IMPLEMENTATION OF 32-BIT COMBINED INTEGER AND FLOATING POINT MULTIPLIER USING REVERSIBLE LOGIC

A Project Report submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

By

CHOWDADA BHARGAVI (20JG1A0419)

KUDDIGANA GURUTEJITHA (20JG1A0450)

KARANAM DEEKSHITHA (20JG1A0440)

AKULA SRAVANI (20JG1A0401)

Under the Esteemed guidance of

Ms. B. DIVYA SATHI

Assistant professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

(Affiliated to Jawaharlal Nehru Technological University Kakinada)

MADHURAWADA, VISAKHAPATNAM-48

(2020-2024)

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project titled **"IMPLEMENTTION OF 32-BIT COMBINED INTEGER AND FLOATING POINT MULTIPLIER USING REVERSIBLE LOGIC"** is a bonafide work of the following IV B-Tech students in the Department of Electronics and Communication Engineering during the academic year 2023-2024, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology of Jawaharlal Nehru Technological University, Kakinada.

CH.BHARGAVI (20JG1A0419) K.GURUTEJITHA (20JG1A0450)
K.DEEKSHITHA (20JG1A0440) A.SRAVANI (20JG1A0401)

Project Guide **Head of the Department**

(Ms. B. DIVYA SATHI) **(Dr.P.M.K PRASAD)**

External Examiner

ACKNOWLEDGEMENT

We sincerely thank our project guide **Ms.B.DIVYA SATHI**, Asst. Professor, for her guidance and constant encouragement to us at every stage and aspect by including the spirit of understanding and support in carrying out our main project at college.

We would like to express sincere thanks to our Head of the Department of Electronics and Communication Engineering **Dr.P.M.K PRASAD** for his valuable suggestions and constant motivation that greatly helped me in completing the Seminar work successfully.

We express sincere thanks to our Vice Principal, Professor **Dr.G.Sudheer,** for his encouragement and co-operation in completion of our project.

We wish to express our deep sense of our gratitude to our Principal , Professor **Dr.R.K Goswami**, for giving us the opportunity to carry out the project work successfully.

We would like to express our gratitude towards our parents & members of Gayatri Vidya Parishad College of Engineering For Women for their kind co-operation and encouragement which helped us in completion of our project.

Chowdada Bhargavi (20JG1A0419)
Karanam Deekshitha (20JG1A0440)
Kuddigana GuruTejitha (20JG1A0450)
Akula Sravani (20JG1A0401)

VISION AND MISSION

Vision of the Institute

To emerge as an acclaimed center of learning that provides value based technical education for the holistic development of students.

Mission of the Institute

- Undertaking activities that provide value based knowledge in science, engineering & technology.
- Provide opportunities for learning through Industry – Institute interaction on the state – of – the – art technologies.
- Create collaborative environment for research, Innovation and entrepreneurship to flourish.
- Promote activities that bring in a sense of social responsibility.

Vision of the Department

Produce competitive engineers instilled with ethical and social responsibilities to deal with the technological challenges in the field of Electronics & Communication Engineering.

Mission of the Department

- Facilitation a value based educational environment that provides updated technical knowledge.
- Provide opportunities for developing creative, innovation and leadership skills.
- Imbue technological and managerial capabilities for a successful career and lifelong learning.

TABLE OF CONTENTS

LIST OF FIGURES:

ABSTRACT

The delay required for operations in electronic products needs to be minimized to ensure reliable performance. Design of multipliers with lesser delay and increased throughput will lead the systems to operate with high speed. This process involves synthesizing circuits with optimized gate configurations to minimize the overall count. The design of traditional digital circuits relies on fundamental computational techniques that entail gate delays during logic operations. This design mitigates such delay in computations by minimizing them. By exploiting the inherent reversibility of the logic gates, the design aims to mitigate energy dissipation and enhance computational efficiency. Leveraging reversible logic not only facilitates the multiplication process but also enables the retrieval of original operands from the product, thus ensuring information preservation.

Floating point multiplication is a common operation in advance Digital Signal Processing (DSP) application. This design proposes a 32-bit Reversible Floating Point Multiplier using reversible gates i.e; Feynman, Peres Gates. Throughout the design of this multiplier circuit, a crucial aspect to consider is its compatibility with the environment and intended application. Furthermore, the circuit's feasibility is heavily influenced by design variation, which can cause notable deviations in the output of digital circuits that are measurable and predictable. In this design proposed multiplier performs multiplications with increased throughput by using minimal power. Implementation of the multiplier circuit is conducted using Xilinx technology, emphasizing practical applicability and seamless integration into existing digital systems. By prioritizing efficiency, throughput, and delay, this project aims to contribute to the advancement of digital signal processing capabilities in various domains. Ultimately, the development of a versatile and high speed multiplier unit capable of handling both integer and floating-point arithmetic operations concurrently will significantly enhance the efficiency and scalability of VLSI systems, catering to a wide range of computational requirements across various applications domains.

CHAPTER 1

INTRODUCTION

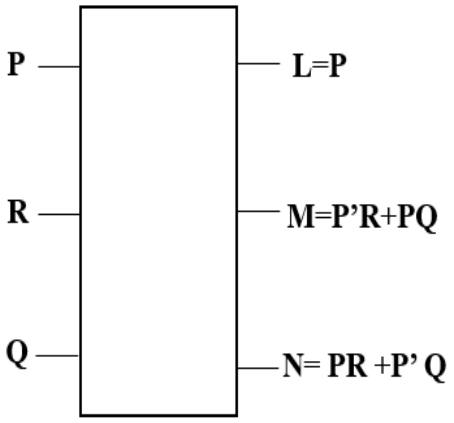
1.1 Introduction

VLSI (Very Large Scale Integration) is essential because it enables the creation of complex electronic systems by integrating millions of transistors onto a single chip. VLSI has become a cornerstone of modern technology, impacting various industries and enhancing our daily lives through increased computational power, connectivity, efficiency, and functionality. VLSI is a critical component of the semiconductor industry, which encompasses the design, manufacturing, and sale of integrated circuits. The semiconductor market has a substantial impact on the global economy, and VLSI plays a vital role in driving innovation and progress within this industry. This technology is the foundation of modern electronics, powering everything from smartphones and computers to medical devices and automotive systems. In order to meet the demand in high computational applications digital systems are designed using different logic's such as dynamic and static circuits. Choosing an appropriate logic can lead to high performance and low power designs. In VLSI systems design, reversible Logic has gained importance for Reducing power.

Basic reversible gates are used to build sequential circuits D Flip flop, Latch and RAM cell are developed. Heat is an important issue in VLSI circuits. But the reversible logic gives zero amount of heat dissipation. So, its an important role in nanotechnology, less energy complementary metal oxide semiconductor [CMOS] designs etc. It has been realized that quantum computing is one of the latest technologies using reversible logic gates. It is contemplating that accumulation extension of transistor density, energy desolation will command their limits in ordinary technologies. In ordinary area or range during the logic influence bits of orientation is erased resulting gratification of power in powerful amount. In the terms of reversible logic results are not lost.This minimize the delays but at the amount of little hardware. So we can use reversible logic technology for decreasing the energy dispersion, heatwave dissipation, increasing rapidness etc. So it is used to maximize the speed and reducing energy consumption. In this, we can describe following reversible logic gates like fredkin, peres, Feynmen and toffoli gate etc.

1.Fredkin Gate:

Fredkin is a reversible logic gate that swaps two bits if a third bit (the control bit) is set to 1 and leaves them unchanged otherwise. It's a fundamental building block in reversible computing, which aims to minimize energy dissipation by ensuring that every computation can be reversed without loss of information. It has three inputs and three outputs. Let's denote the inputs as A, B, and C, and the outputs as A', B', and C'. If C is 1, A' becomes B, B' becomes A, and C' remains 1. If C is 0, then A' remains A, B' remains B, and C' remains 0.



FREDKIN GATE

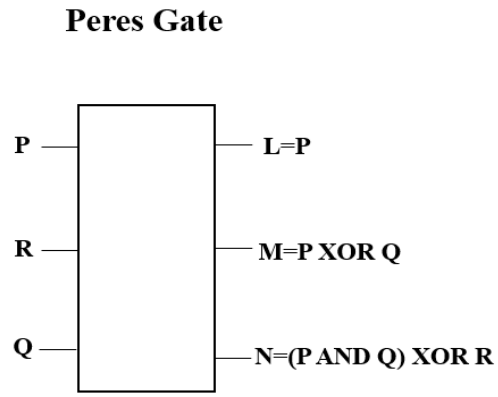
P	Q	R	L	M	N
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

TRUTH TABLE

Fig.1.1 Fredkin Gate and It's Truth Table

2.Peres Gate:

The Peres gate is a reversible logic gate operates on three qubits and is designed to implement various Boolean functions while preserving quantum information. The gate performs a controlled-NOT operation where the control qubit's state determines whether the target qubit's state is flipped or not. The Peres gate is particularly useful in quantum computing and quantum information processing due to its reversible nature, which ensures that no information is lost during computation.



Truth Table

P	Q	R	L	M	N
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

Fig.1.2 Peres Gate and It's Truth Table

3.Transparancy Symmetric Gate (TSG) :

The Transparency Symmetric (TS) gate is a reversible logic gate that preserves the value of its inputs when a control signal is set to a specific state. It's a symmetric gate, meaning it performs the same operation regardless of the order of its inputs.The TS gate has three inputs (A, B, and C) and three outputs (A', B', and C'). Its behavior can be described as follows:

If the control input (C) is 0, the gate behaves as a pass-through gate, meaning the outputs (A' and B') are equal to the inputs (A and B), and the control output (C') is also 0. If the control input (C) is 1, the gate swaps the values of its input bits A and B, while the control output (C') remains 1.

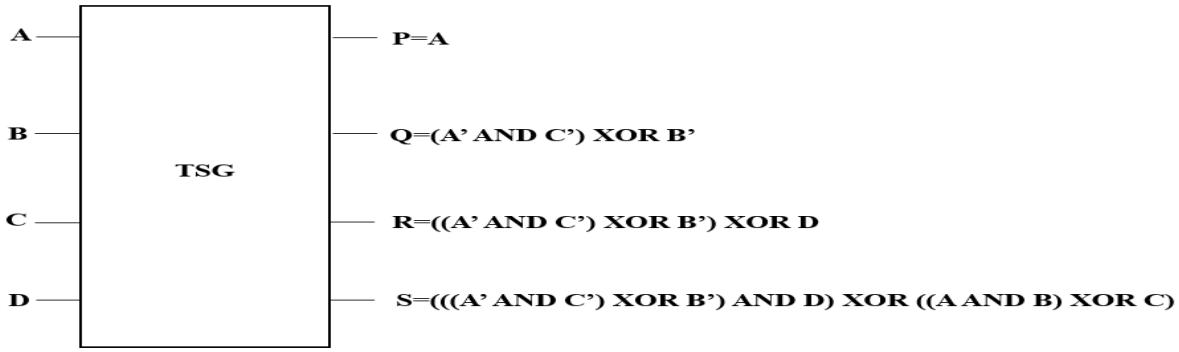
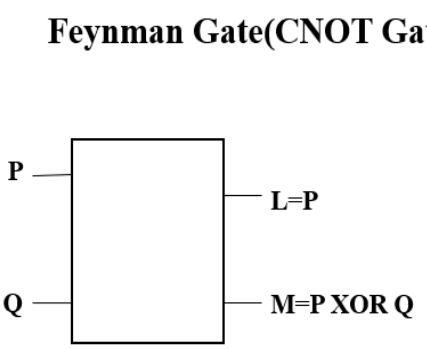


Fig .1.3 Transparancy Symmetry Gate

4. Feynman Gate(CNOT Gate) :

The Feynman gate, also known as the Controlled-Not (CNOT) gate, is a fundamental reversible logic gate commonly used in quantum computing and classical reversible computing. It's named after the physicist Richard Feynman, who made significant contributions to the field of quantum mechanics. The CNOT gate operates on two qubits, usually labeled as the control qubit (C) and the target qubit (T). Its behavior can be described as follows : If the control qubit (C) is in the state |0⟩, the target qubit (T) remains unchanged. If the control qubit (C) is in the state |1⟩, the target qubit (T) is flipped (NOT operation applied).



