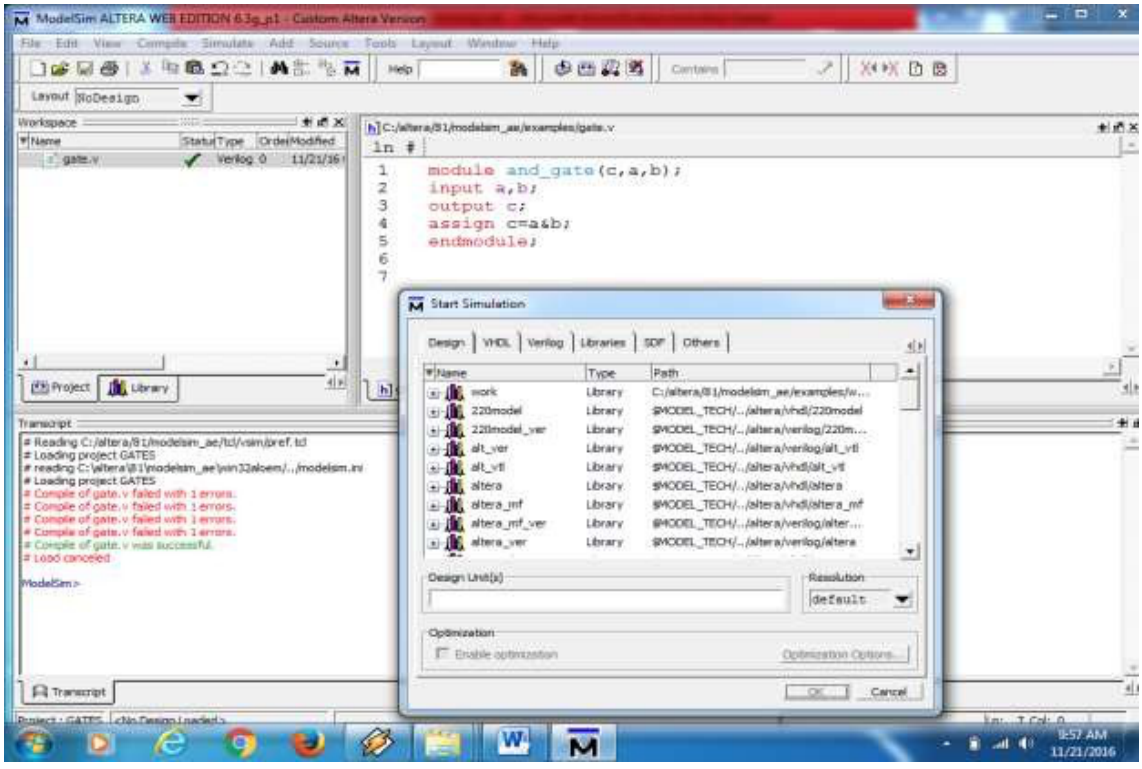


Figure 6.10: Start simulation

11.Go to work library

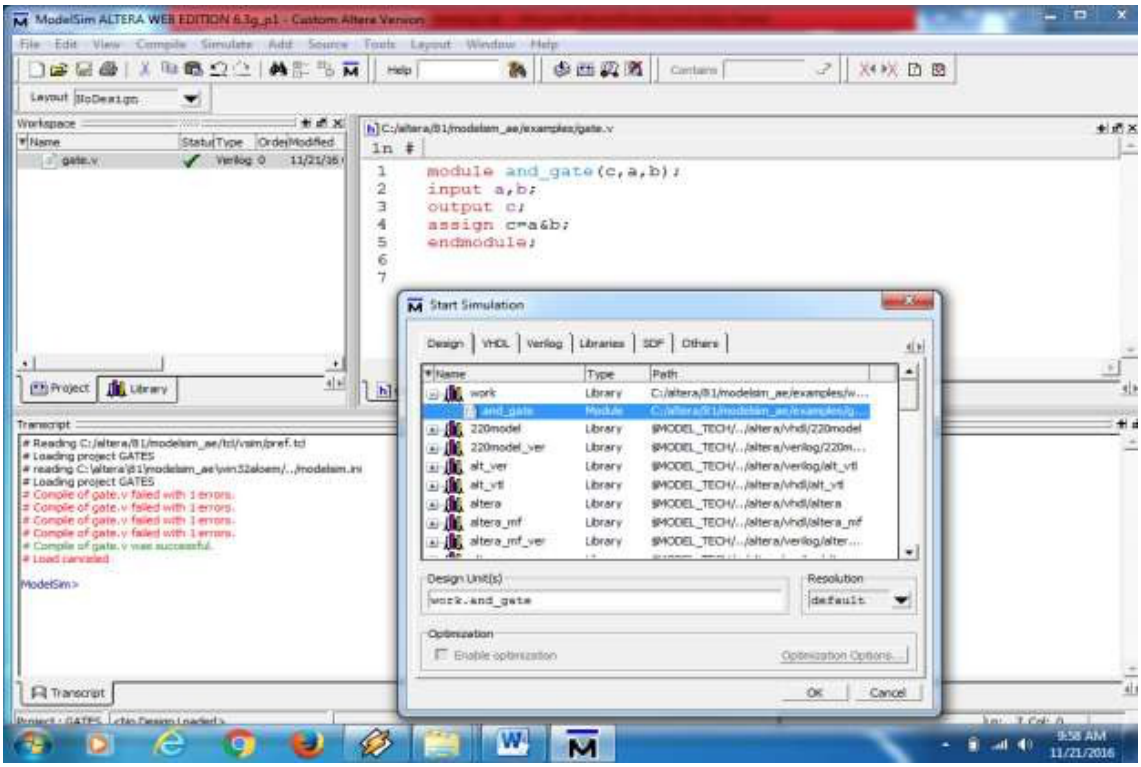


Figure 6.11: Go to work library

12.In work library select your module and click ok

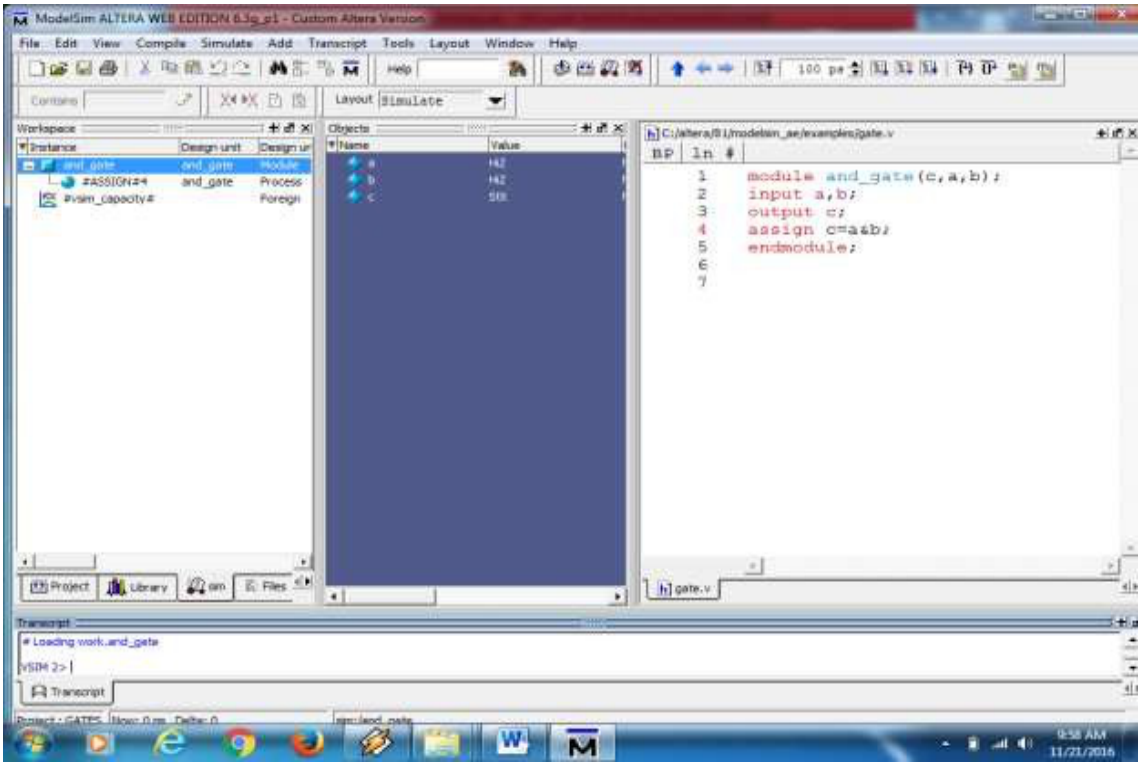


Figure 6.12: In work library select your module and click ok

13.Add all signals to wave

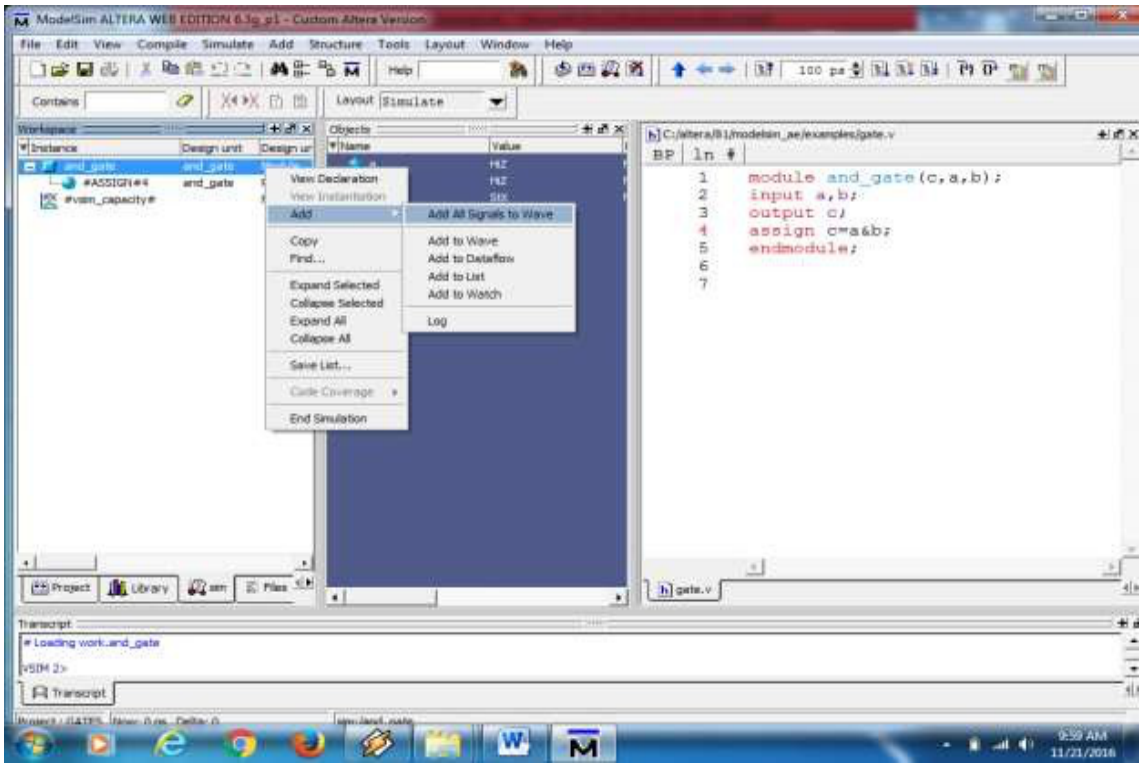


Figure 6.13: Add all signals to wave

14.Inputs and outputs will be in waveform window

Figure 6.14: Inputs and outputs will be in waveform window

15.Assign values to inputs and run

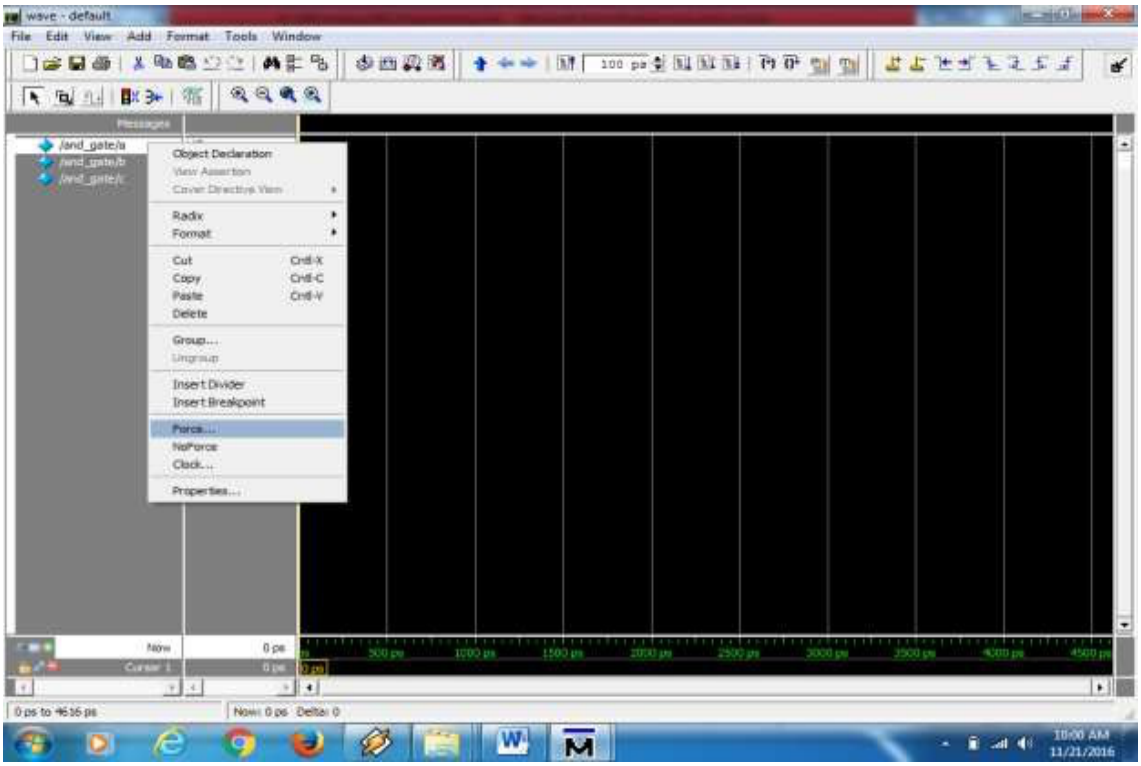
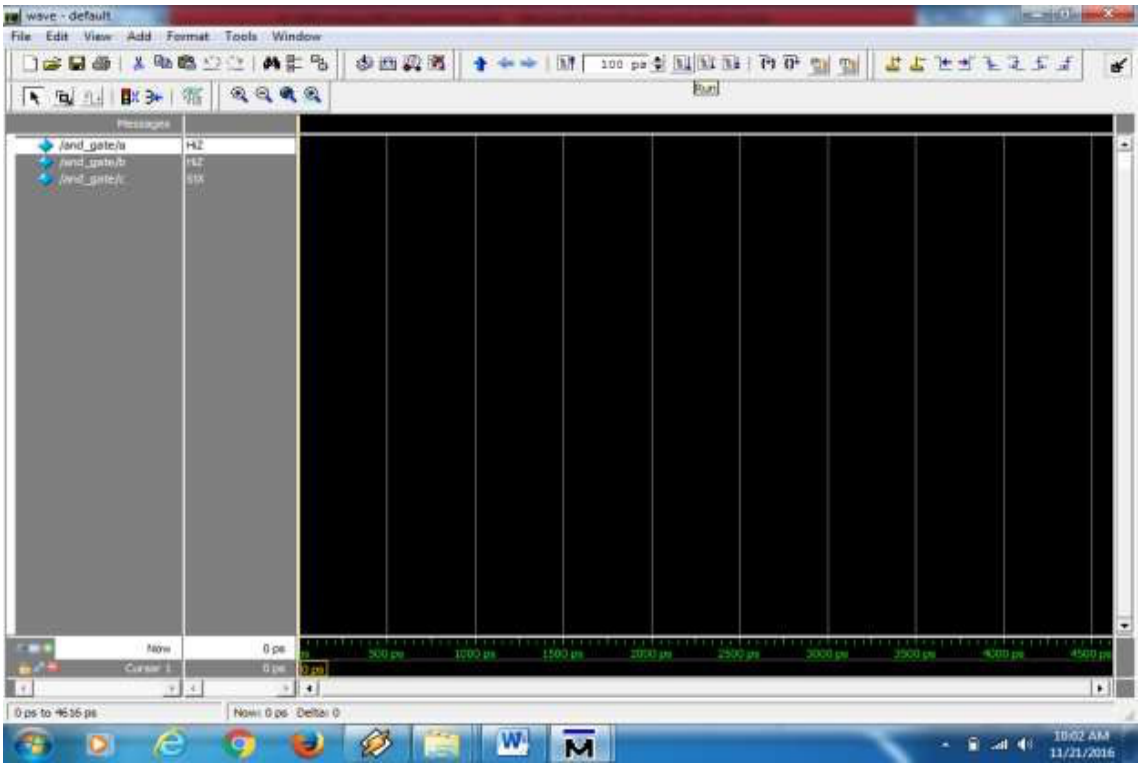
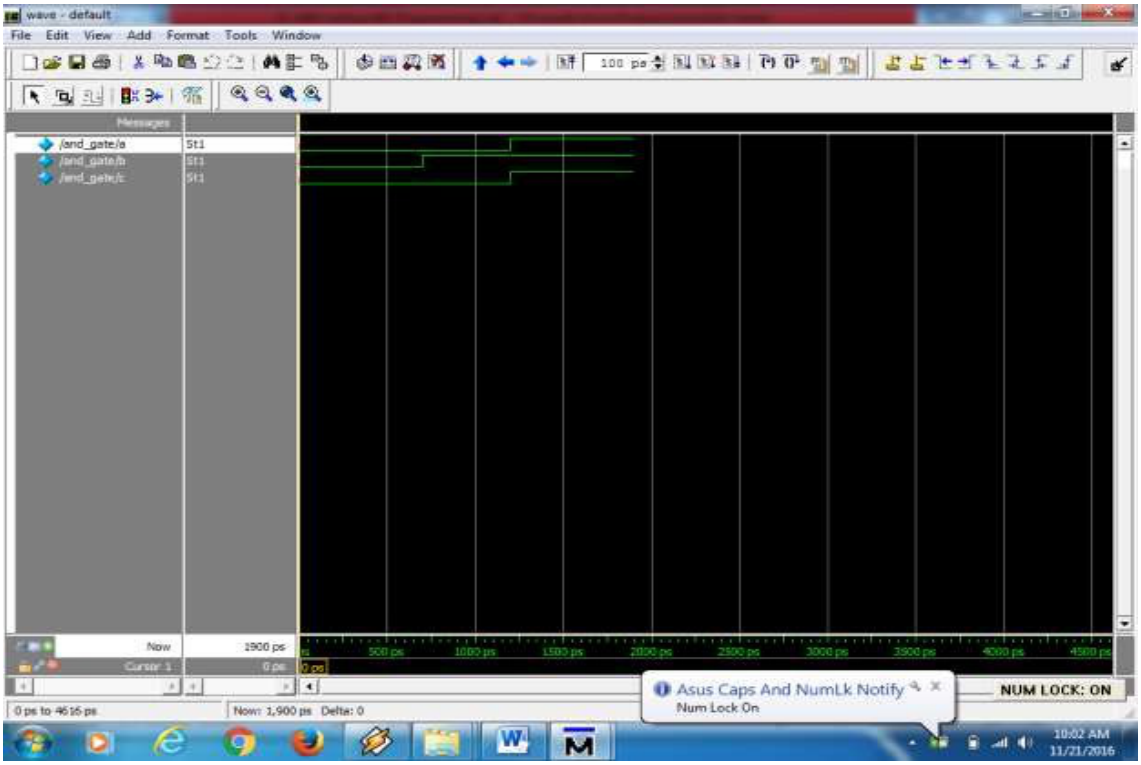


Figure 6.15: Assign values to inputs and run





16.End simulation once done

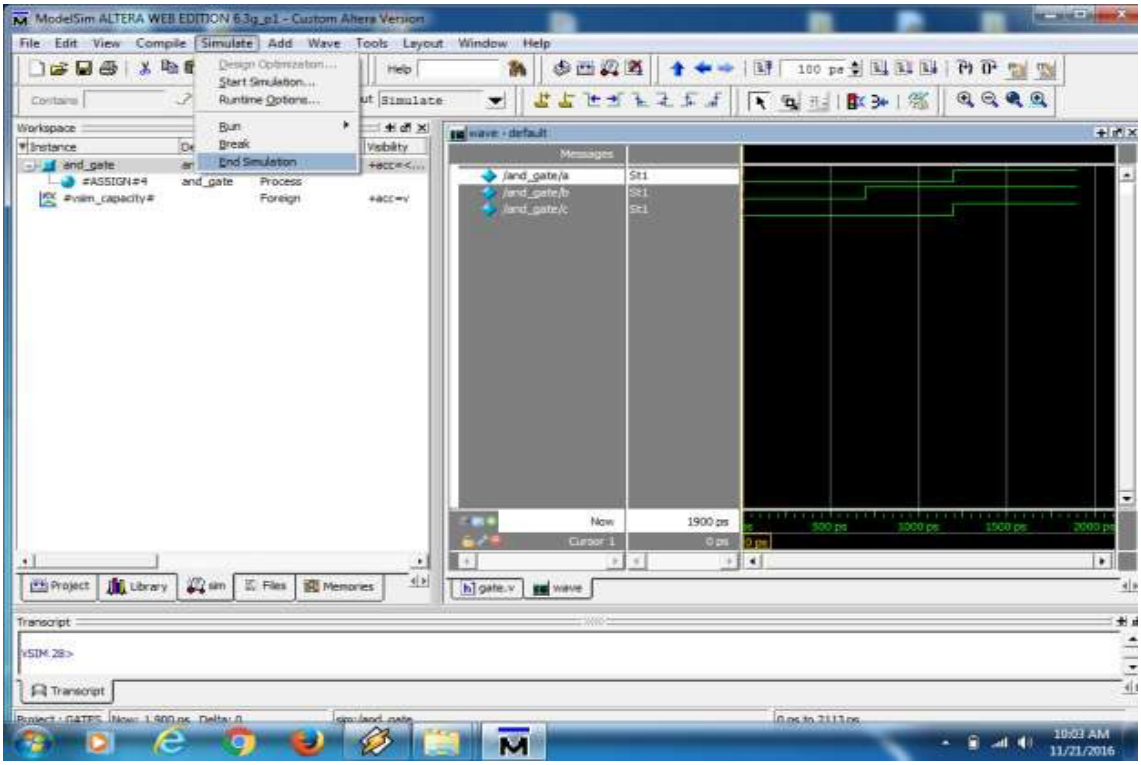


Figure 6.16: End simulation

6.9 Xilinx Verilog HDL Tutorial

Getting started

On the off chance that you wish to take a shot at this instructional exercise and the research facility at home, you should download and introduce Xilinx and ModelSim. These apparatuses both have free understudy forms. It would be ideal if you perform Appendix B, C, and D in a specific order before proceeding with this instructional exercise. Moreover in the event that you wish to buy your own particular Spartan3 board, you can do as such at Diligent's Website. Diligent offers scholastic evaluating. If you don't mind take note of that you should download and introduce Diligent Adept programming. The product contains the drivers for the board that you require furthermore gives the interface to program the board.

6.9.1 Introduction

Xilinx Tools is a suite of programming devices utilized for the outline of advanced circuits actualized utilizing Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The outline method comprises of (a) configuration passage, (b) blend and execution of the outline, (c) useful reenactment and (d) testing and confirmation. Advanced outlines can be entered in different ways utilizing the above CAD devices: utilizing a schematic section instrument, utilizing an equipment portrayal dialect (HDL) – Verilog or VHDL or a mix of both. In this lab we will just utilize the configuration stream that includes the utilization of Verilog HDL.

The CAD devices empower you to outline combinational and consecutive circuits beginning with Verilog HDL plan determinations. The progressions of this outline strategy are recorded beneath:

Make Verilog outline information file(s) utilizing format driven editorial manager. Accumulate and execute the Verilog outline file(s).

Make the test-vectors and recreate the outline (useful reenactment) without utilizing a PLD (FPGA or CPLD).

Allot info/yield pins to execute the configuration on an objective gadget.

Download bit stream to a FPGA or CPLD gadget.

Test plan on FPGA/CPLD gadget

A Verilog information document in the Xilinx programming environment comprises of the accompanying portions:
Header: module name, rundown of information and yield ports.
Announcements: info and yield ports, registers and wires.
Rationale Descriptions: conditions, state machines and rationale capacities. End: endmodule. All your designs for this lab must be specified in the above Verilog input format. Note that the state diagram segment does not exist for combinational logic designs

Programmable Logic Device FPGA

On this lab digital designs will also be implemented within the Basys2 board which has a Xilinx Spartan3E –XC3S250E FPGA with CP132 package deal. This FPGA phase belongs to the Spartan cherished ones of FPGAs. These contraptions are available in a type of programs. We can be utilizing devices that are packaged in 132 pin bundle care for the next phase wide variety: XC3S250E-CP132. This FPGA is a gadget with about 50K gates.
Specific information on this device is on hand on the Xilinx internet site

Creating a New Project

Xilinx instruments can also be started by clicking on the assignment Navigator Icon on the home windows computer. This must open up the assignment Navigator window on your display. This window indicates the last accessed project. Xilinx Project Navigator window.