Truth Table

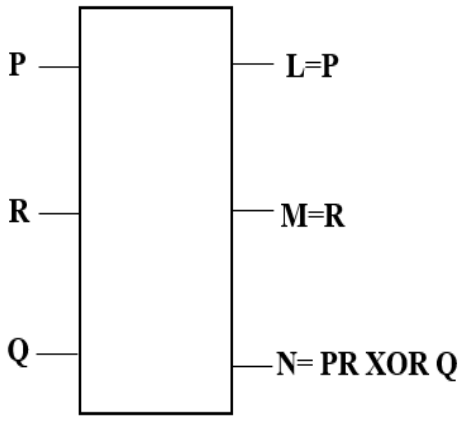
P	Q	L	M
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Fig .1.4 Feynman Gate and It's Truth table

5.TOFFOLI GATE:

The Toffoli gate, also known as the Controlled-Controlled-Not (CCNOT) gate or the Controlled-Controlled-X gate, is a reversible logic gate that performs a controlled NOT operation on a target qubit (T) depending on the states of two control qubits (C1 and C2). It's named after the physicist Tommaso Toffoli, who introduced it in the early 1980s.The Toffoli gate operates on three qubits, with two control qubits (C1 and C2) and one target qubit (T). Its behavior can be described as follows:

If both control qubits (C1 and C2) are in the state |1⟩, the NOT operation is applied to the target qubit (T), flipping its state.If one or both of the control qubits (C1 and C2) are in the state |0⟩, the target qubit (T) remains unchanged.



P	Q	R	L	M	N
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

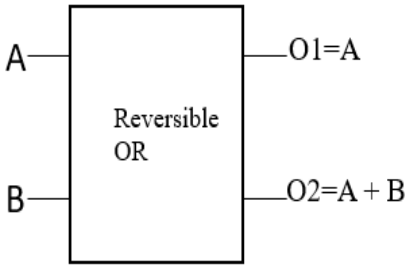
TOFFOLI GATE **TRUTH TABLE**

Fig 1.5 Toffoli Gate and It's Truth table

6. REVERSIBLE OR GATE:

A reversible OR gate is a logic gate that performs an OR operation, but unlike traditional OR gates, it's designed to be reversible, meaning it can also undo its operation. In reversible computing, the principle of reversibility is crucial because it allows for the conservation of information and energy.

In classical computing, OR gates take two inputs and produce an output that is true (1) if at least one of the inputs is true. However, in reversible computing, it's essential that every operation can be undone without loss of information.



A	B	O1	O2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	1

Reversible OR **TRUTH TABLE**

Fig 1.6 Reversible OR and It's Truth table

1.2 Aim:

Numerous multiplier designs in prior research have utilized logic gates, which inherently introduce gate delay. Some implementations have employed Toffoli and Fredkin Reversible logic, resulting in increased power consumption and heat dissipation. The aim of this project is to design and implement a 32-bit combined integer and floating point Multiplier utilizing reversible gates i.e, Peres Gate and Feynman with a focus on compatibility with advanced Digital Signal Processing (DSP) applications. The primary objective is to enhance the throughput of multiplication operations while minimizing the delay. Through careful design considerations, the project aims to achieve measurable and predictable performance deviations, ensuring reliability and accuracy in the output of digital circuits. Implementation will be carried out using Xilinx tools to provide a practical and efficient solution for high-performance Digital Signal Processing (DSP) environments.

CHAPTER 2

LITERATURE SURVEY

INTRODUCTION:

We have taken some reference papers and gone through them and acquired some knowledge about our proposed system and also we get to know the limitations (or drawbacks) of that specified design from each paper and that drawbacks are covered in another paper. In this way by analyzing many papers we are settled with one design which performs better than other designs that are specified in reference papers.

2.1 LITERATURE SURVEY -1

TITLE: Design and Comparison of Power, Area and Delay of 32-bit Reversible MAC unit.

AUTHORS: Pandimeena R, Annapoorani A, Kaviya T.D, Rajalakshimi A and Dhanagopal R

PUBLICATION: IEEE

YEAR OF PUBLICATION: 2020

2.1.1 OVERVIEW:

In this paper, reversible logic gates were specifically used to implement the MAC unit. The fundamental parameters such as delay, area, power of reversible MAC unit is measured and compared with conventional MAC unit. When comparing power and delay, Reversible implementation is efficient. Both addition and multiplication functions are done in MAC. Multiplier is the important component in the construction of MAC unit. Vedic Multiplier is used here to do the multiplication process. There are sixteen sutras in Vedic Mathematics. It is an efficient method inorder to produce the intermediate partial products and has the advantage of parallel processing. Urdhva Tiryagbhyam sutra uses Vertical Crosswise procedure to solve the problem efficiently. For the purpose of addition, 64-bit adders. It elaborates about basic reversible gates and also about Adders.

2.1.2 IMPLEMENTATION:

In this paper Vedic Multiplier is used as a multiplier of 32*32 bit of binary numbers. The algorithm is used here for the multiplication purpose which is Urdhva Tiryagbhyam. 2X2,

4X4, 8X8 and 16X16 bit multipliers are used to implement 32X32 bit multiplier. These conventional full adders are replaced with reversible HNG gates . It requires n number of HNG gates for n bit binary numbers. Initially the carry is zero. The carry output of each gate is rippled to adjacent gate as one of the inputs.

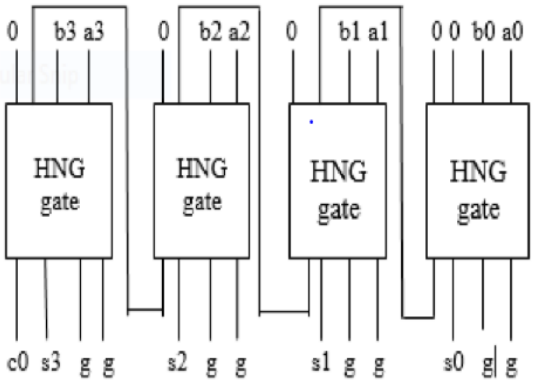


Fig. 2.1 Reversible Ripple Carry Adder

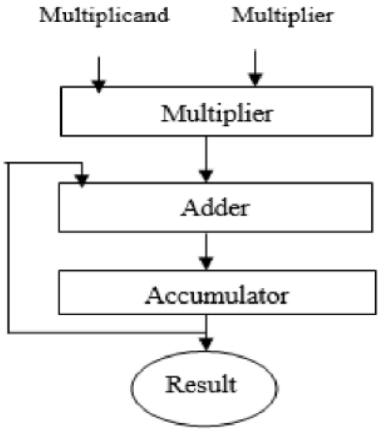


Fig. 2.2 MAC unit

Reversible 4X4 Vedic Multiplier used four 2X2 Vedic Multipliers, 2 Peres gates and has a Reversible OR gate to implement the design. The Carry Save Adder consists of (n-1) full adders and (n+1) half adders. In this the conventional half and full adders are replaced by Peres and HNG gate respectively. It is used to add three or more number of bits. Synthesis is done using Xilinx ISE 14.3 and implemented using Xilinx and Cadence RTL Power, delay and Area is done.

CONCLUSION:

In this paper, reversible logic gates were specifically used to implement the MAC unit. The fundamental parameters such as delay, area, power of reversible MAC unit is measured and compared with conventional MAC unit. When comparing power and delay, Reversible implementation is efficient.

2.2 LITERATURE SURVEY-2

TITLE: Reversible Gates: A Paradigm Shift in Computing

AUTHORS: Syed Farah Naz (Student Member, IEEE), Ambika Prasad Shah (Senior Member, IEEE)

PUBLICATION: IEEE

YEAR OF PUBLICATION: 2023

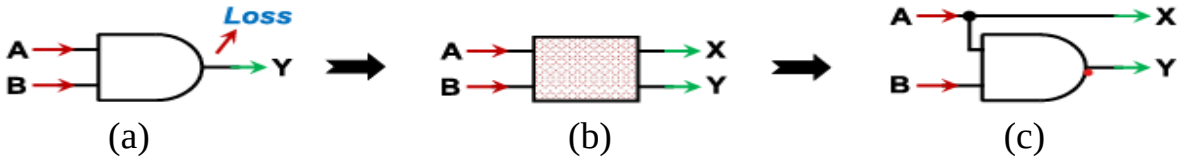
2.2.1 OVERVIEW:

The broad knowledge about reversible gates and their uses that we researched in this paper is compiled from several academic works and research publications. Future designers will be assisted by this survey in creating complicated computer systems with reversible gates. Low-power digital circuit design has taken into account reversible gates. There are numerous uses for reversible gates in cutting-edge technologies, including nanotechnology, quantum computing, optical computing, spacecraft, smart tags on inventory, low-power CMOS, biomolecular computation, communication, and FPGAs. Several important questions are still unanswered, which we are confident will be answered in the coming years. These questions can serve as prompts for readers to reflect on the content and potentially explore related research directions or areas of improvement. Hence, the researchers can explore innovative circuit architectures and logic gate configurations that offer improved performance, reduced quantum cost, lower complexity, and other desirable characteristics.

2.2.2 IMPLEMENTATION

Quantum circuit implementations utilize quantum gates to realize reversible operations. Quantum gates, such as the Controlled-NOT (CNOT) gate and Toffoli gate, inherently possess reversibility. These gates can be combined and controlled to construct larger reversible circuits. Quantum circuit implementations are particularly relevant in the context of quantum computing, where quantum gates are the building blocks for quantum algorithms. Another approach includes the reversible logic synthesis implementation which is a systematic approach to design reversible circuits. It involves the transformation of irreversible Boolean functions or logic circuits into their reversible equivalents. Various

algorithms and techniques are employed in reversible logic synthesis, such as using additional ancillary bits, exploiting symmetry properties, and applying gate-level optimizations. Reversible logic synthesis aims to minimize the number of gates, quantum cost, or other metrics while ensuring reversibility. These basic implementation methods can be combined and adapted to suit specific design requirements**.**



(a) (b) (c)

Fig. 2.3 Irreversible computation (a) 2-input AND gate (b) 2×2 module (c) 2×2 AND gate

FUTURE SCOPE:

Researchers can explore innovative circuit architectures and logic gate configurations that offer improved performance, reduced quantum cost, lower complexity, and other desirable characteristics. The review paper has discussed performance evaluation measures for reversible gates. Future work can delve deeper into optimization techniques for reversible logic circuits, aiming to minimize quantum cost, gate count, or circuit delay. Researchers can explore algorithms, heuristics, or machine-learning approaches to optimize gate level or circuit-level designs. They can investigate physical implementations of reversible gates using emerging technologies such as quantum computing, adiabatic circuits, or nanoscale/molecular electronics.

CONCLUSION:

The broad knowledge about reversible gates and their uses that we researched in this paper is compiled from several academic works and research publications. Future designers will be assisted by this survey in creating complicated computer systems with reversible gates. Low-power digital circuit design has taken into account reversible gates. There are numerous uses for reversible gates in cutting-edge technologies, including nanotechnology, quantum computing, optical computing, spacecraft, smart tags on inventory, low power CMOS, biomolecular computation, communication, and FPGAs.

2.3 LITERATURE SURVEY -3

TITLE: Design of Reversible Single Precision and Double Precision Floating Point Multiplier.