Opening a Project

Choose File− >New venture to create a new undertaking. This will likely deliver up a new assignment window on the laptop. Refill the crucial entries as follows:
Project Title: Write the name of your new undertaking.
Task Vicinity: The listing the place you want to retailer the brand new venture (note: do not specify the undertaking vicinity as a folder on desktop or a folder within the Xilinx in listing. Your H: pressure is the high-quality situation to place it. The challenge place direction is not to have any spaces in it eg: C:NivashTAnewlabsampleexerciseso-gate

will not be for use) go away the highest degree module form as HDL.
Instance: If the assignment title have been "o-gate", enter "o-gate" because the task identify and then click on "subsequent".
For each of the properties given below, click on the 'value' area and select from the list of values that appear.
Device Household: Household of the FPGA/CPLD used. On this laboratory we will be making use of the Spartan3E FPGA
Gadget: The range of the distinctive gadget. For this lab you can also enter XC3S250E (this will also be placed on the hooked up prototyping board).
Package Deal: The style of package handle the quantity of pins. The Spartan FPGA used on this lab is packaged in CP132 bundle.
% Grade: The velocity grade is "-4".
Synthesis instrument: XST [VHDL/Verilog]
Simulator: The instrument used to simulate and confirm the performance of the design. Modelsim simulator is integrated within the Xilinx ISE. As an end result select "ModelsimXE Verilog" as the simulator or even Xilinx ISE Simulator can be used.
Then click on subsequent to avoid wasting tons of the entries.
All undertaking files reminiscent of schematics, net lists, Verilog documents, VHDL files, and many others.Will likely be stored in a subdirectory with the assignment title. A task can only have one top degree HDL source file (or schematic). Modules will also be delivered to the venture to create a modular, hierarchical design.
With a purpose to open an existing mission in Xilinx instruments, decide on File− >Open mission to show the list of tasks on the laptop. Pick the undertaking you need and click ok.
Clicking on NEXT on the above window brings up the following window:
In this lab we can enter a design making use of a structural or RTL description using the Verilog HDL. Which you can create a Verilog HDL enter file (.V file) making use of the HDL Editor available within the Xilinx ISE tools (or any text editor).
Within the previous window, click on the new supply
A window pops up as proven in Figure 4. (observe: "Add to task" choice is selected with the aid of default. If you don't decide upon it then you'll have to add the new supply file to the challenge manually.) Prefer Verilog Module and inside the "File determine:" area, enter the determine of the Verilog supply file you're going to create. Moreover

make targeted that the alternative Add to project is chosen so that the supply needn't be delivered to the venture again. Then click on subsequent to take delivery of the entries. This pops up the following window.
Define Verilog Source window (snapshot from Xilinx ISE software)
Within the Port identify column, enter the names of all input and output pins and specify the path thus. A Vector/Bus will also be outlined by way of getting into right bit numbers in the MSB/LSB columns. Then click on subsequent> to get a window showing all of the new supply information. If any changes are to be made, simply click on ¡Back to go back and make changes. If everything is accept table, click on Finish > next >next > finish to proceed.
Whilst you click on on conclude, the deliver file will most likely be displayed inside the sources window within the mission Navigator.
If a source has to be eliminated, without problems appropriate click on on the supply file inside the Sources in mission window within the challenge Navigator and opt for dispose of in that. Then choose mission $-$ > Delete Implementation understanding from the challenge Navigator menu bar to dispose of any associated records