AUTHORS: Anekant Jain , Rakhee Jain , Jitendra Jain

PUBLICATION: IEEE

YEAR OF PUBLICATION: 2018

2.3.1 OVERVIEW:

A traditional circuit dissipates kTIn2 joules of energy each time one bit of information is lost. On the other hand, Reversible circuits prove its utility in low power computing as they minimize this information loss. We have been presented new designs of Reversible DPFP multiplier which uses our designs of 12X12, 5X12 and 5X5 Reversible multipliers. We have also presented our new design of Reversible SPFP multiplier which has shown a significant improvement in terms of Quantum cost and garbage output over the existing design. We have also concluded that in the method of multiplication by operand decomposition as the size of the groups in the decomposed operand increased, value of Quantum cost and garbage outputs tends to reduce but at the same time it increases the complexity involved. We have proposed two generalized methods of partial products generation with the comparison between them in terms of two important attributes of Reversible logic. Partial product generation of each multiplier has been done by these two methods in this paper. The quantum cost and garbage outputs of the entire SPFP and DPFP multiplier has been presented in this paper.

2.3.2 IMPLEMENTATION:

Multipliers in two stages namely Reversible partial products generation stage (RPPGS) and partial products addition stage (RPPAS) are implemented. Finally products are added from different multipliers by carefully shifting the terms according to which part of the first operand is multiplied to the part of the second operand. In each multiplier design two types of RPPGS, first by using all Peres gates and the second by using the combination of Toffoli and Peres gates. By comparing both the ways of partial products generation, it is found that the RPPGS with all peres gates exhibits less Quantum cost but more garbage outputs over the RPPGS with the Toffoli gates cascaded with Peres

gates. In RPPGS, partial products of single bit of one operand with the single bit of second operand are generated. For this work, Toffoli gate and Peres gate are the suitable candidate with their third input as '0'. Toffoli gate and Peres gate, both gives the product (as a AND gate) of the first two inputs at their third output when their third input is zero. In RPPAS, the addition of the various partial product terms is done with the carefully chosen RHAs, RFAs and R4:2Cs so as to get the minimized value of Quantum cost and garbage outputs. Four 12X12 multipliers for single precision floating point mantissa multiplier are used. Sixteen 12X12 multipliers, eight 5X12 multipliers and one 5X5 multiplier for single precision floating point mantissa multiplier is implemented.

2.4 LITERATURE SURVEY -4

TITLE: Reversible Logic Gates and its Performances.

AUTHORS: Anamika, Rockey Bhardwaj

PUBLICATION: IEEE

YEAR OF PUBLICATION: 2018

2.4.1 OVERVIEW:

Use of the reversible logic gates are increasing day by day but the scientists are still continue with their work on this to further more decreasing environmental decay growth . This paper explains about the types of the logic gates and its uses and how we can implement on these logic gates. This shows the BCD adder circuit and after use of reversible logic gates definitely its energy consumption and delays are less than its actual circuit design. In this reference it compares all types of logic gates and outputs the result of energy consumption and delays, in this Fredkin gate, Feynman gate, Sayem gate has low power consumption and Feynman gate and toffoli gate has less amount of delays. In this the output of the power radiation in a logical process deliver a blunt communication to the numeral of bits abolish during estimating. In this the better extended function of reversible connection lies in quantum computers.It has functions in different research operations such as quantum computing, nano technology, low power CMOS design, DNA computing.

Name Of Gate	Input Given	Output Response	Energy Consumption[mw]	Delay [ns]
Feymen Gate	3	3	18	7.465
Fredkin Gate	3	3	18	7.824
Sayem Gate	4	4	18	7.824
Toffoli Gate	3	3	24	7.465
Peres Gate	3	3	24	7.824
TSG Gate	4	4	24	7.850

Fig. 2.4 Differential reversible logic gates

2.5 LITERATURE SURVEY – 5

TITLE: Design of Low Power Multiplier Using Reversible Logic Gate

AUTHORS: Mr. Ashish K. Thakre, Prof. Sujata S. Chiwande, Mr. Sumit D. Chafale

PUBLICATION: IEEE

YEAR OF PUBLICATION: 2014

2.5.1 OVERVIEW:

Multiplier is a basic arithmetic cell in computer arithmetic units. The energy consumption in computation turns out to be deeply linked to the reversibility of the computation. The focus of this paper is to reduce the number of garbage output. The reduction of garbage output reduces the power consumption. It is proved that the proposed multiplier (MFA) is better than the existing multiplier. In the proposed multiplier we synthesized a reversible multiplier circuit with the help of Peres gate. This circuit can be also helpful for high speed multiplier for dedicated hardware. The prospect for further research includes the reversible implementation of more complex arithmetic circuits such as function evaluation and multiplicative division circuits using this multiplier.

2.5.2 IMPLEMENTATION:

The efficiency of proposed reversible multiplier depends on the choice of reversible gate. The efficient parallel adders will significantly improve the multiplier efficiency. The partial products are generated in parallel using Peres gates and thereafter the addition is done of all product terms by using reversible parallel adder designed from TSG gates. After replacing the reversible parallel adder by modified reversible adder comparison of both the multiplier and terms like garbage outputs, power dissipation, number of gates required and number of constant inputs. Thus, this paper provides the comparison of two multipliers according to their garbage outputs, power dissipation, number of gates required and number of constant inputs. The proposed multiplier is efficient in terms of number of Garbage outputs. The proposed 4*4 bit reversible multiplier is designed with minimum of 19 garbage outputs while the existing multiplier has 32 garbage outputs. Because of this garbage outputs the proposed multiplier will lead to a great reduction in power consumption.

Reversible Multiplier	No. of Reversible Gate	No. of Constant Outputs	No. of Constant Inputs
Existing circuit with TSG	29	58	31
Existing circuit with MKG	28	56	29
Existing circuit with HNG	28	52	28
Our Proposed Design with MFA	28	36	20

Fig.2.5 Comparative Experimental Results of Different Reversible Multiplier Circuits

CONCLUSION:

Multiplier is a basic arithmetic cell in computer arithmetic units. The energy consumption in computation turns out to be deeply linked to the reversibility of the computation. The focus of this paper is to reduce the number of garbage output. The reduction of garbage output reduces the power consumption. It is proved that the proposed multiplier (MFA) is better than the existing multiplier. In the proposed multiplier we synthesized a reversible multiplier circuit with the help of Peres gate. This circuit can be also helpful for high speed multiplier for dedicated hardware. The prospect for further research includes the reversible implementation of more complex arithmetic circuits such as function evaluation and multiplicative division circuits using this multiplier.

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

3.1 OVERVIEW:

32-bit combined integer and Floating point multiplier is a common operation in advance Digital Signal Processing (DSP) application. This design proposes a combined integer and Floating point Multiplier using Reversible logic. The Peres reversible gate used, contributes to the advancement of quantum computing technology by providing a fundamental building block for reversible quantum circuits and enabling the exploration of quantum algorithms with potential applications in various fields. It preserves information, which is crucial in quantum computing where maintaining coherence and avoiding information loss. Full Adder is implemented priorly using Peres reversible gate. Once the reversible gates are designed, the overall architecture of the multiplier is established. This involves determining how the integer and floating-point components will be processed in parallel and how their results will be combined at the end. This architecture is designed such that it minimizes the gate count and circuit depth while ensuring that all operations are reversible. More over many parameters such as delay, power consumption are obtained and compared that of with circuitry implemented utilizing logic gates.

3.2 IMPLEMENTATION METHODOLOGY:

A 32-bit combined integer and Floating point multiplier is constructed using Reversible logic gate i.e; Peres gate and Feynman gate as well as higher-level components like 2-bit, 4-bit, 8-bit, and 16-bit multipliers which are constructed using Peres Gate. In the beginning a half adder was constructed utilizing Peres gates, followed by the design of a full adder and a 2x2 bit multiplier. Building upon this, a 4x4 multiplier was constructed using the 2x2 multiplier, further advancing to an 8x8 multiplier employing the 4x4 multiplier. This progression continued with the creation of a 16x16 multiplier utilizing the 8x8 multiplier, ultimately culminating in the development of a 32x32 bit multiplier using the 16x16 multiplier, with this a Integer Multiplier is implemented. This multiplier takes two 32-bit integer inputs and delivers a 64-bit product.

The next phase focused on the floating-point multiplier. Initially, a sign bit calculation was devised using Feynman gate, and an 8-bit ripple carry adder was constructed using full adders. Further components included a subtractor and a 24x24 bit multiplier for mantissa calculation. Ripple carry adders were employed for exponent calculation.. The system was designed to provide integer output when both input values are integers, floating-point output when both are floating-point numbers, and floating-point output when one input is an integer and the other is a floating-point number. Floating-point numbers typically consist of three components: a sign bit, a significand (also known as mantissa), and an exponent. The sign bit indicates whether the number is positive or negative, the significand represents the significant digits of the number, and the exponent determines the scale of the number. With this Floating point Multiplier is implemented. Then, leveraging this groundwork, a floating-point multiplier is developed. Floating-point numbers are a way of representing real numbers in computer systems, where the decimal point can "float" relative to the significant digits of the number. This allows a wide range of values to be represented with a fixed number of bits, suitable for both very large and very small numbers.Finally, the floating-point multiplier was assembled, integrating both the integer multiplier and the floating-point multiplier A floating-point multiplier takes two floating-point numbers as input and produces their product as output, while correctly handling the exponent and significand to maintain precision and range.

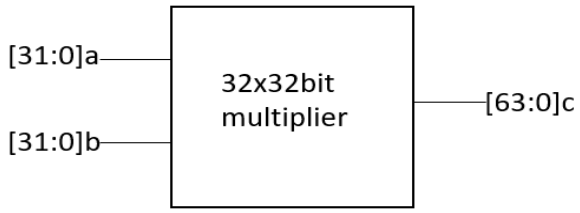


Fig.3.1 Block Diagram of Integer Multiplier

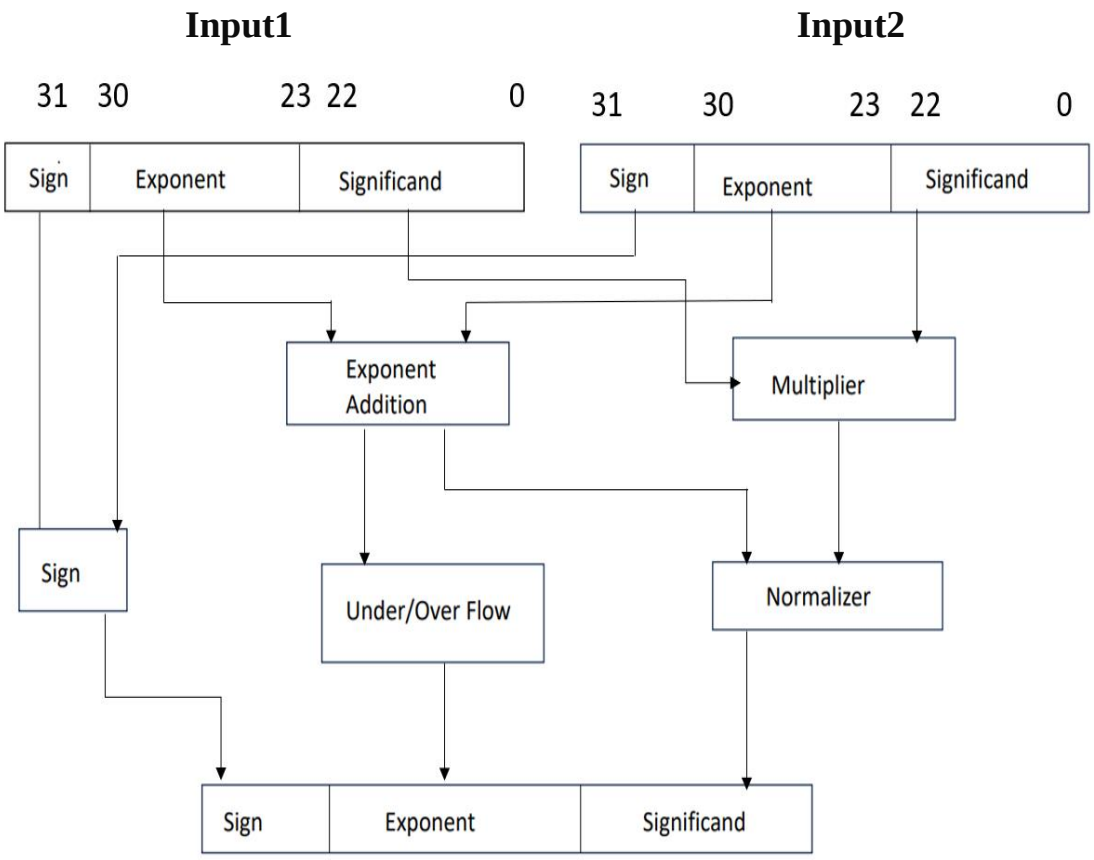


Fig.3.2 Block Diagram of floating Point Multiplier

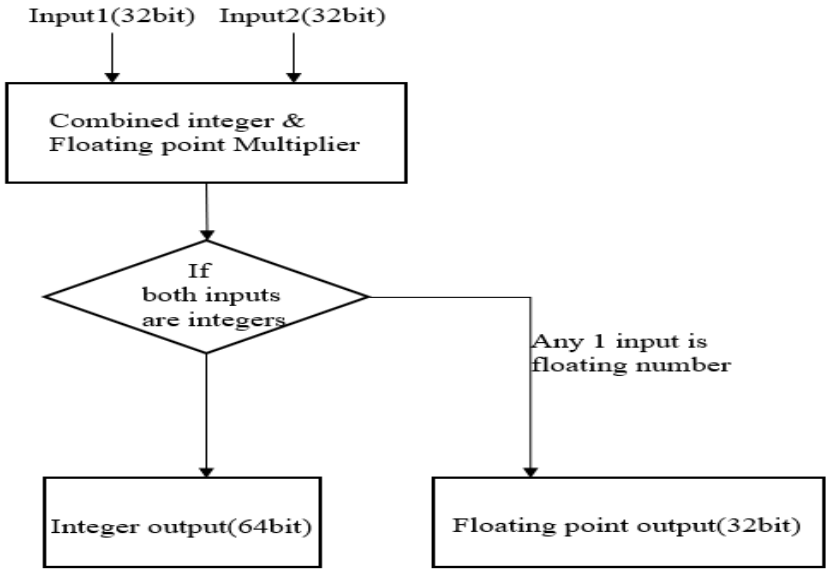


Fig.3.3 Flow Chart for Combined Integer and Floating Point Multiplier

IEEE 754 FORMAT:

The IEEE 754 standard defines formats for representing floating-point numbers, as well as rules for performing arithmetic operations on them. It specifies formats for single precision (32-bit) and double precision (64-bit) floating-point numbers, along with encoding schemes for special values such as positive and negative infinity, zero, and NaN (Not a Number).

Floating-point numbers represented according to IEEE 754 consist of three components: a sign bit, an exponent, and a significand (also known as a mantissa). The sign bit determines whether the number is positive or negative, the exponent specifies the magnitude of the number, and the significand holds the precision or accuracy of the number.

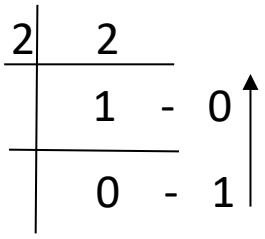
The IEEE 754 standard has become ubiquitous in modern computing, as it provides a consistent and reliable way to perform floating-point arithmetic across different platforms. However, it's important to be aware of its limitations, such as the potential for rounding errors and the finite precision of floating-point numbers, which can sometimes lead to unexpected behavior in numerical computations.

Building upon this, a 4x4 multiplier was constructed using the 2x2 multiplier. This progression continued with the creation of a 16x16 multiplier utilizing the 8x8x multiplier, ultimately culminating in the development of a 32x32 bit multiplier using the 16x16 multiplier.

Conversion of Decimal to IEEE 754 32-bit floating point notation :

Example: 2.5

- Convert 2 into binary form
-



2=10

- convert 0.5 into decimal form

0.5x2 | 1.0 | = 0.5 =1

- Step-1: Representation of 2.5 in binary form

  10.1

- Step-2: Need to shift the decimal point to the left till it reaches the 2^{nd} bit.

  $1.01x2^1$ (scientific notation)

- Step-3: For positive number representation of sign bit is 0 and for negative number sign bit is 1.

  Taken number is a positive number ,so sign bit is 0.

In the IEEE 754 notation Exponent has to cover both positive and negative numbers so biasadded(127).

For taken number bias is 127+1=128 so, binary representation of 128 is the exponent of this number.

1 sign bit	**8 exponent bits**	**23 fraction bits**
0	10000000	01000000000000000000000000000000

32 bit Floating point number representation of 2.5 is
01000000001000000000000000000000000000

Calculation of Floating point Multiplication:

Decimal IEEE format:

Input1=+2.5 → $+1.25x2^1$

Input2=+2.5 →$+1.25x2^1$

Sign = (+) + (+) =+

Exponent=$2^1 +2^1 =2^2$

Mantissa=1.25x1.25=1.5625

Output=$+1.5625x2^2$ =+6.25

Binary Representation of output:01000000110010000000000000000000

Calculation:

Decimal	IEEE format:

Input1=+110.692764 → $+1.7295744 \times 2^6$

Input2=+166.45981 → $+1.3004673 \times 2^7$

Sign = (+) + (+) =+

Exponent=$2^6 + 2^7 = 2^{13}$

Mantissa=1.7295744x1.3004673 =2.24925495

For 32 bit floating point multiplication exponent takes 8 bits if exponent exceeds 2^8 ,Normalization is required,it can be done by 2.24925495 divided and multiplied by 2 equals to 1.1246275x2

Output=$+1.1246275 \times 2^{14}$ =+18425.896

Binary Representation of output:01000110100011111111001111001011

Conversion of binary to decimal:

Output:

0	10000001	10010000000000000000
Sign	exponent	fraction

According to sign bit it is a positive number: +1

Exponent: 10000001 $=(129)_{10}$

Remove bias from the exponent (127) = 129-127=2

Convert fraction bits back to base 10

10010000000000000000= $1x2^{-1}+0x2^{-2}+0x2^{-3}+1x2^{-4}$

=1/2 +1/16

=0.5625

For the single precision floating number add 1 to fraction

1+0.5625=1.5625

Combine all parts:$1x(1.5625)x2^{129-127}$ =+6.25

Integer multiplication: (2x2 multiplication)

B1 **B0** (Multiplicand)

A1 **A0** (Multiplier)

A0B1 **A0B0** ⎫ Partial Products

A1B1 **A1B0** ⎭

P3 **P2** **P1** **P0** (Product)

Example:

Input1=01

Input2=10

```
      0  1
      1  0
   ---------
      0  0
   0  1
   ---------
   0  0  1  0
```

(4x4 Multiplication)

B3 **B2** **B1** **B0**(Multiplicand)

x **A3** **A2** **A1** **A0** (Multiplier)

A0B3 **A0B2** **A0B1** **A0B0** → **2x2 Multiplier**

+ A1B3 **A1B2** **A1B1** **A1B0** Partial Products

+ A2B3 **A2B2** **A2B1** **A2B0**

+ A3B3 **A3B2** **A3B1** **A3B0**

P7 **P6** **P5** **P4** **P3** **P2** **P1** **P0** (Product)

Example:

Input1=1010 , Input2=0101

				1	0	1	0	
			x	0	1	0	1	
				1	0	1	0	
			0	0	0	0		
		1	0	1	0			
	0	0	0	0				
0	0	1	1	0	0	1	0	

Example for 32x32 Multiplication:

Input 1: 11010101010110010110010011001010

Input 2: 01010011011001000111010100110011

Output: 01000101011111111001100000011000001111001101000001100110001111 10

CHAPTER 4

SIMULATION AND RESULTS

4.1 DESIGN OF PERES GATE:

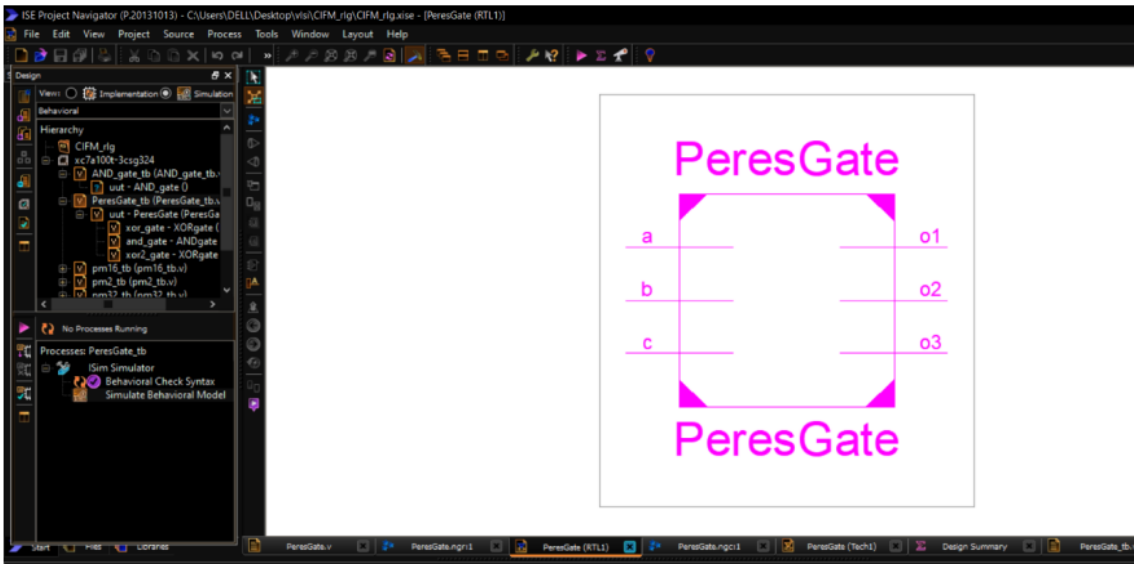


Fig.4.1 Design of Peres Gate

4.1.1 RTL SCHEMATIC OF PERES GATE

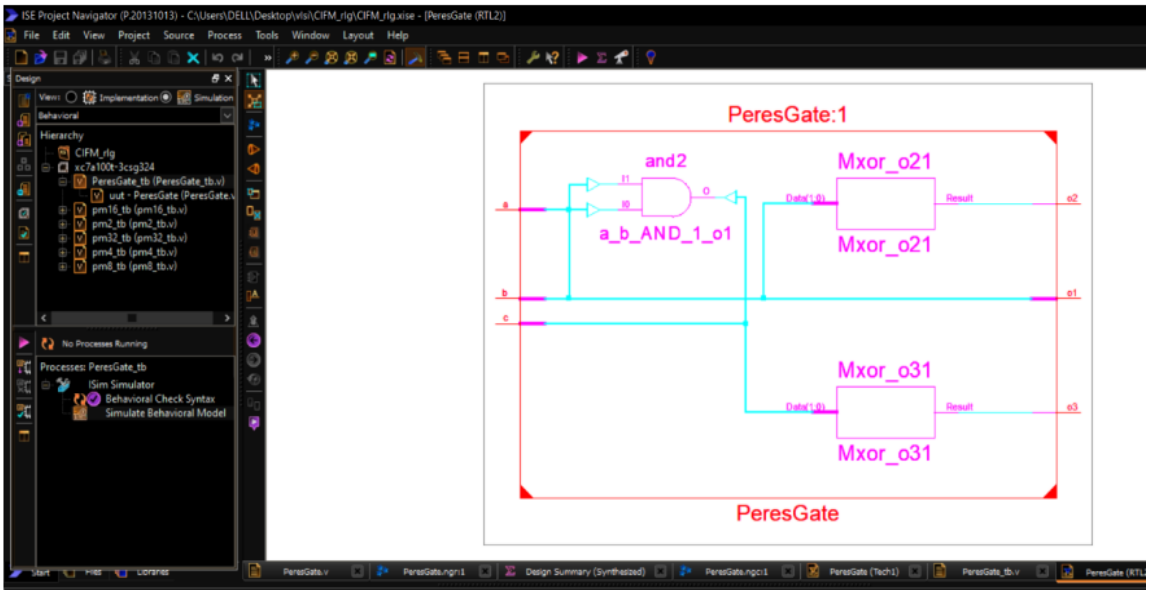


Fig.4.2 RTL Schematic of Peres Gate

4.1.2 TECHNOLOGY SCHEMATIC OF PERES GATE

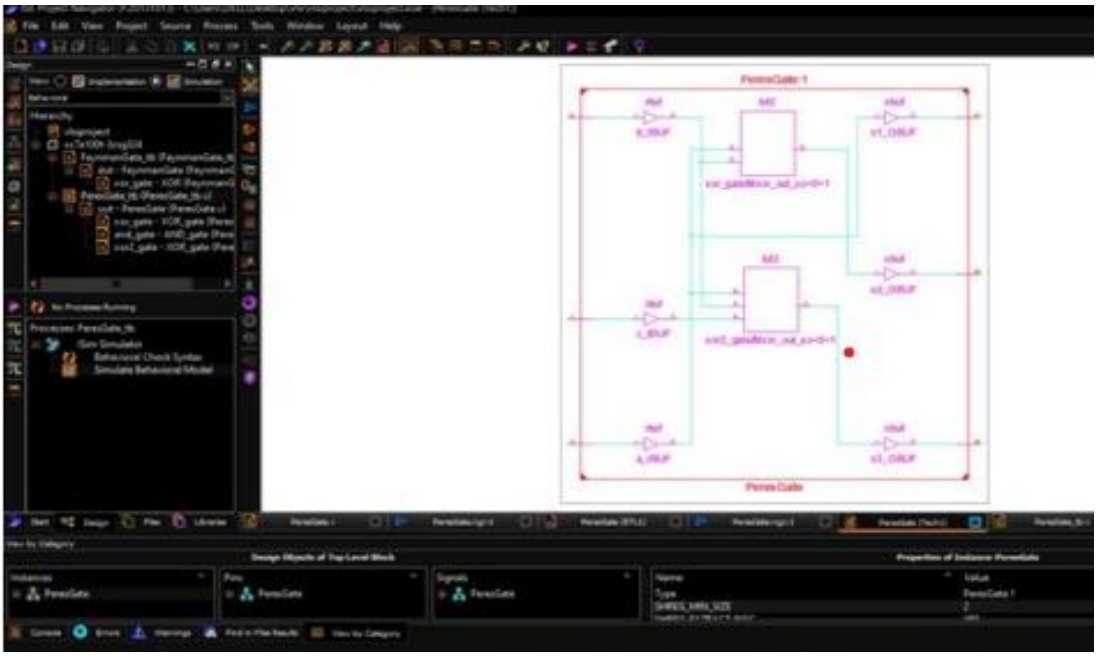


Fig.4.3 Technology Schematic of Peres Gate

4.1.3 OUTPUT WAVEFORM

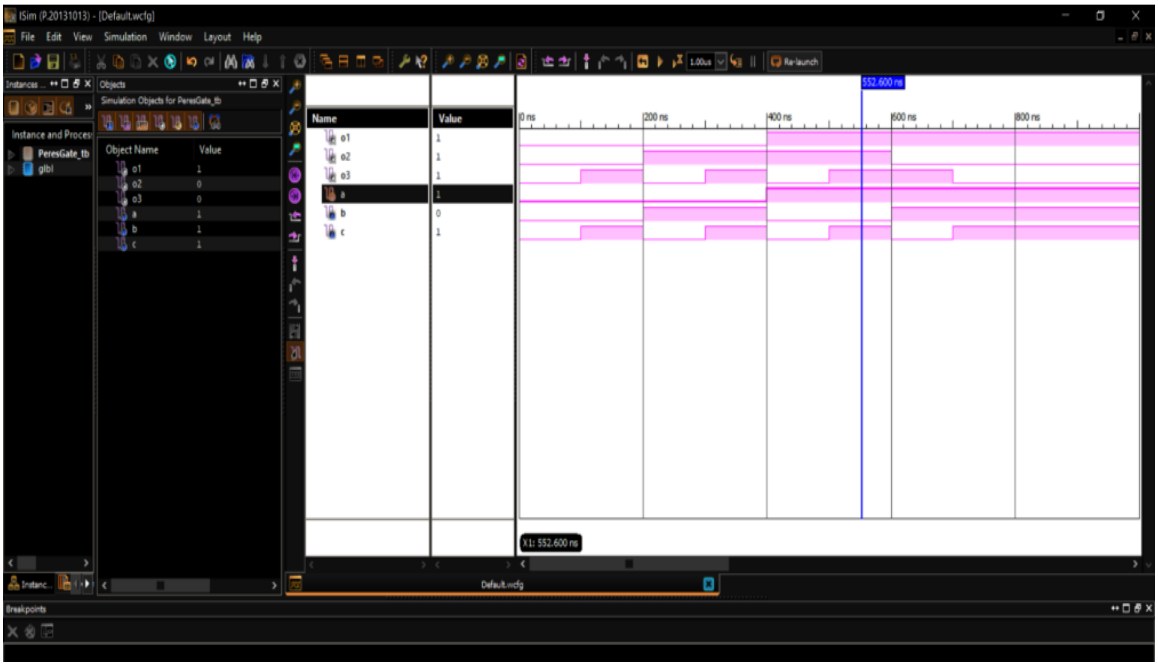


Fig.4.4 Output Waveform of Peres Gate

4.1.4 Synthesis Report of Peres Gate:

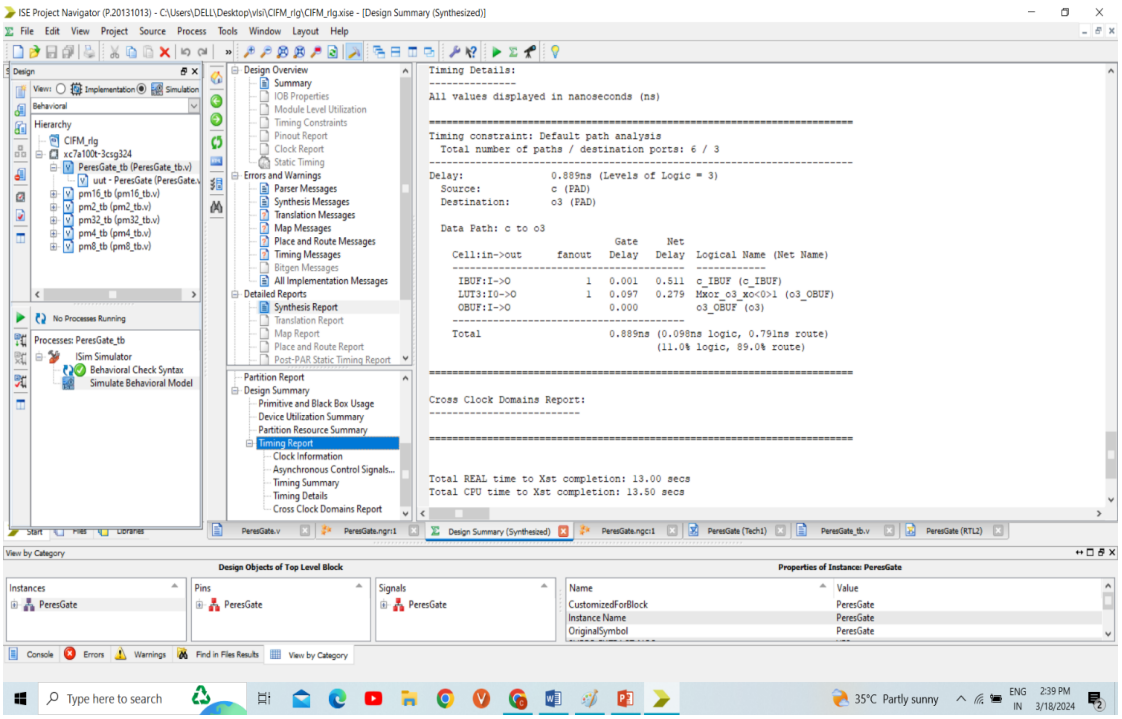


Fig.4.5 Synthesis Report of Peres Gate

4.2 DESIGN OF FEYNMAN GATE:

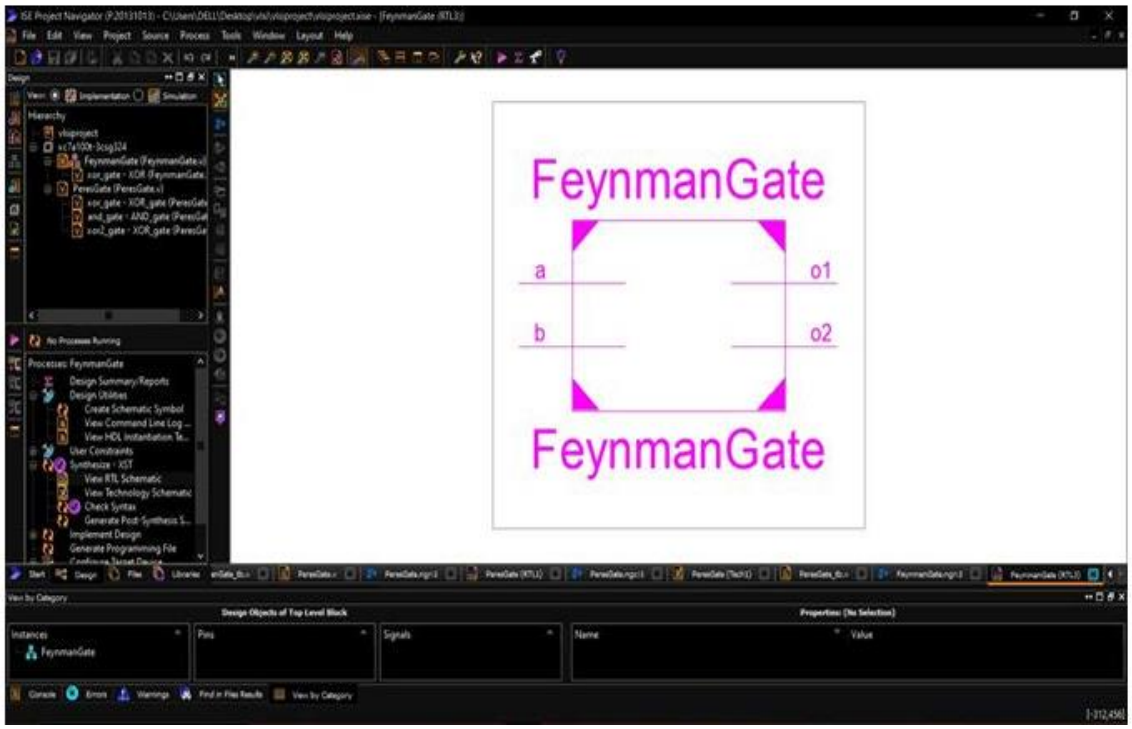


Fig.4.6 Design of Feynman Gate

4.2.1 RTL SCHEMATIC OF FEYNMAN GATE

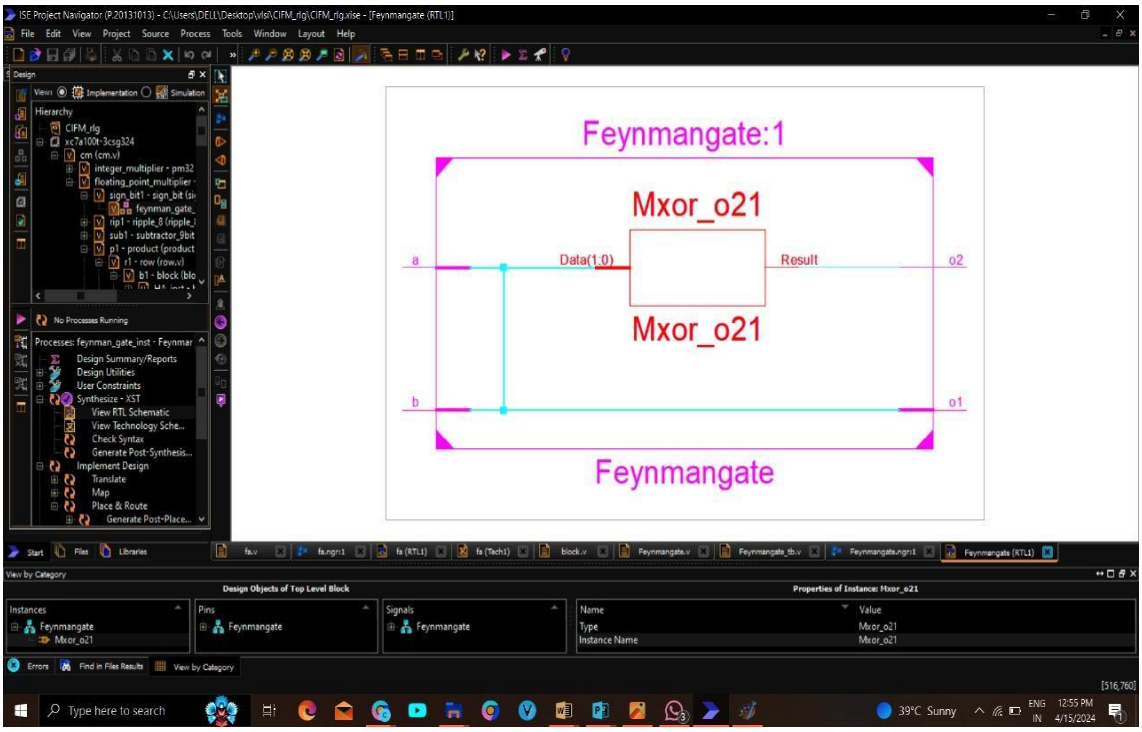


Fig.4.7 RTL Schematic of Feynman Gate

4.2.2 TECHNOLOGY SCHEMATIC OF FEYNMAN GATE

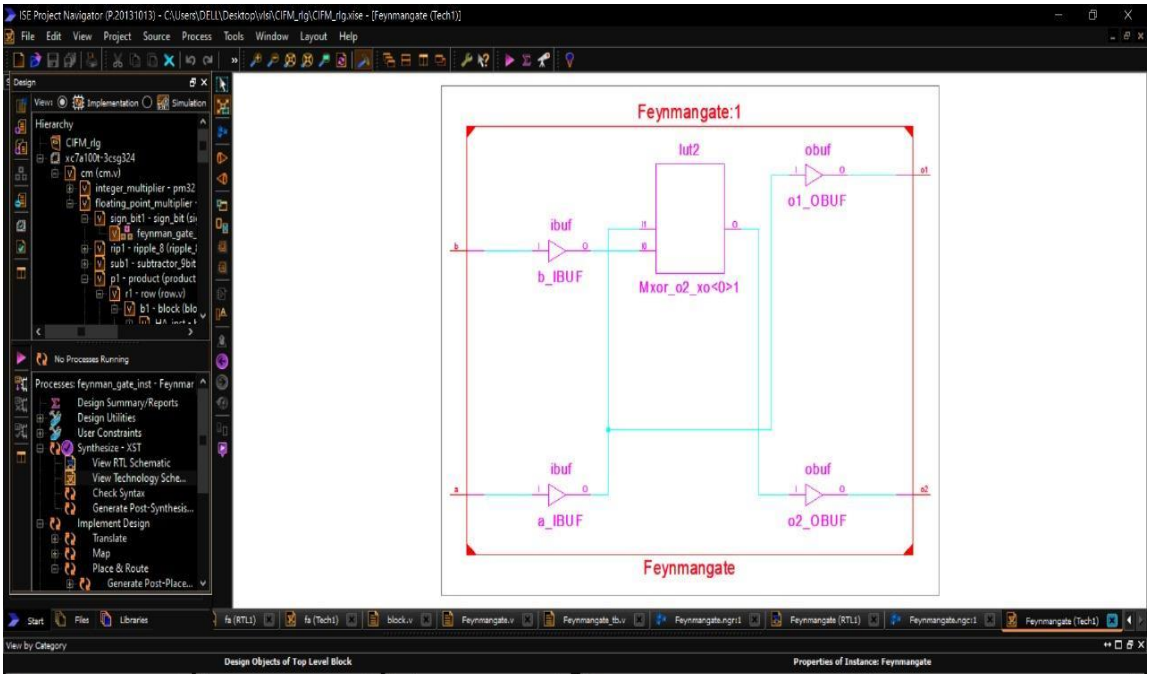


Fig.4.8 Technology Schematic of Feynman Gate

4.2.3 OUTPUT WAVEFORM OF FEYNMAN GATE

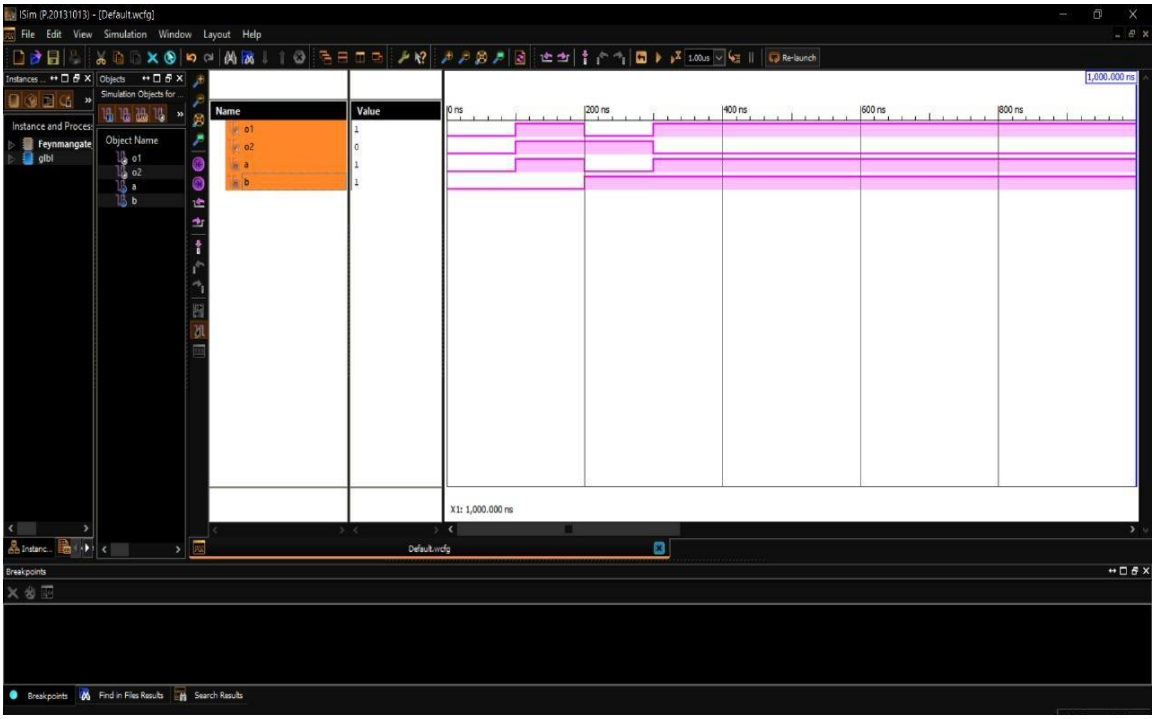


Fig.4.9 Output Waveform of Feynman Gate

4.2.4 SYNTHESIS REPORT OF FEYNMAN GATE

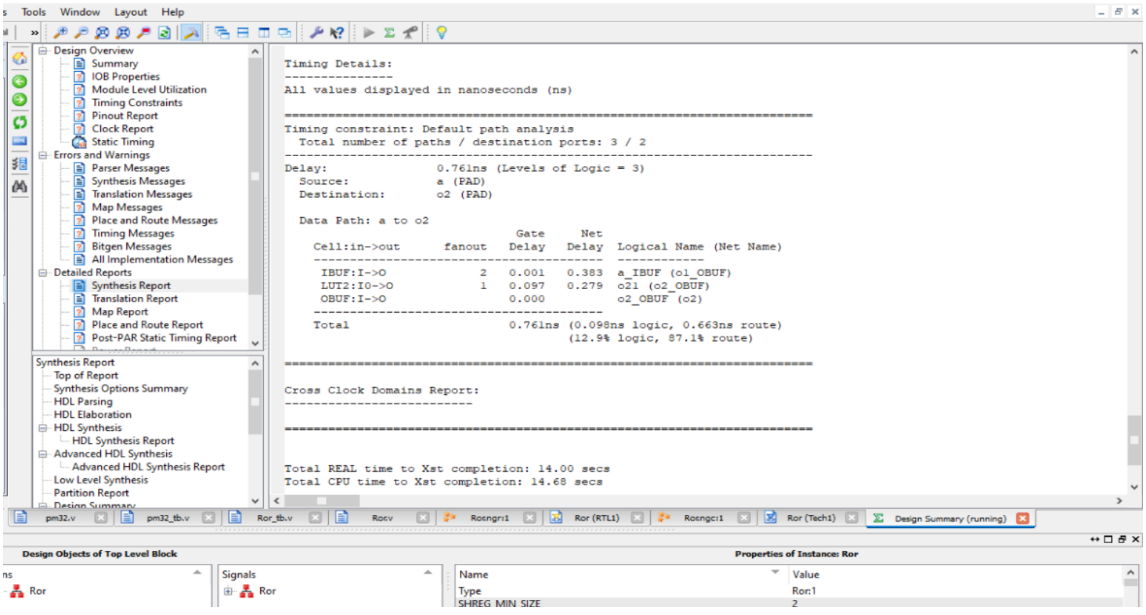


Fig.5 Synthesis Report of Feynman Gate

4.3 DESIGN OF HALF ADDER USING PERES GATE