Editing the Verilog Source File

The source file will now be displayed in the task Navigator window. The supply file window can be utilized as a textual content editor to make any vital alterations to the supply file. The entire input/output pins will likely be displayed. Save your Verilog application periodically through settling on the File-¿save from the menu. You can also edit Verilog applications in any textual content editor and add them to the undertaking directory utilizing "Add replica source".
Verilog supply code editor window in the task Navigator (from Xilinx ISE utility) together with good judgment within the generated Verilog supply code template a short Verilog Tutorial is to be had in Appendix-A. As a result, the language syntax and development of logic equations may also be pointed out Appendix-A.
The Verilog supply code template generated suggests the module name, the report of ports and likewise the declarations (input/output) for each and every port.
Combinational common feel code can be delivered to the Verilog code

after the declarations and earlier than the endmodule line.
For example, an output z in an OR gate with inputs a and b will even be b; take into account that the names are case sensitive. Other constructs for modeling the great judgment perform: A given excellent judgment function can also be modeled in lots of approaches in Verilog. Correct right here is another illustration where the good judgment operate, is applied as a fact desk utilizing a case announcement:

6.9.2 Synthesis and Implementation of the design

The design wants to be synthesized and utilized earlier than it could be checked for correctness, by means of going for walks practical simulation or downloaded onto the prototyping board. With the highest-measure Verilog file opened (can also be completed through double-clicking that file) inside the HDL editor window inside the correct half of the task Navigator, and the view of the venture being within the Module view , the put into effect design option will also be obvious within the procedure view. Design entry utilities and Generate Programming File options can also be seen inside the strategy view. The former can be utilized to include person constraints, if any and the latter may also be mentioned later.
To synthesize the design, double click on on the Synthesize Design choice inside the approaches window.
To put into effect the design, double click on the enforce design substitute inside the strategies window. It will go via steps like Translate, Map and role & Route. If any of these steps would now not be completed or executed with blunders, it is going to location a X mark in front of that, or else a tick mark can be positioned after each of them to indicate the optimistic completion. If the entire factor is finished effectually, a tick mark will also be put before the put in force Design substitute. That you would be able to nonetheless seem on the warnings or mistakes inside the Console window reward at the backside of the Navigator window. At any time when the design file is saved; all these marks disappear soliciting for a contemporary compilation.
Implementing the Design (snapshot from Xilinx ISE software).
The schematic diagram of the synthesized Verilog code can be considered by means of double clicking View RTL Schematic below Synthesize-XST menu in the approach Window. This is competent to be a useful procedure to debug the code if the output just isn't assembly our re-

quirements inside the proto kind board. Via utilizing double clicking it opens the easiest stage module displaying handiest input(s) and output(s) as proven under. By double clicking the rectangle, it opens the realized internal logic as shown below.

6.9.3 Working

1. Now start the Xilinx ISE Design Suite 12.1.

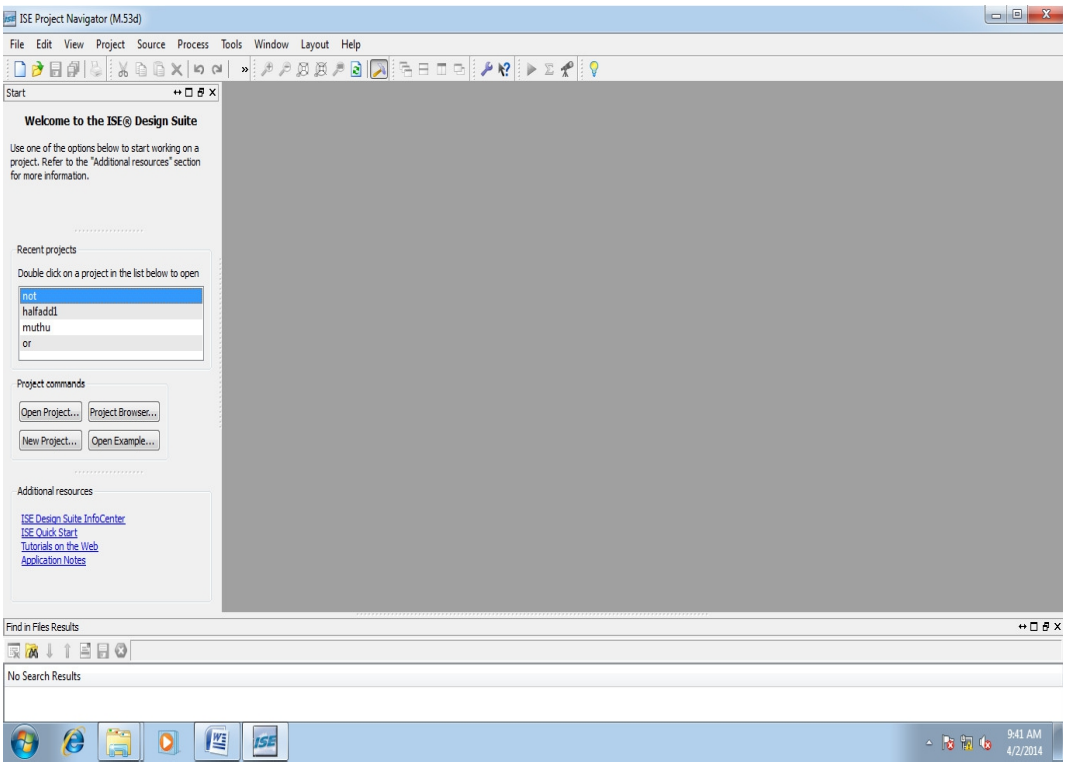


Figure 6.17: Open Xilinx ISE Design suite 12.1

2. Go to file and click new project.

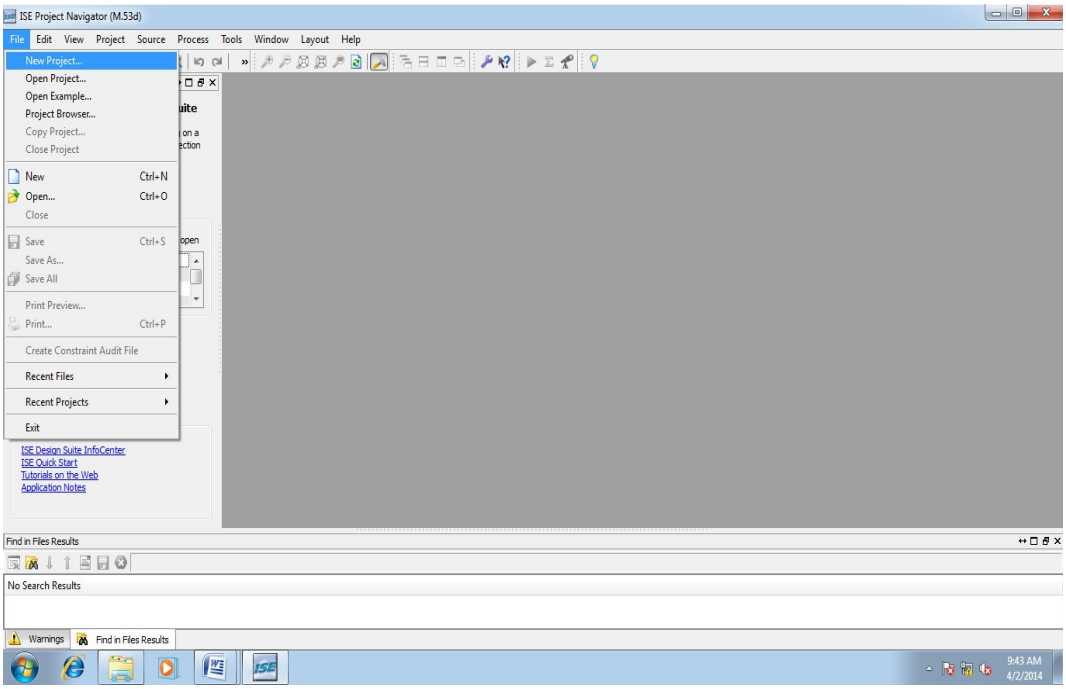


Figure 6.18: Create new project

3. Enter the project name and click next

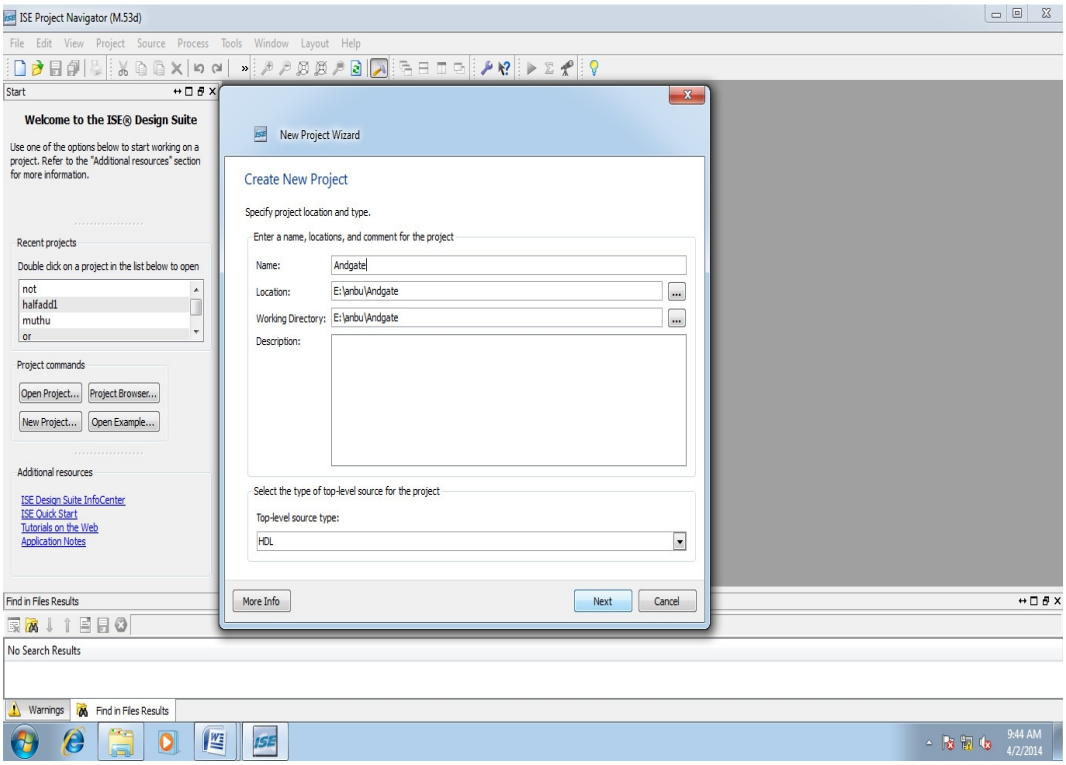


Figure 6.19: Enter the project name

4. Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next and click Finish.

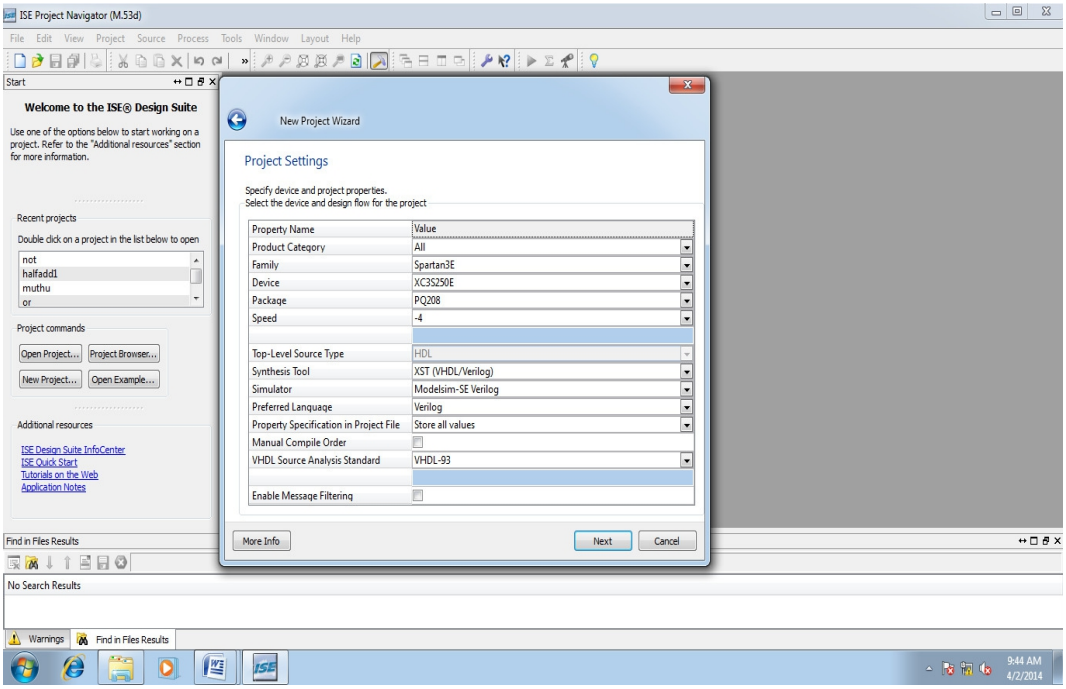


Figure 6.20: Select Family name,Speed,Simulator as Verilog

5. Click new source.

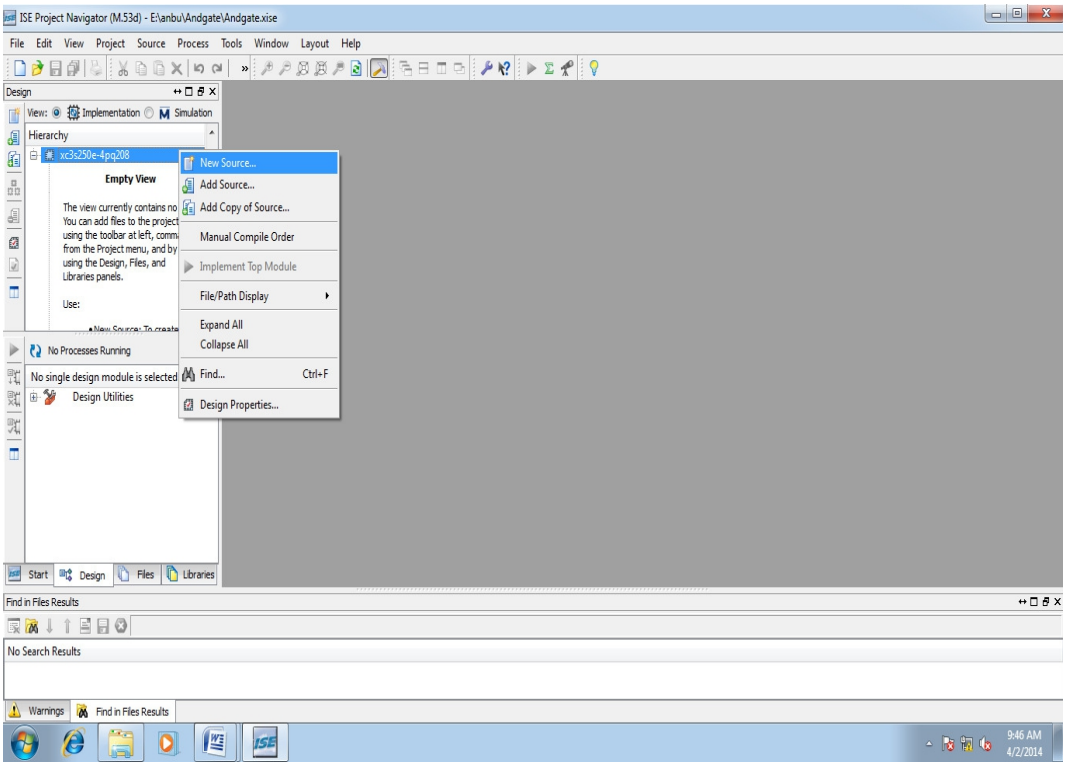


Figure 6.21: Select New Source

6. Select verilog module and type file name and click next.

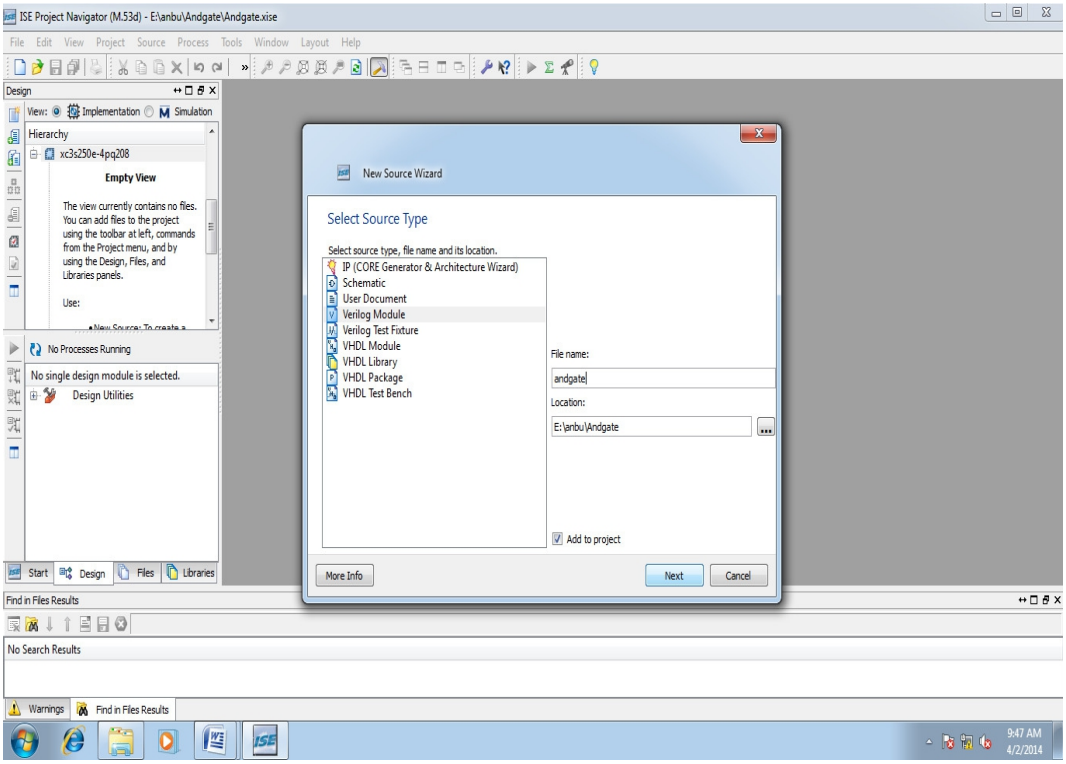


Figure 6.22: Select Verilog Module

7. Assign input and output port and click next.

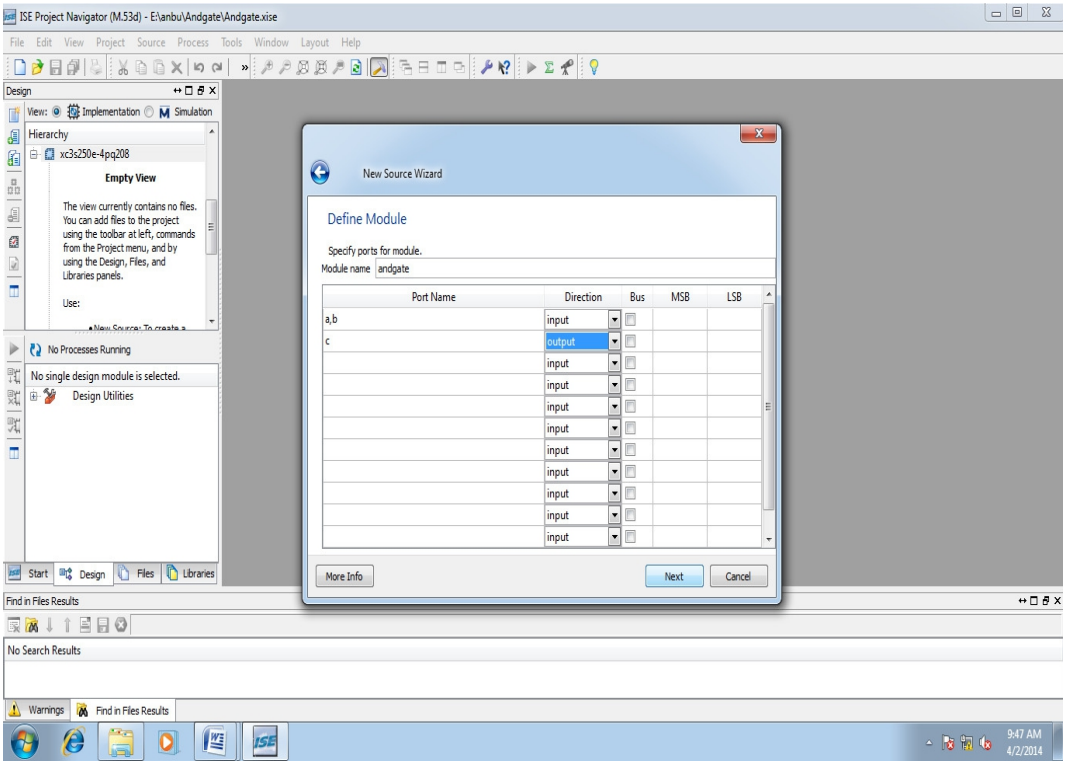


Figure 6.23: Assign input and output

8. Finally the report is shown click finish.

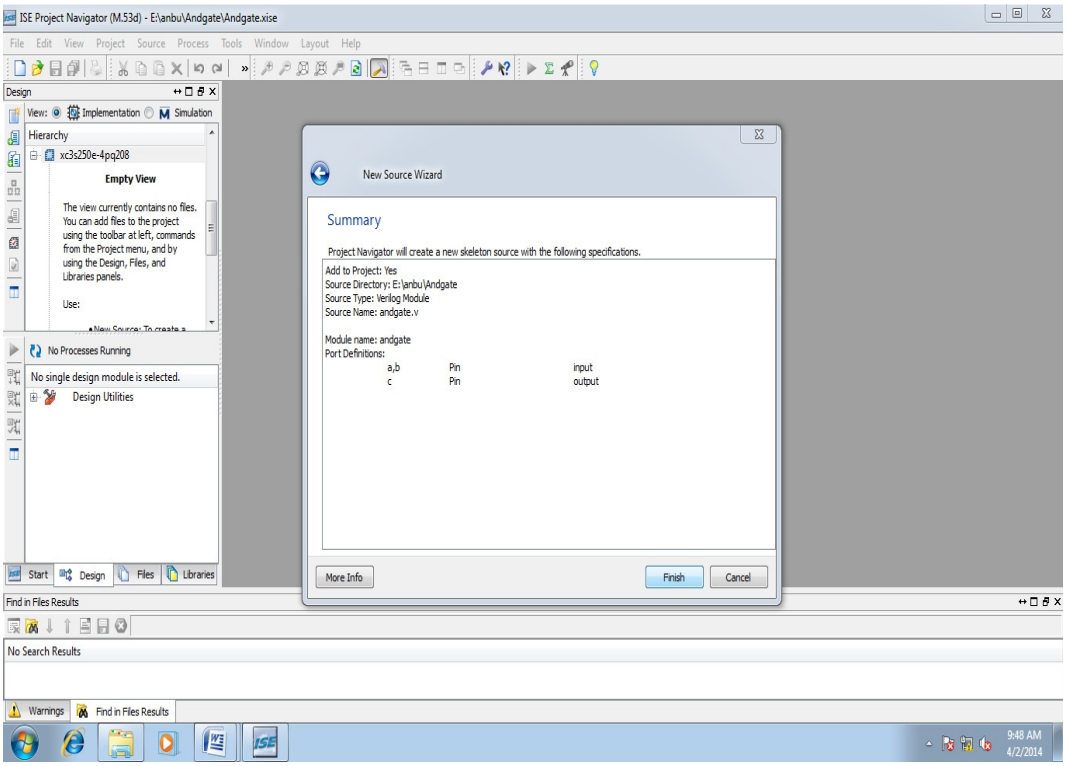


Figure 6.24: End the report

9. Type the program save and click synthesis.

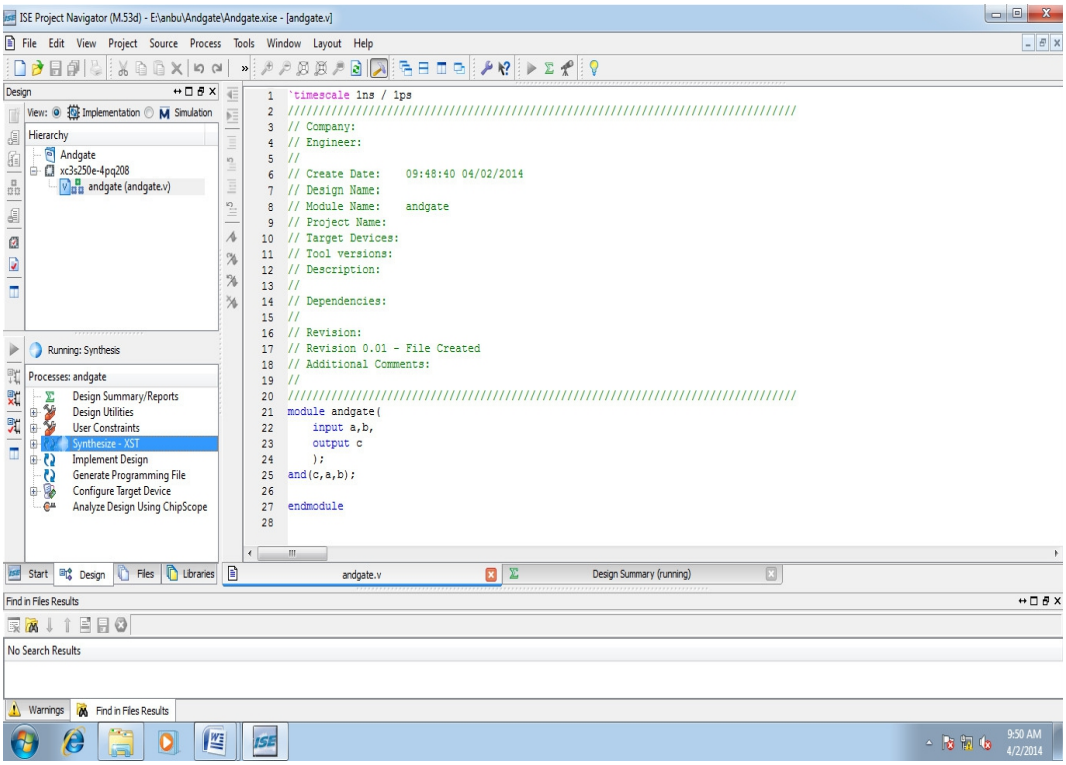


Figure 6.25: Type the program to synthesis

10. To see the output wave form change the source to behavioral simulation and click simulator behavior model in modelsim simulator. And Click No.