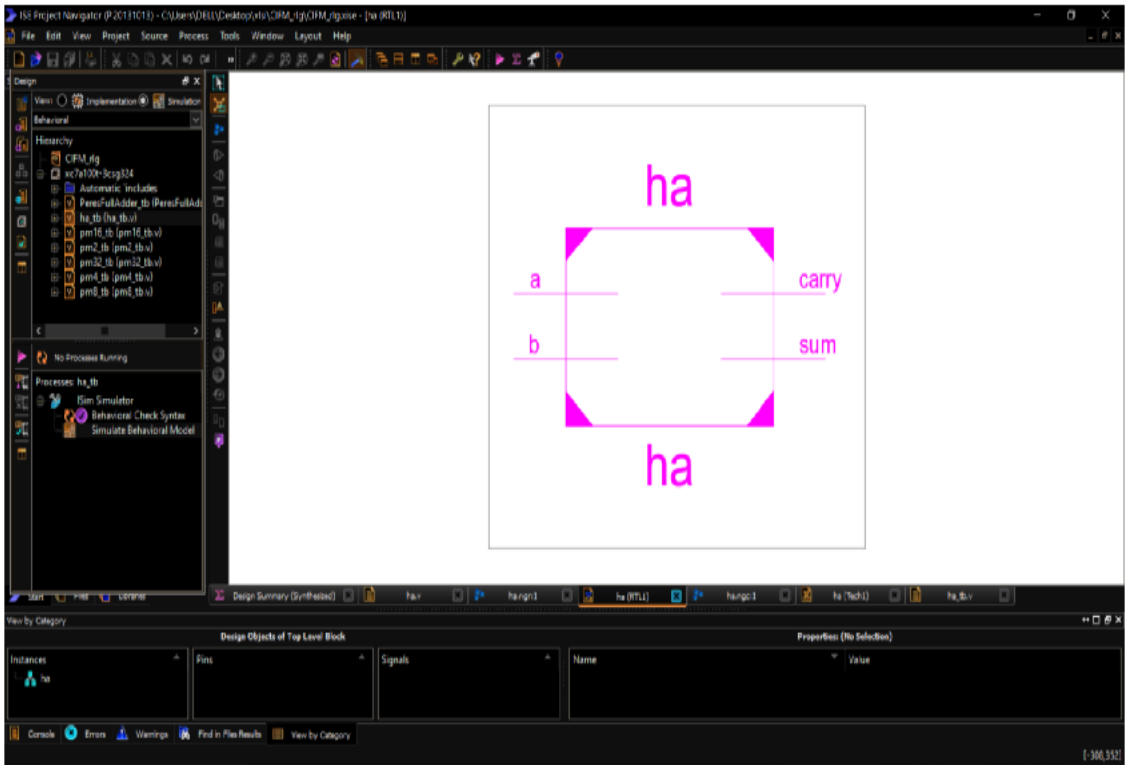


Fig.5.1 Design of half adder using Peres Gate

4.3.1 RTL SCHEMATIC OF HALF ADDER

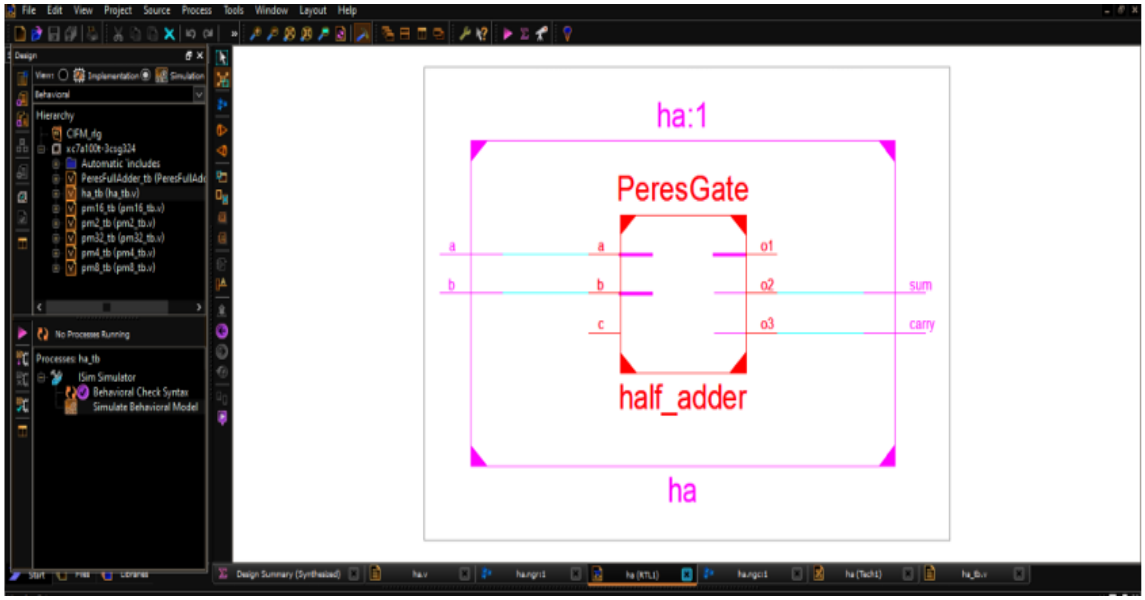


Fig.5.2 RTL schematic of half adder using Peres Gate

4.3.2 TECHNOLOGY SCHEMATIC OF HALF ADDER

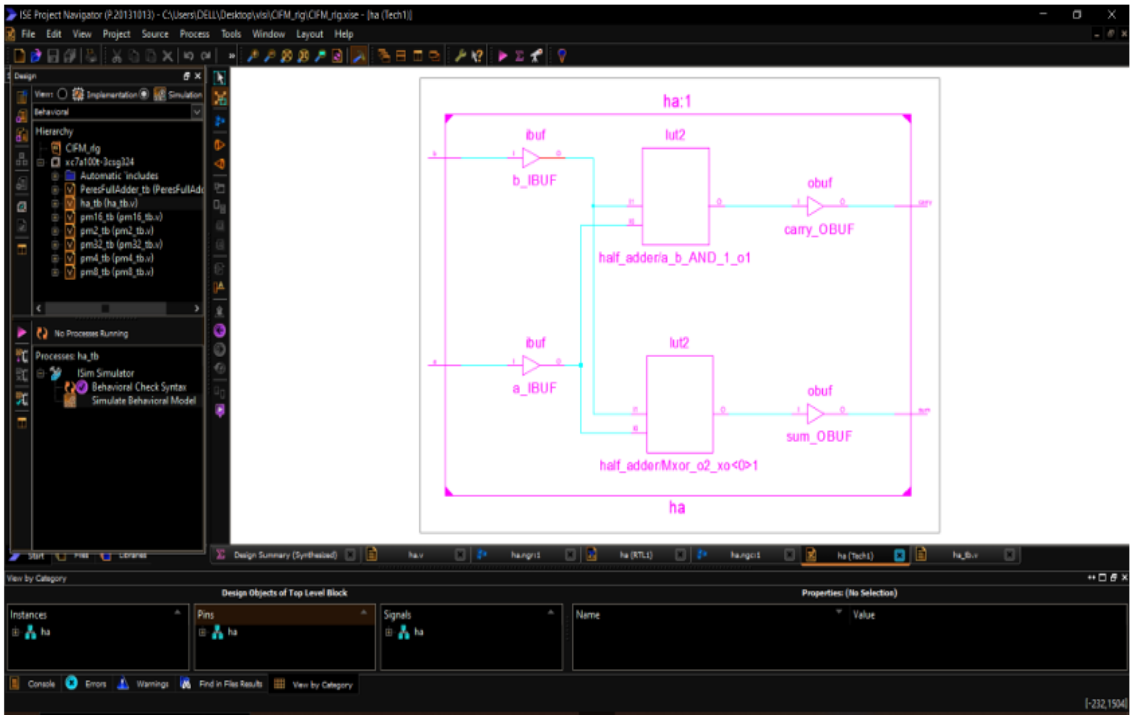


Fig.5.3 Technology schematic of half adder

4.3.3 OUTPUT WAVEFORM OF HALF ADDER

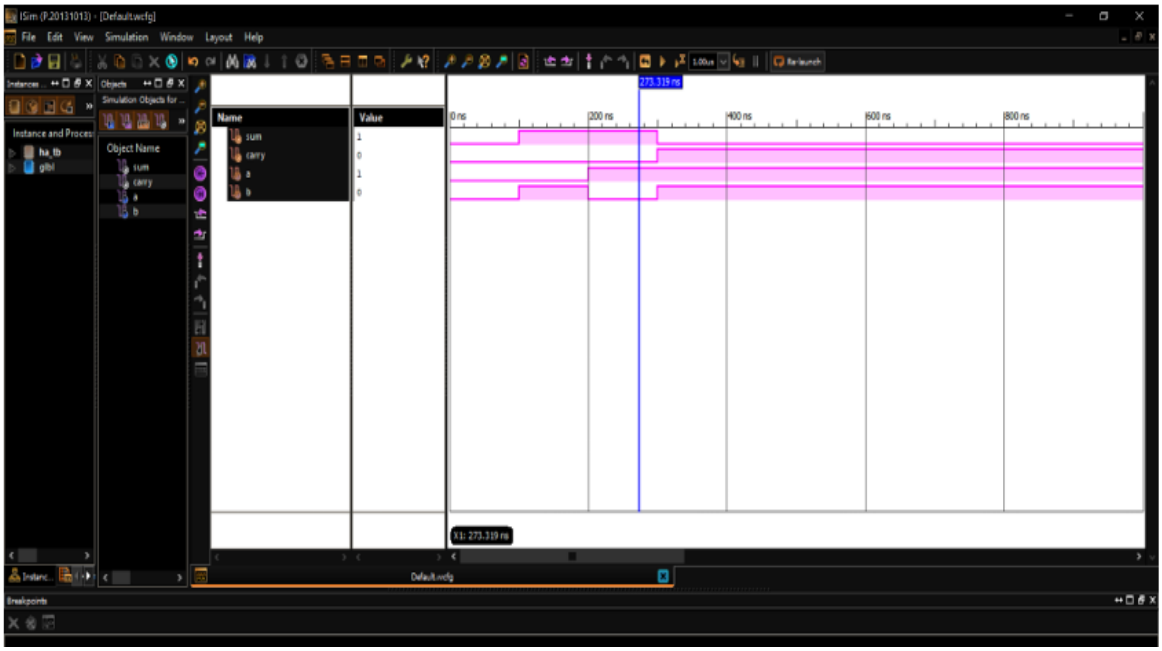


Fig.5.4 Output waveform of half adder

4.3.4 SYNTHESIS REPORT OF HALF ADDER

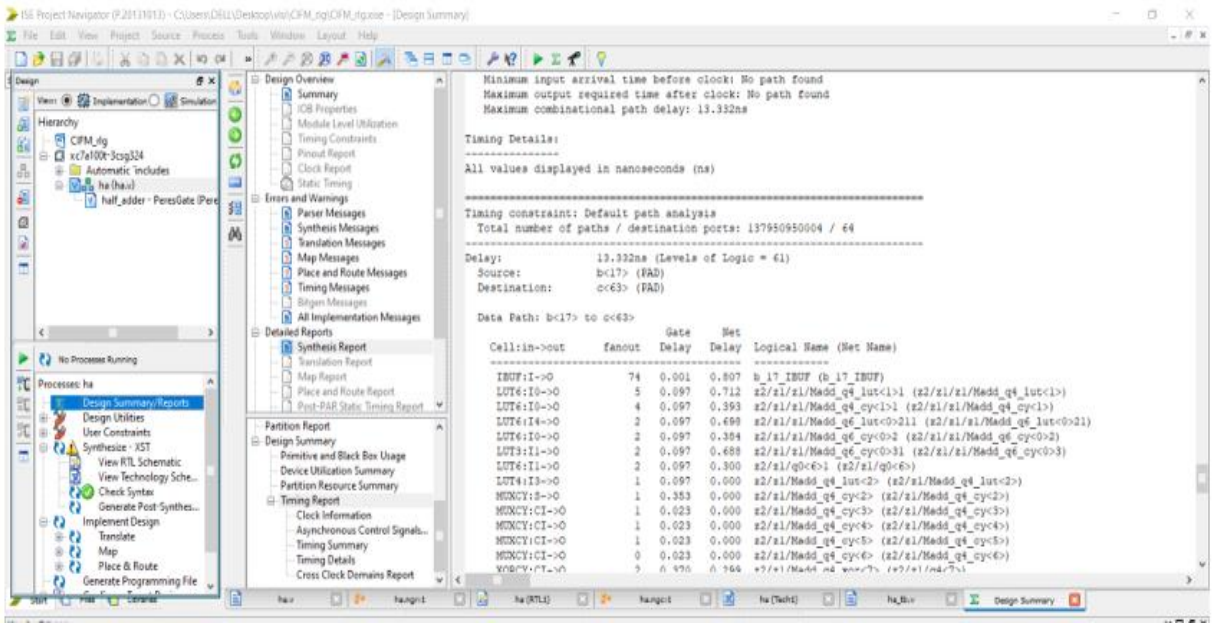


Fig.5.5 synthesis report of half adder

4.4 DESIGN OF 2BIT MULTIPLIER USING PERES GATE

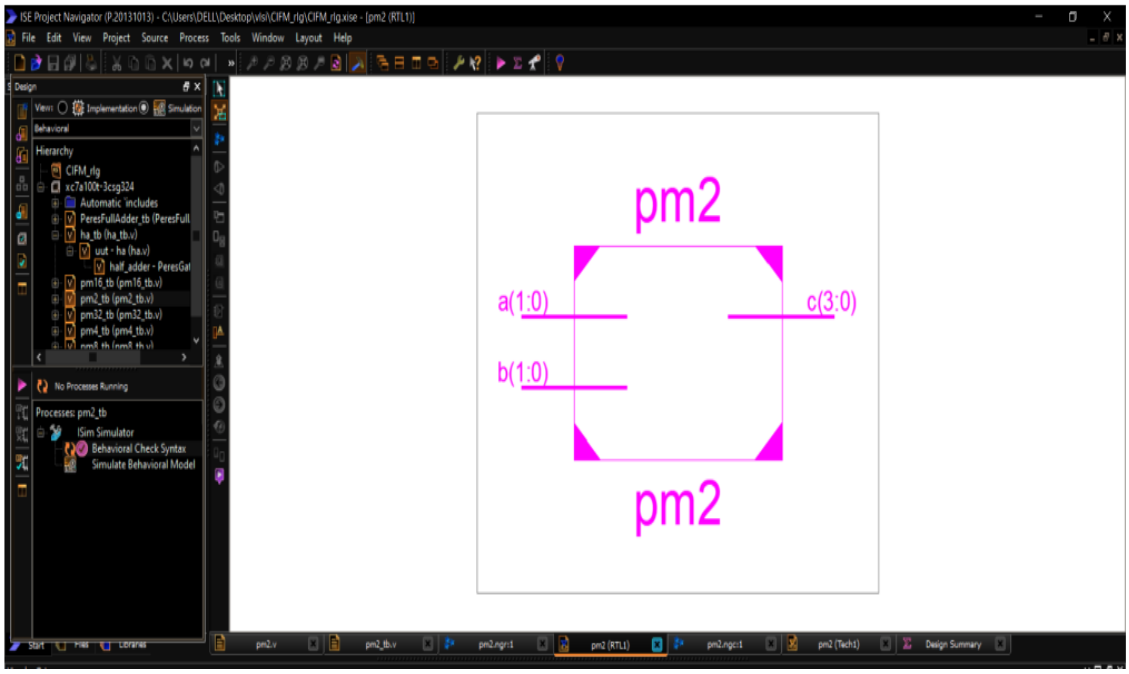