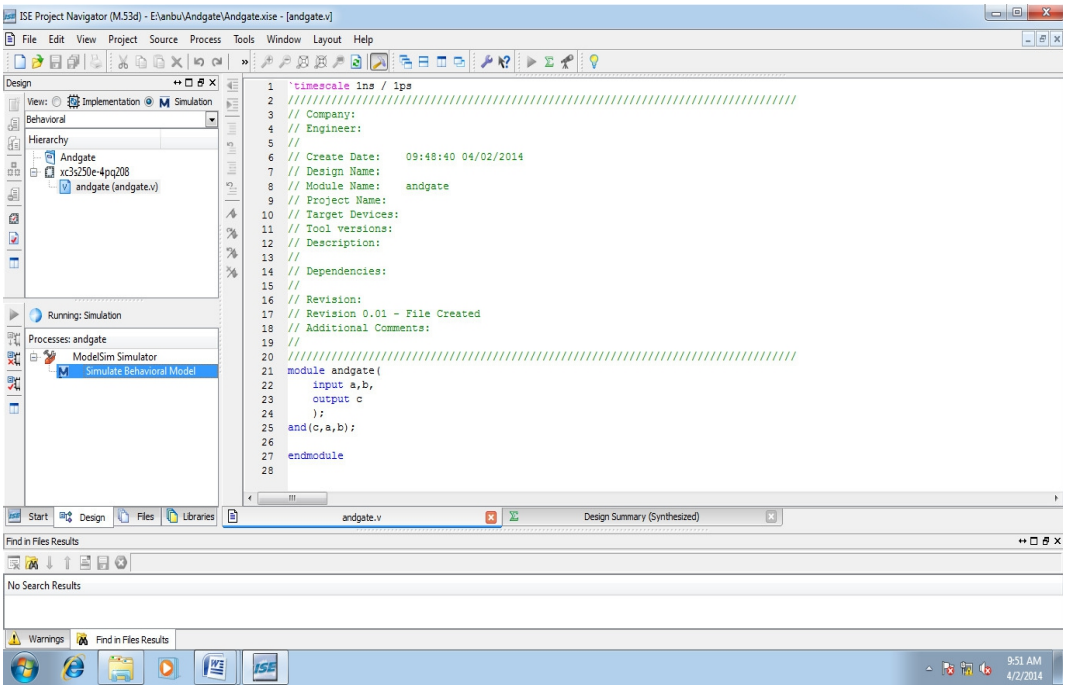


Figure 6.26: Simulate the program to get output wave form

11. Select your file in work area.

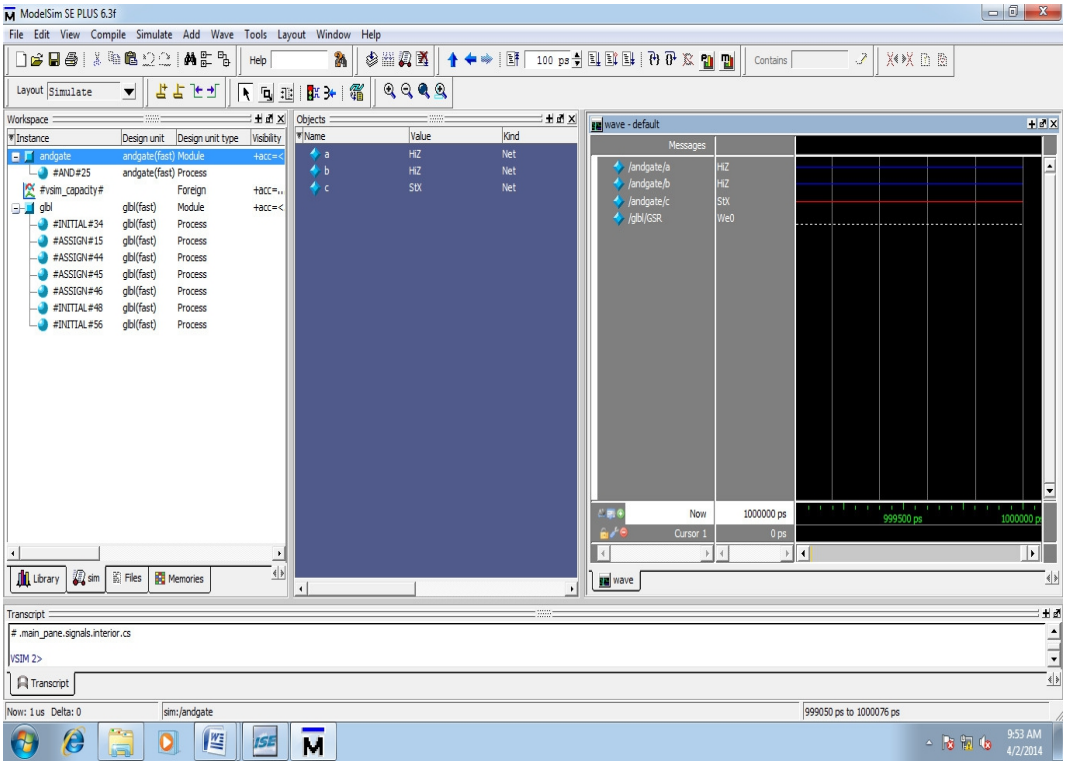


Figure 6.27: Select the file

12. In object window select input and output signal and provide input.

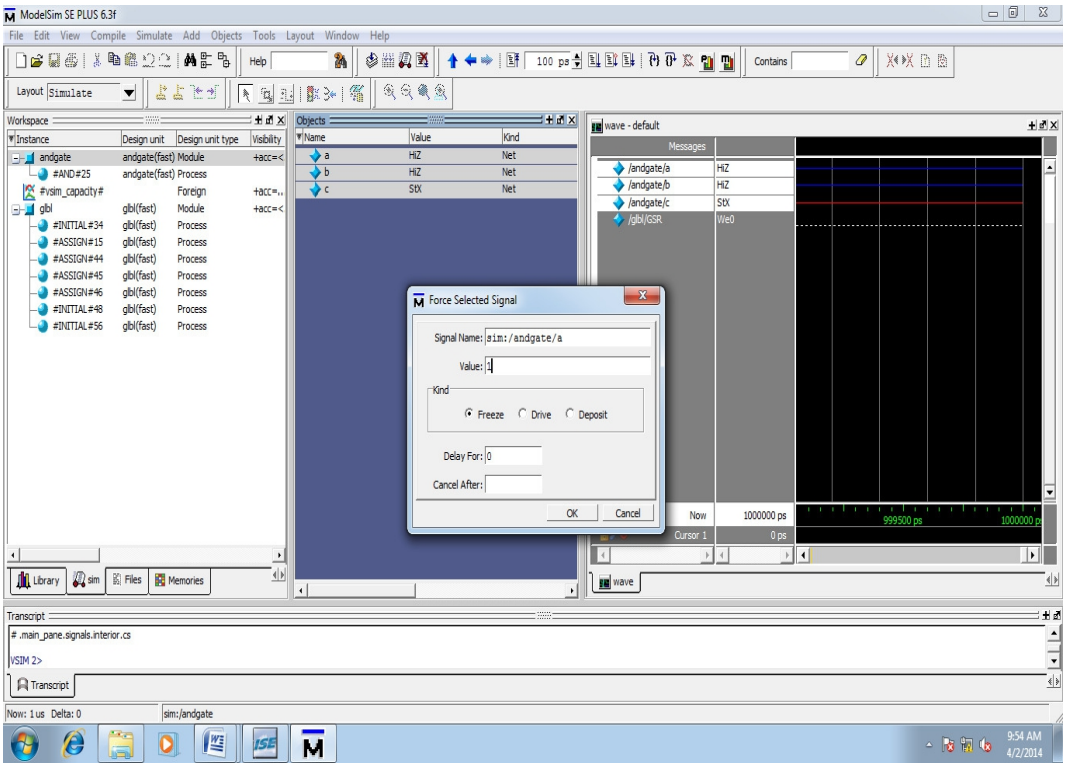


Figure 6.28: In object window select input and output signal and provide input.

13. In wave window, click run icon and you can see corresponding output.

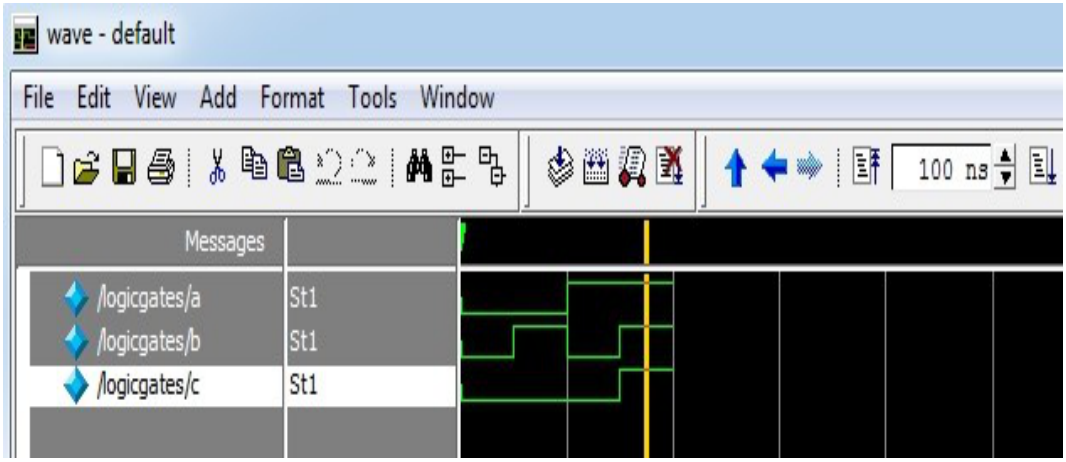


Figure 6.29: Run the program to get corresponding output

Chapter 7

RESULTS

7.1 32 bit ripple carry adder using reversible logic gates Results

The fig-7.1 below shows the RTL Schematic Diagram of 32-bit Ripple Carry adder reversible logic gates

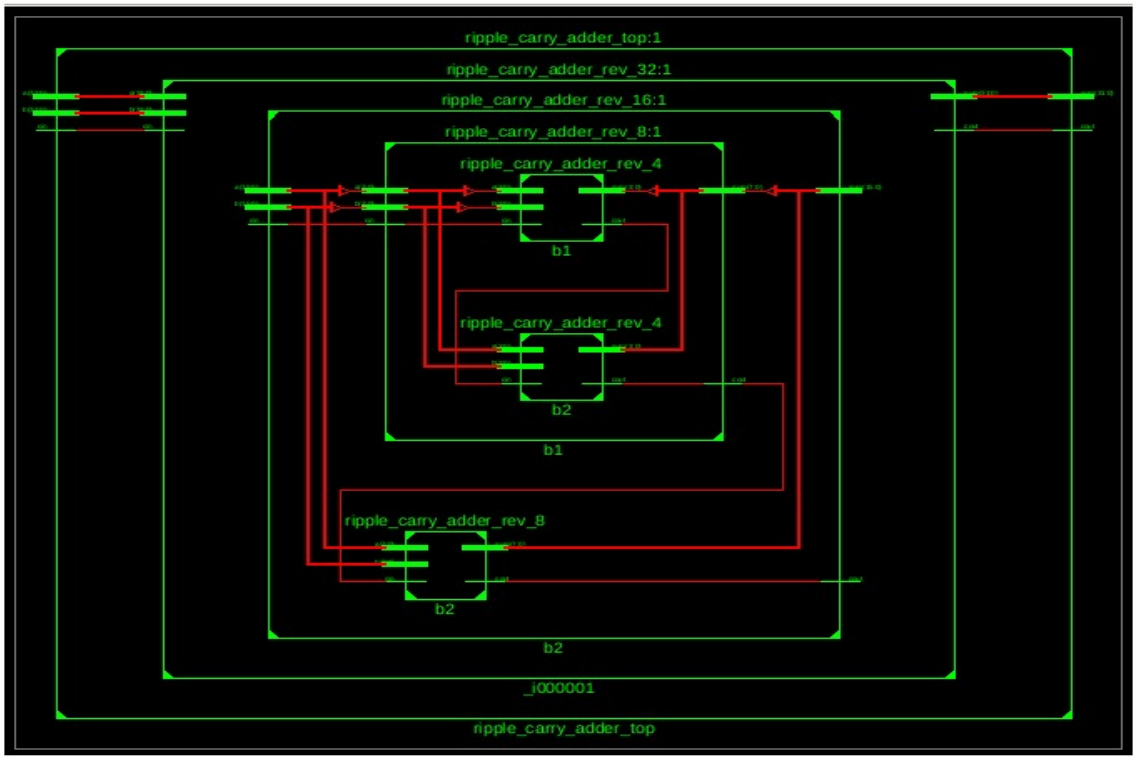


Figure 7.1: RTL Schematic Diagram of 32-bit ripple carry adder using reversible logic gates

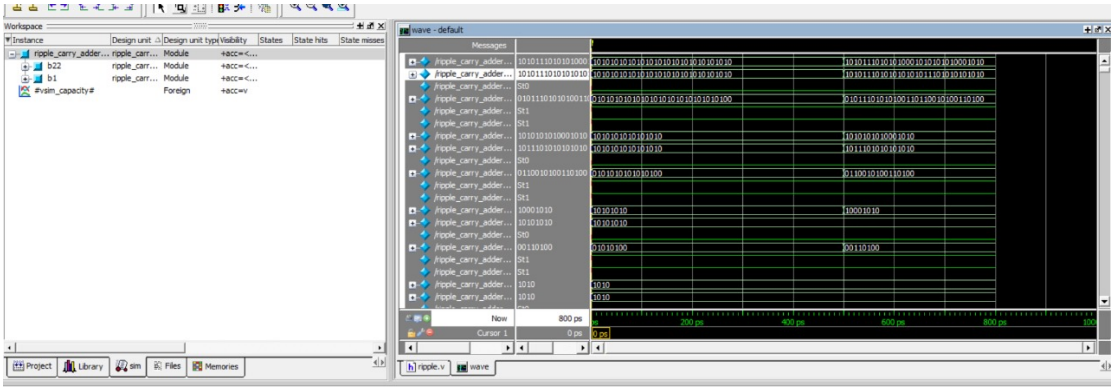


Figure 7.2: 32-bit Ripple Carry Adder using Reversible Logic Gates output

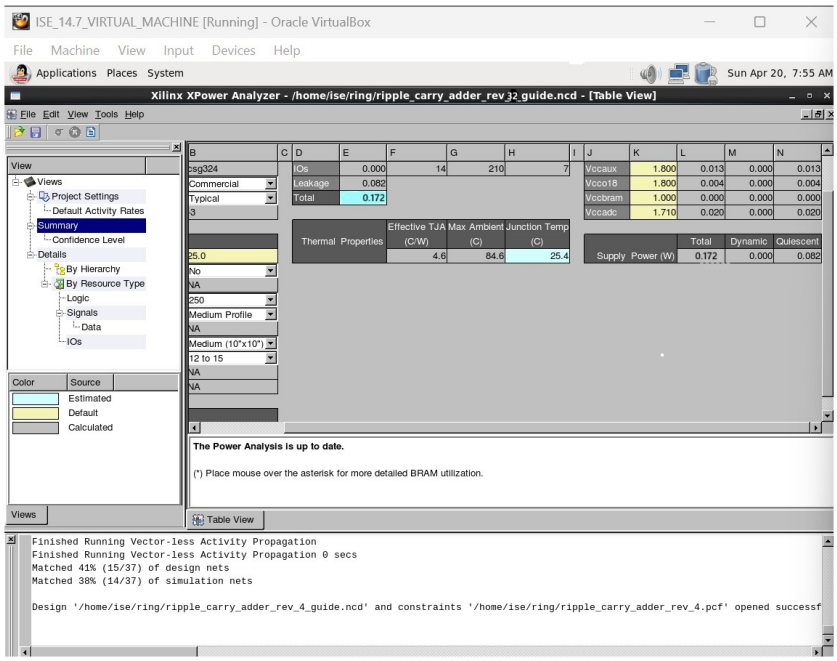


Figure 7.3: 32-bit Ripple Carry Adder using Reversible Logic Gates power

7.2 32 bit carry Increment adder using reversible logic gates Results

The fig-7.4 below shows the RTL Schematic Diagram of 32-bit Carry Increment Adder using reversible logic gates.

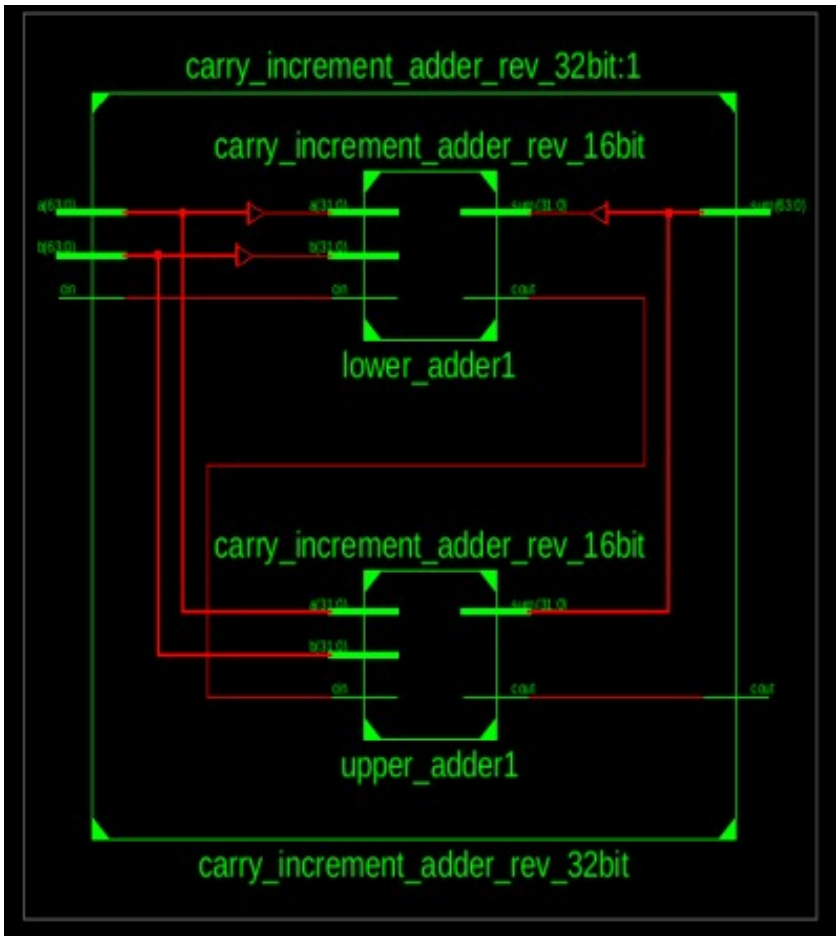


Figure 7.4: RTL Schematic Diagram of 32-bit carry Increment adder using reversible logic gates

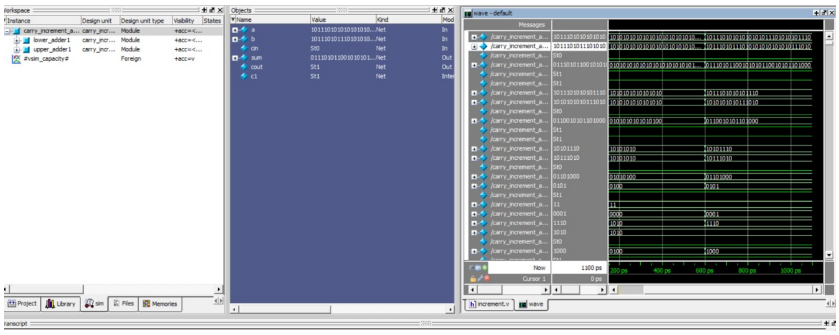


Figure 7.5: 32-bit Carry Increment Adder using Reversible Logic Gates output

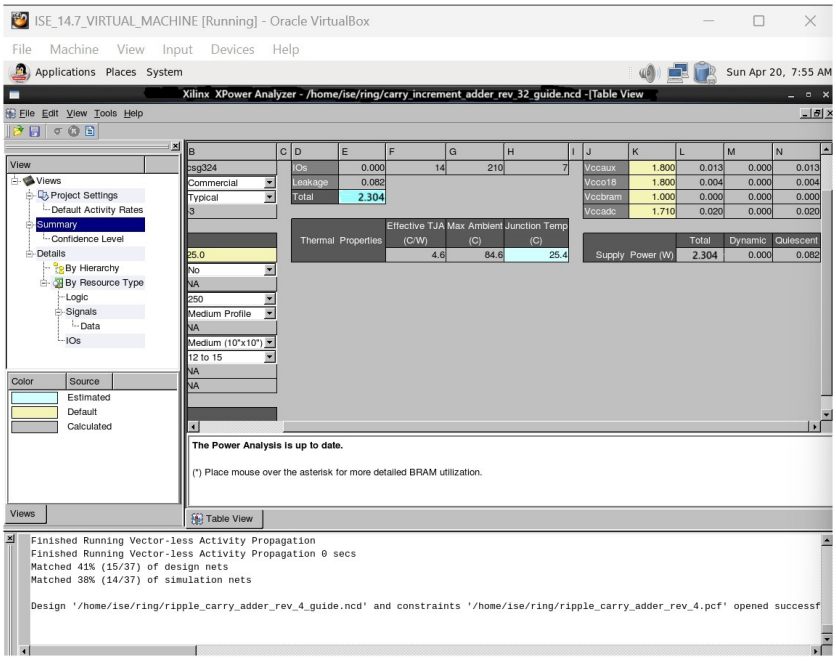


Figure 7.6: 32-bit Carry Increment Adder using Reversible Logic Gates power

7.3 32 bit carry Select adder using reversible logic gates Results

The fig-7.7 below shows the RTL Schematic Diagram of 32-bit Carry Select Adder using reversible logic gates.

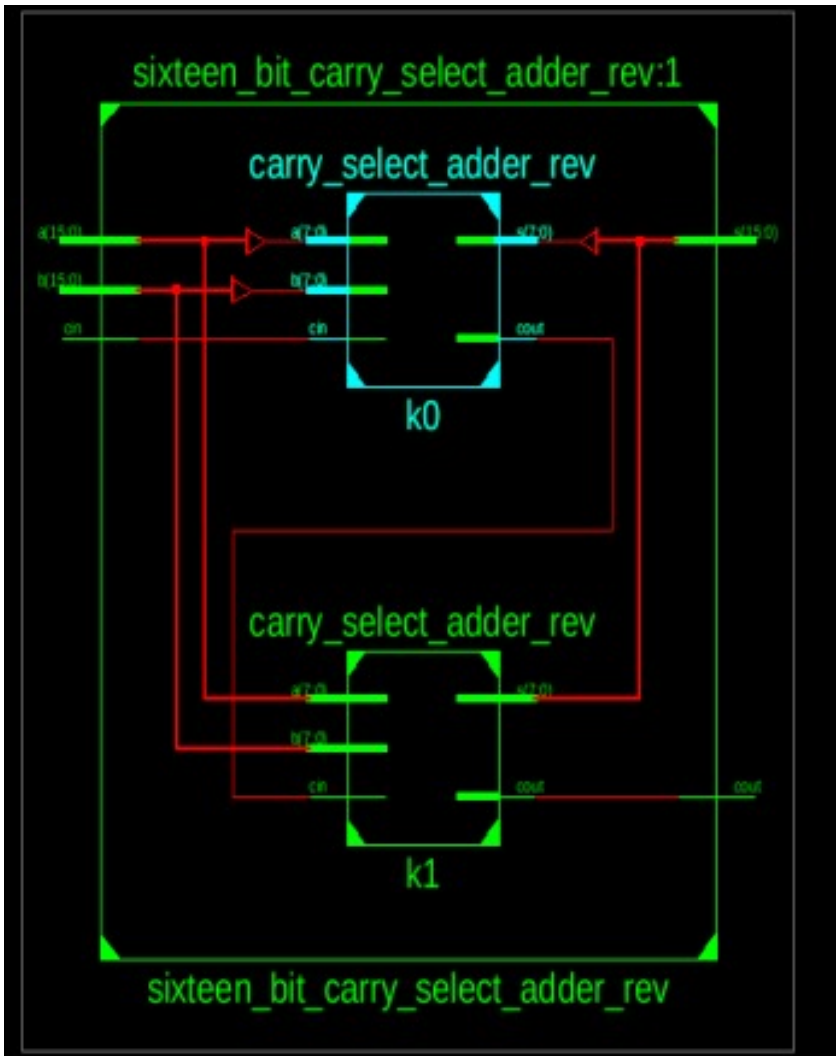


Figure 7.7: RTL Schematic Diagram of 32-bit carry Select adder using reversible logic gates

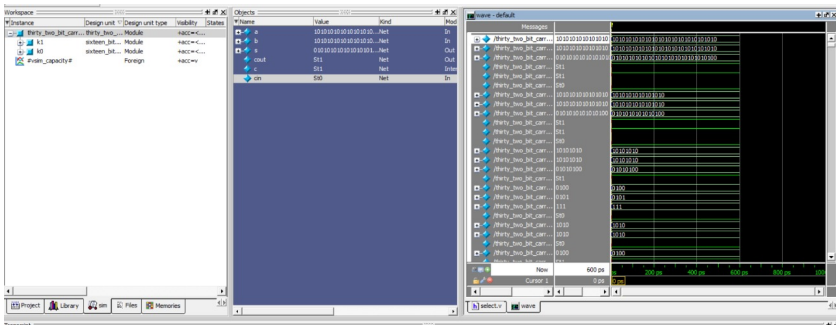


Figure 7.8: 32-bit Carry Select Adder using Reversible Logic Gates output

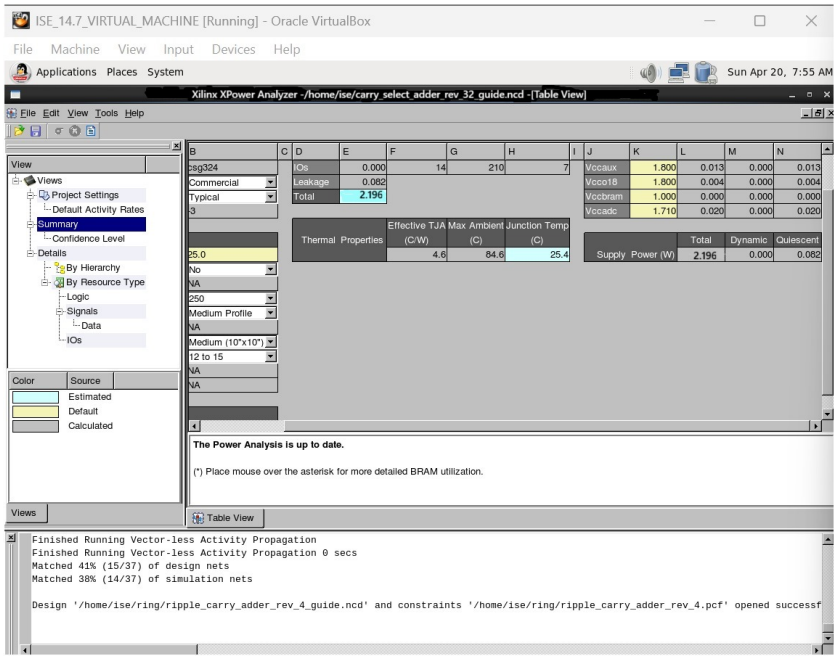


Figure 7.9: 32-bit Carry Select Adder using Reversible Logic Gates power

7.4 32 bit carry Skip adder using reversible logic gates Results

The fig-7.10 below shows the RTL Schematic Diagram of 32-bit Carry Skip Adder using reversible logic gates.

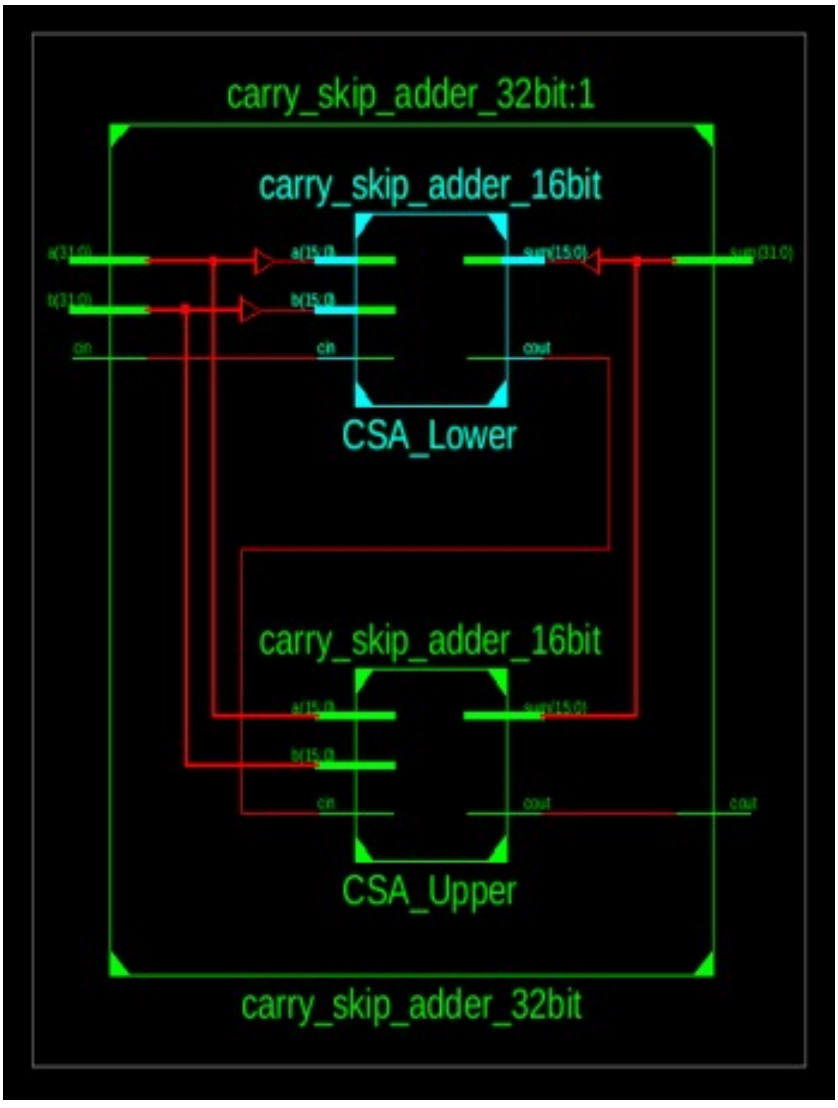


Figure 7.10: RTL Schematic Diagram of 32-bit carry Skip adder using reversible logic gates

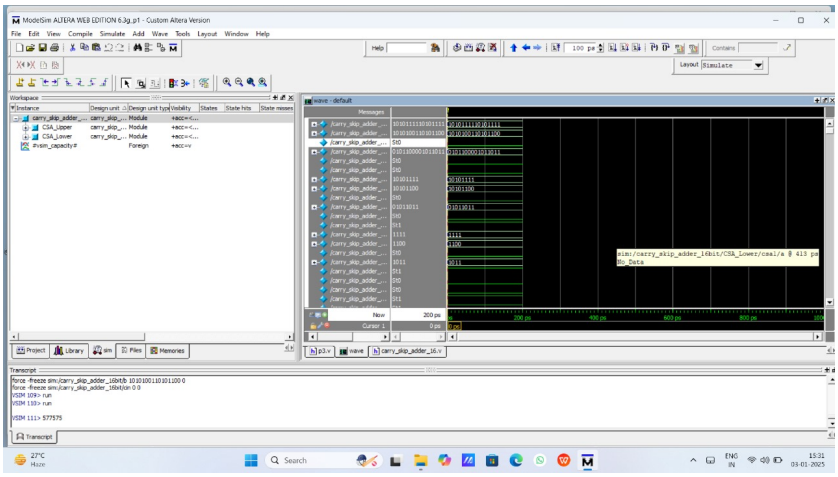


Figure 7.11: 32-bit Carry Skip Adder using Reversible Logic Gates output

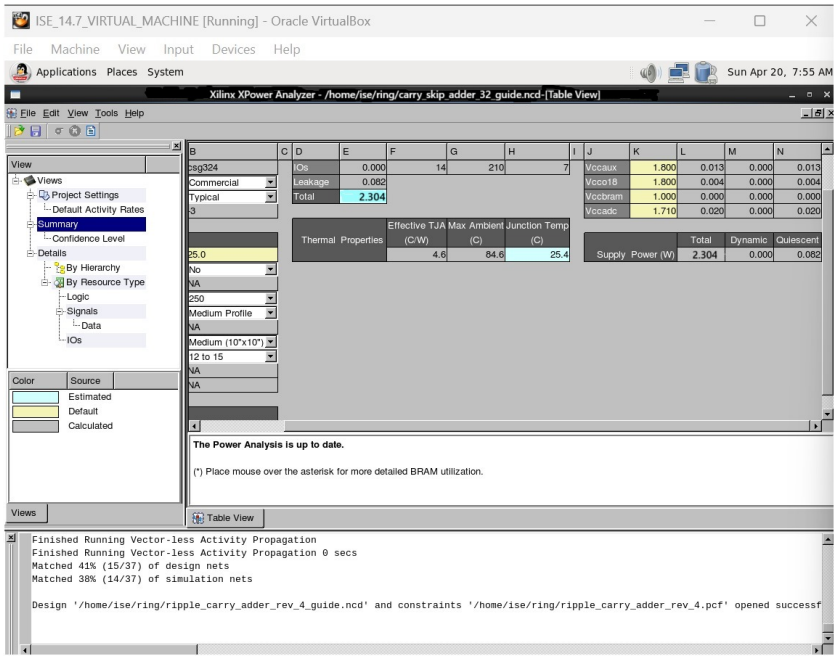


Figure 7.12: 32-bit Carry Skip Adder using Reversible Logic Gates power

7.5 32 bit carry Save adder using reversible logic gates Results

The figures below shows the RTL Schematic Diagram of 32-bit Carry Save Adder using reversible logic gates.

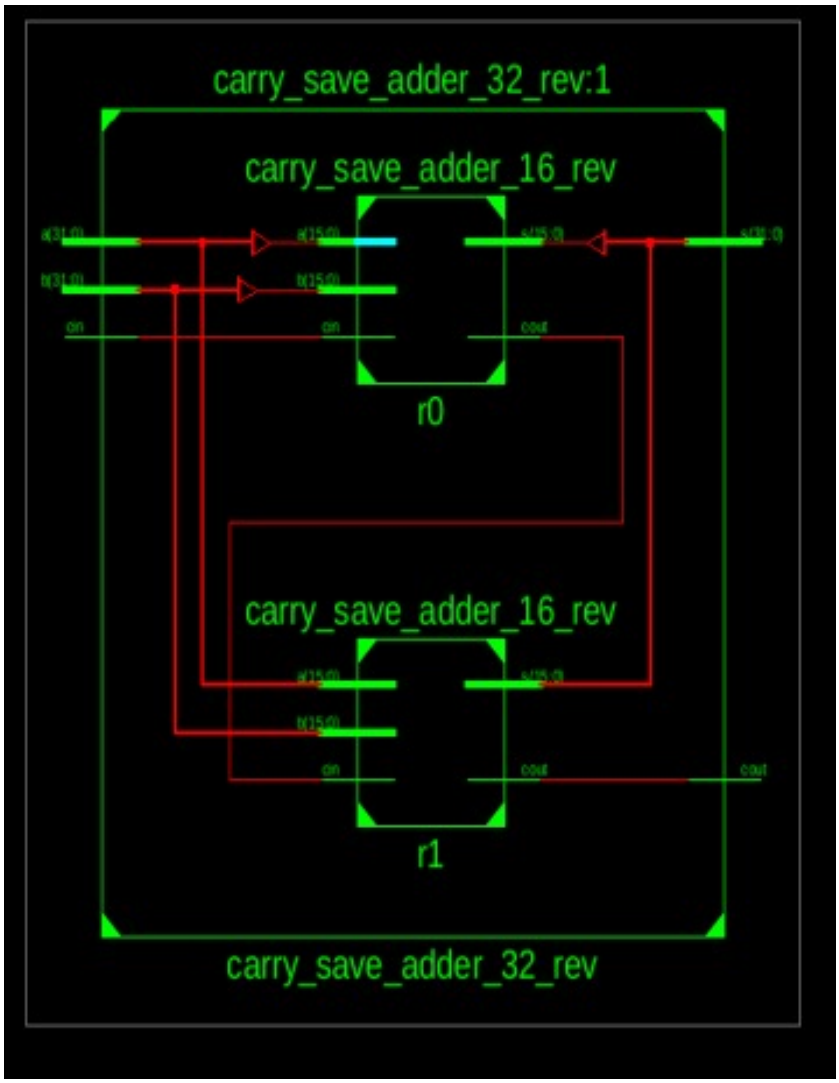


Figure 7.13: RTL Schematic Diagram of 32-bit carry Save adder using reversible logic gates

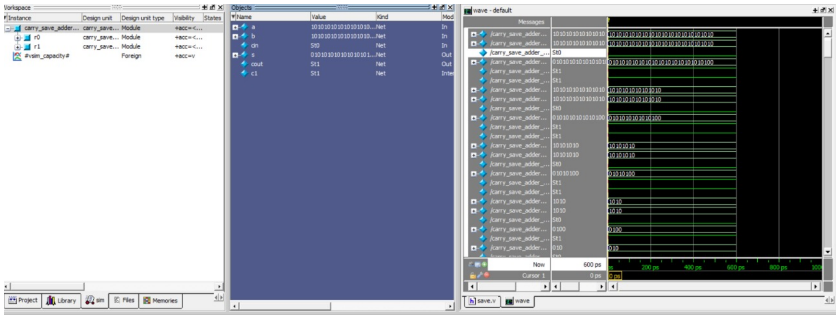


Figure 7.14: 32-bit Carry Save Adder using Reversible Logic Gates output

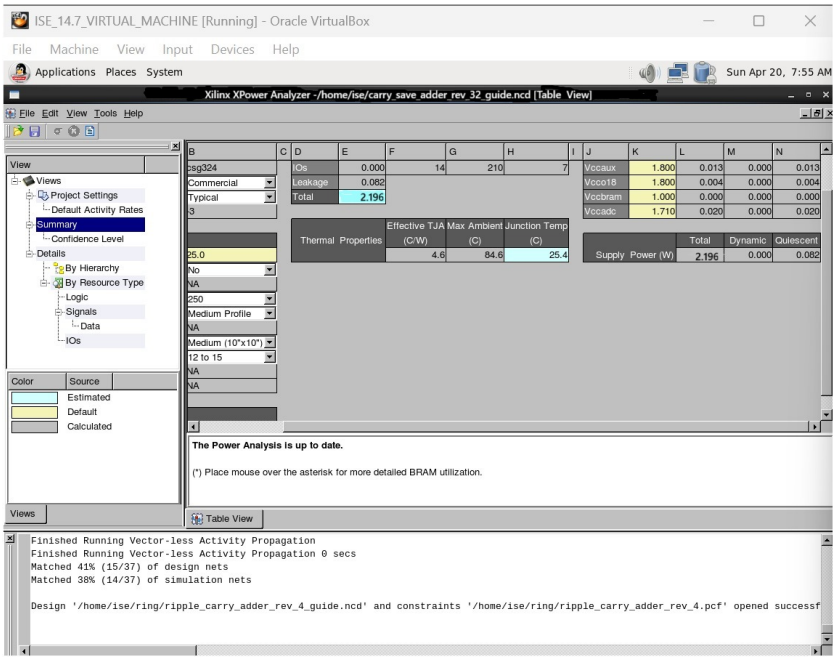


Figure 7.15: 32-bit Carry Save Adder using Reversible Logic Gates power

7.6 Comparative Analysis of Reversible Logic gates

The table-7.1 shows power, delay, area of ripple carry adder, carry save adder, carry increment adder,carry select adder,carry skip adder by using basic logic gates

S.NO	ADDERS	NUMBER OF LUT'S	DELAY(ns)	POWER(mw)
Ripple Carry Adder				
1.	8-Bit	16	16.062	0.092
2.	16-Bit	32	26.686	0.123
3.	32-Bit	64	48.086	0.182
Carry Increment Adder				
1.	8-Bit	20	14.426	0.726
2.	16-Bit	37	32.422	1.283
3.	32-Bit	72	64.028	2.323
Carry Select Adder				
1.	8-Bit	21	15.473	0.576
2.	16-Bit	38	31.409	1.192
3.	32-Bit	76	59.152	2.341
Carry Skip Adder				
1.	8-Bit	16	11.570	0.682
2.	16-Bit	34	23.720	1.172
3.	32-Bit	56	48.028	2.126
Carry Save Adder				
1.	8-Bit	19	21.473	0.426
2.	16-Bit	33	38.570	1.231
3.	32-Bit	65	71.030	2.409

Table 7.1: Logic Gates

The table-7.2 shows power, delay, area of ripple carry adder, carry save adder, carry increment adder,carry select adder,carry skip adder by using reversible logic gates

The fig.7.16 shows ripple carry adder(RCA) results comparison of 8-bit,16-bit and 32-bit using basic logic gates and reversible logic gates.

The fig.7.17 shows carry increment adder(CIA) results comparison of 8-bit,16-bit and 32-bit using basic logic gates and reversible logic gates.

S.NO	ADDERS	NUMBER OF LUT'S	DELAY(ns)	POWER(mw)
Ripple Carry Adder				
1.	8-Bit	14	14.722	0.069
2.	16-Bit	28	24.686	0.116
3.	32-Bit	58	41.086	0.176
Carry Increment Adder				
1.	8-Bit	17	12.421	0.576
2.	16-Bit	35	28.422	1.152
3.	32-Bit	68	56.042	2.304
Carry Select Adder				
1.	8-Bit	19	12.363	0.476
2.	16-Bit	35	26.582	1.052
3.	32-Bit	68	48.421	2.196
Carry Skip Adder				
1.	8-Bit	14	9.08	0.576
2.	16-Bit	23	17.099	1.076
3.	32-Bit	47	38.562	1.982
Carry Save Adder				
1.	8-Bit	16	19.501	0.328
2.	16-Bit	29	31.473	1.152
3.	32-Bit	56	64.409	2.304

Table 7.2: Reversible Logic Gates

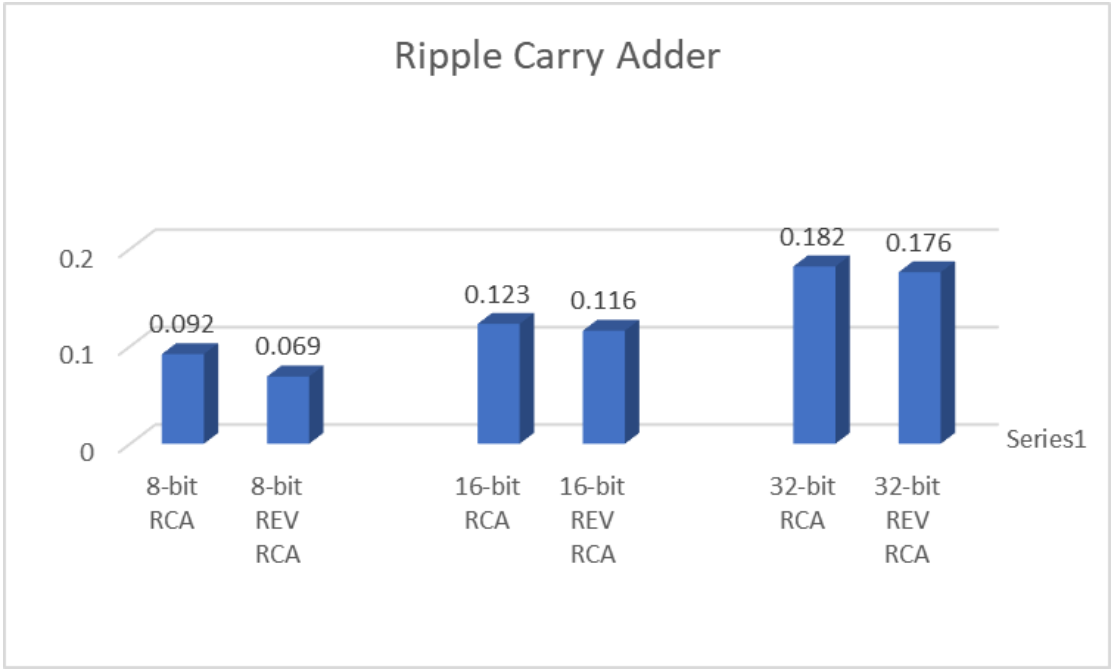


Figure 7.16: Ripple Carry Adder Comparison

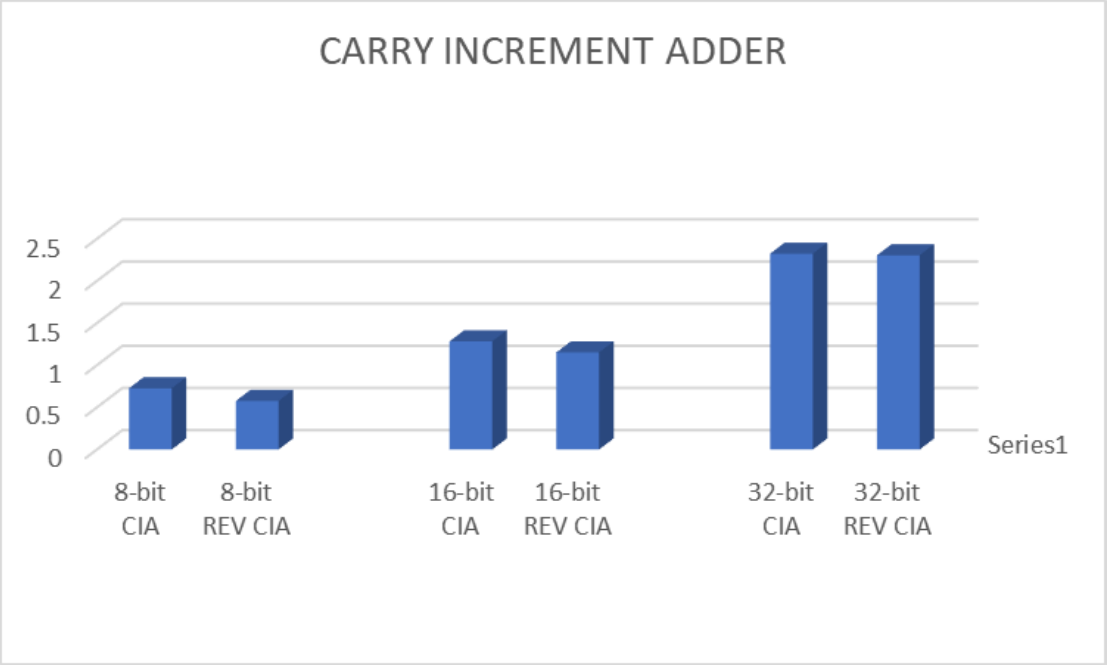


Figure 7.17: Carry Increment Adder Comparison

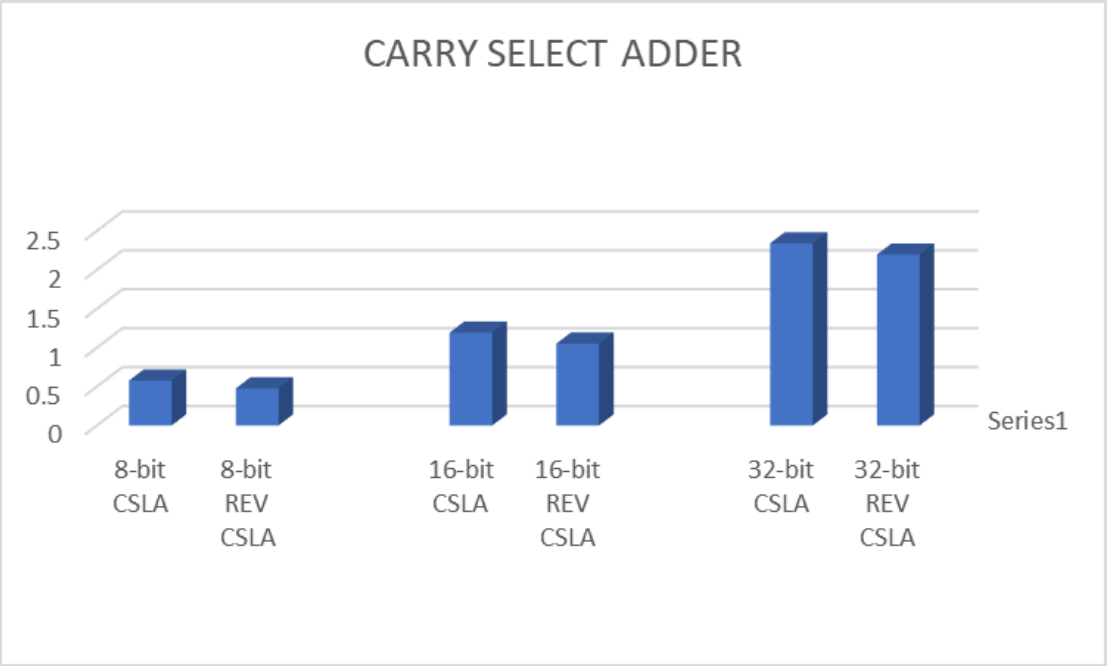


Figure 7.18: Carry Select Adder Comparison

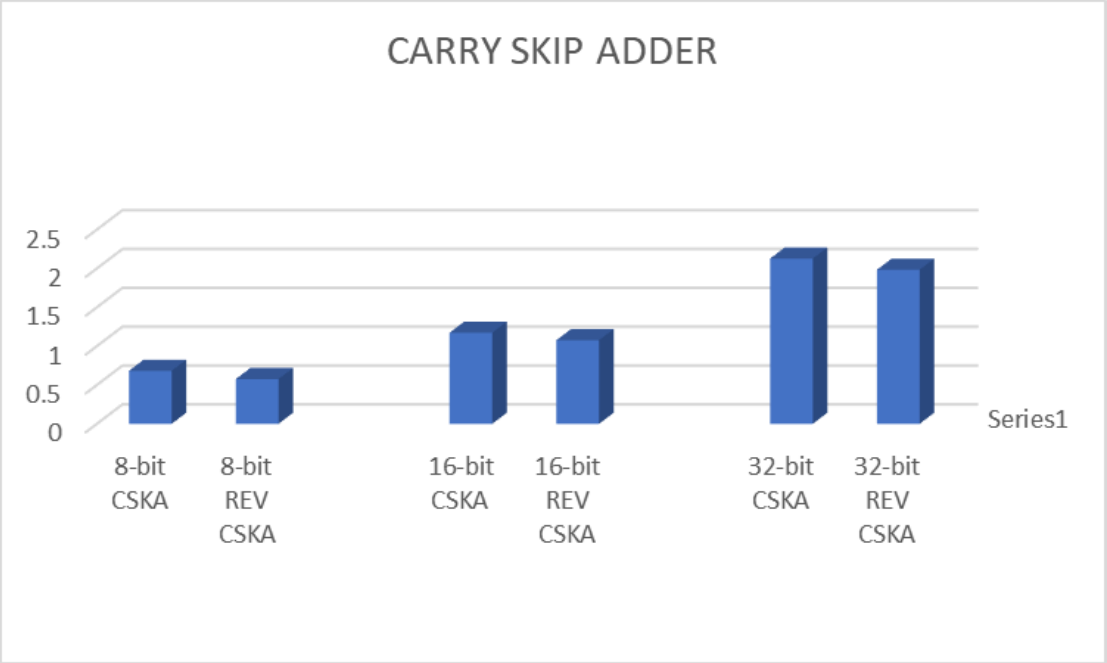


Figure 7.19: Carry Skip Adder Comparison

The fig.7.18 shows carry select adder(CSLA) results comparison of 8-bit,16-bit and 32-bit using basic logic gates and reversible logic gates.

The fig.7.19 shows carry skip adder(CSKA) results comparison of 8-bit,16-bit and 32-bit using basic logic gates and reversible logic gates.

The fig.7.20 shows carry save adder(CSA) results comparison of 8-bit,16-bit and 32-bit using basic logic gates and reversible logic gates.

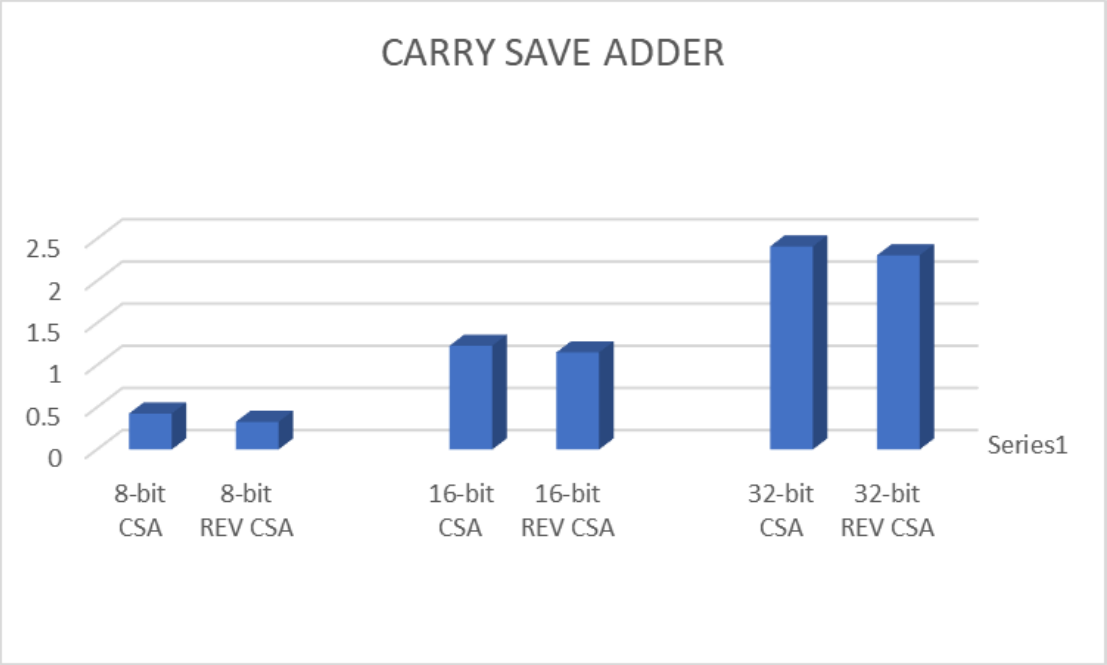


Figure 7.20: Carry Save Adder Comparison

Chapter 8

CONCLUSION

In this work, the design and implementation of reversible logic gates using different adder architectures have been successfully explored. Reversible logic, which plays a crucial role in low-power and quantum computing, was implemented to develop energy-efficient arithmetic circuits, particularly adders. By analyzing various reversible adders such as the Ripple Carry Adder, Carry Increment Adder,Carry Select Adder,Carry Skip Adder and Carry Save Adder, the study highlights their respective trade-offs in terms of gate count, garbage outputs, quantum cost, and overall efficiency.

The results demonstrate that the choice of adder greatly influences the performance and resource utilization in reversible logic circuits. Among the designs analyzed, [mention the best-performing adder if you have results], offered the most optimal balance between speed and minimal resource overhead. The study reinforces the importance of selecting suitable reversible logic designs for future technologies in quantum computing, nanotechnology, and low-power VLSI systems.

Further improvements can be made by optimizing existing gate designs or introducing novel reversible gates that reduce quantum cost and garbage outputs even further. This project lays a strong foundation for continued research and development in the domain of reversible computing.

8.1 Future Scope

The design and implementation of reversible logic gates using various adder architectures open up several promising avenues for future research and development. As the demand for low-power, high-performance computing systems continues to grow, the relevance of reversible logic in both classical and quantum computing becomes increasingly significant. The following points highlight key areas for future exploration:
1.Integration with Quantum Computing
2.Optimization of Reversible Circuits
3.Design of Complex Arithmetic Circuits
4.Nanoelectronics and DNA Computing
5.Fault Tolerance and Error Correction
6.VLSI Implementation and FPGA Prototyping

References

[1] ”High-Speed Vedic Mathematics: Multiplication Made Faster” by Tirthaji Maharaj and V. S. Agrawala.

[2] ”Vedic Mathematics for Schools, Book 1” by James T. Glover.

[3] ”Vedic Mathematics Made Easy” by Dhaval Bathia.

[4] ”Vedic Mathematics” by Sri Bharati Krsna Tirthaji.

[5] Noorallahzadeh, Mojtaba, et al. ”A new design of parity preserving reversible Vedic multiplier targeting emerging quantum circuits.” International Journal of Numerical Modelling: Electronic Networks, Devices and Fields (2023): e3089.

[6] Verma, Aishita, Anum Khan, and Subodh Wairya. ”Design and Analysis of Efficient Vedic Multiplier for Fast Computing Applications.” International Journal of Computing and Digital Systems 13.1 (2023): 190-201.

[7] Nandhini, V., and K. Sambath. ”VLSI implementation of multiplier design using reversible logic gate.” Analog Integrated Circuits and Signal Processing (2023): 1-8.

[8] G.R.GOkhale,P.D.Bahirgonde”Design of vedic multiplier using area efficient carry select adder (IEEE) in the year 2015.

[9] Srinithraavi,T.satyanarayana.”Implementation of high-speed hybrid Carry Select Adder using Binary to Excess-1 Converter”(IEEE-2022)

[10] Shikha Singh,B.Yagnesh”Low Power Carry Select Adder using FinFET Technology(IEEE-2022).

[11] Kaja Naga Venkata Akhil,P.Sathish Kumar”Implementation of Low Power N-bit Hybrid Carry Select Adder with Sum - Carry Selection(IEEE-2022)”

[12] Nagaraj, S; Thyagarajan, K; Srihari, D; Gopi, K”Design and analysis of Wallace tree multiplier for CMOS and CPL logic in 2018 (IEEE).

[13] Nagaraj, S; Reddy, GM Sreerama; Mastani, S Aruna"Analysis of different Adders using CMOS, CPL and DPL logic in 2017(IEEE).

[14] Y. Harshavardhan SVCET, RVS NAGAR,Dept. of ECE,Chittoor, AP; Nagaraj, S.; Jaahnavi, S.; Reddy, T. Manasa"Analysis of 8-bit Vedic Multiplier using high speed CLA Adder in 2020(IEEE).

[15] Yaswanth, D; Nagaraj, S; Vijeth, R Vishnu"Design and analysis of high speed and low area vedic multiplier using carry select adder in 2020(IEEE).

[16] Nagaraj, S; Krishna, B Vamsi; Chakradhar, Botta; Sarkar, Debanjan; "Comparison of 32-bit ALU for Reversible Logic and Irreversible Logic in 2021(IEEE).

[17] Haripriya, A; Nagaraj, S; Samanth, C"Design and Analysis of 16-bit Vedic Multiplier using RCA and CSLA in 2023(IEEE).

[18] Lavanya, A; Nagaraj, S; Lekhya, M"Design and Implementation of Vedic Multiplier using Carry Increment Adder in 2023(IEEE).