Fig.5.6 Design of 2bit Multiplier using Peres Gate

4.4.1 RTL SCHEMATIC OF 2BIT MULTIPLIER

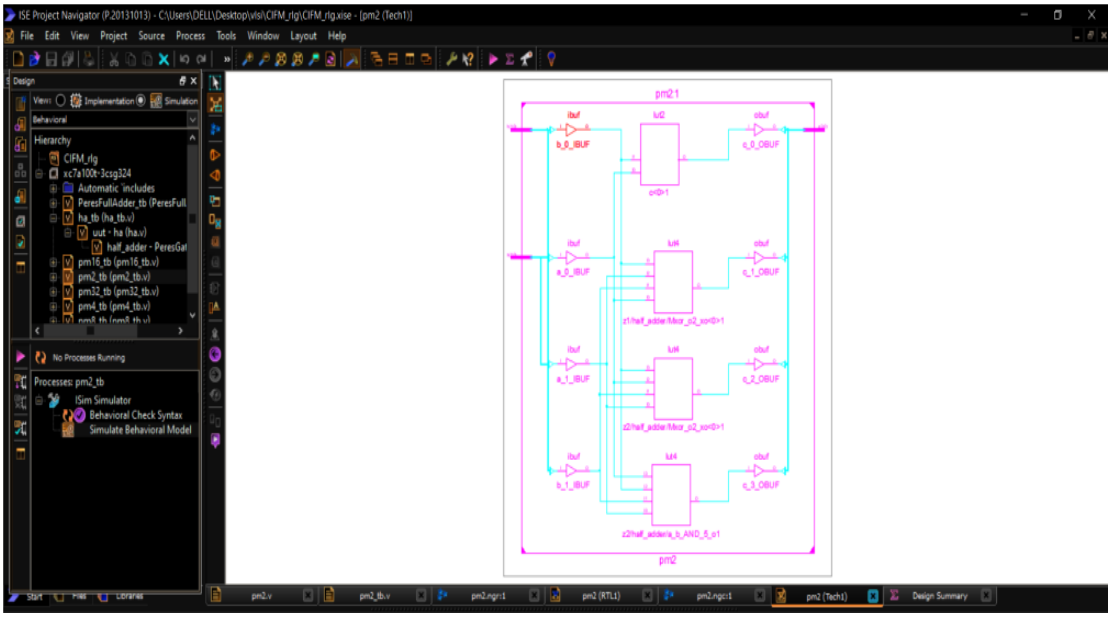


Fig.5.7 RTL Schematic of 2bit Multiplier using Peres Gate

4.4.2 OUTPUT WAVEFORM OF 2BIT MULTIPLIER

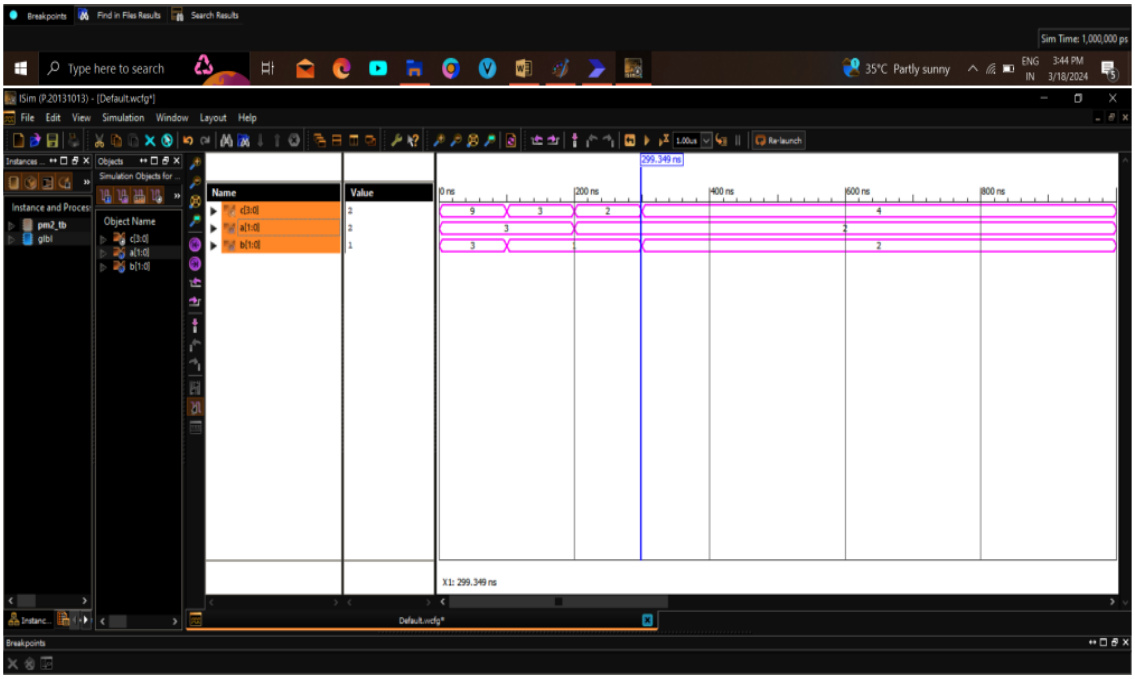


Fig.5.8 Output Waveform of 2bit Multiplier using Peres Gate

4.4.3 SYNTHESIS REPORT OF 2BIT MULTIPLIER

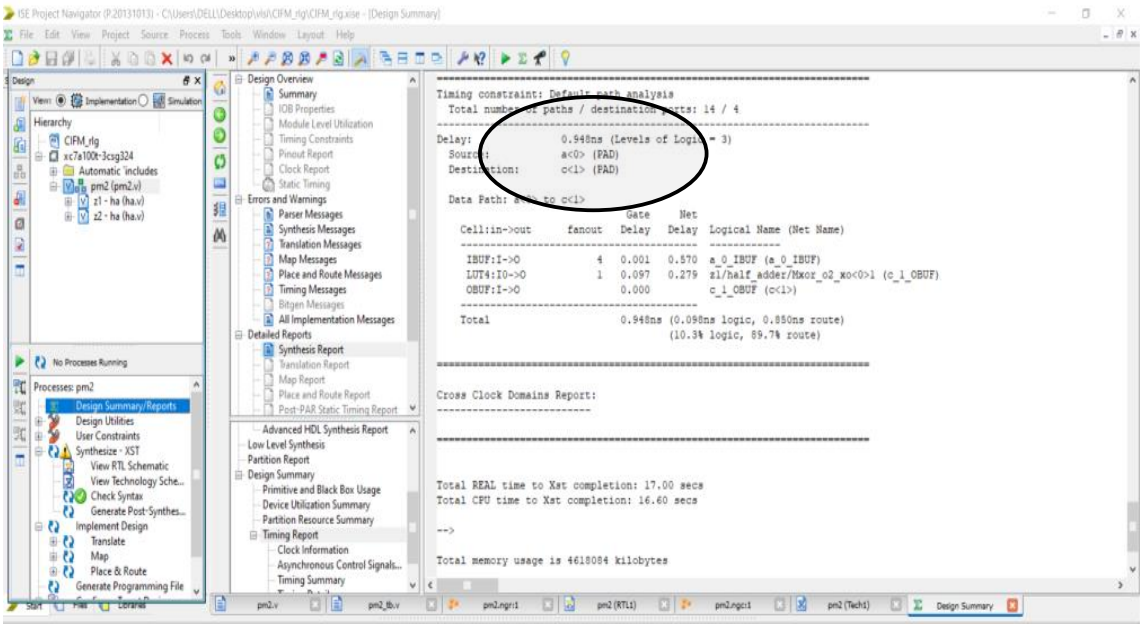


Fig.5.9 Synthesis Report of 2bit Multiplier using Peres Gate

4.5 Design of 4bit Multiplier Using Peres Gate :

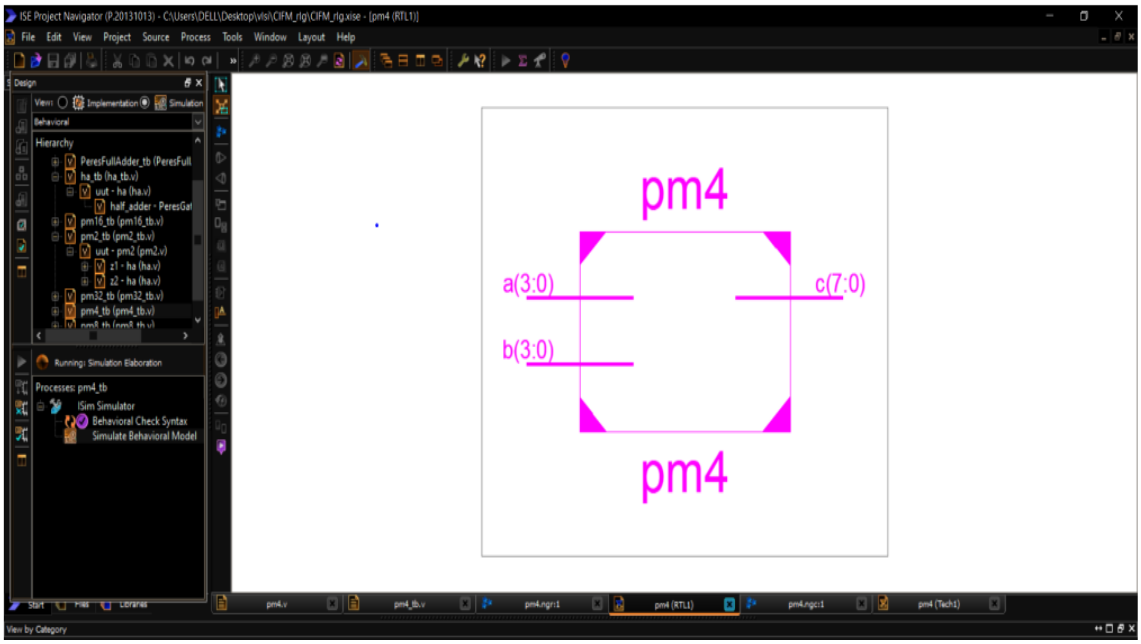


Fig.6 Design of 4bit Multiplier using Peres Gate

4.5.1 RTL SCHEMATIC OF 4BIT MULTIPLIER USING PERES GATE

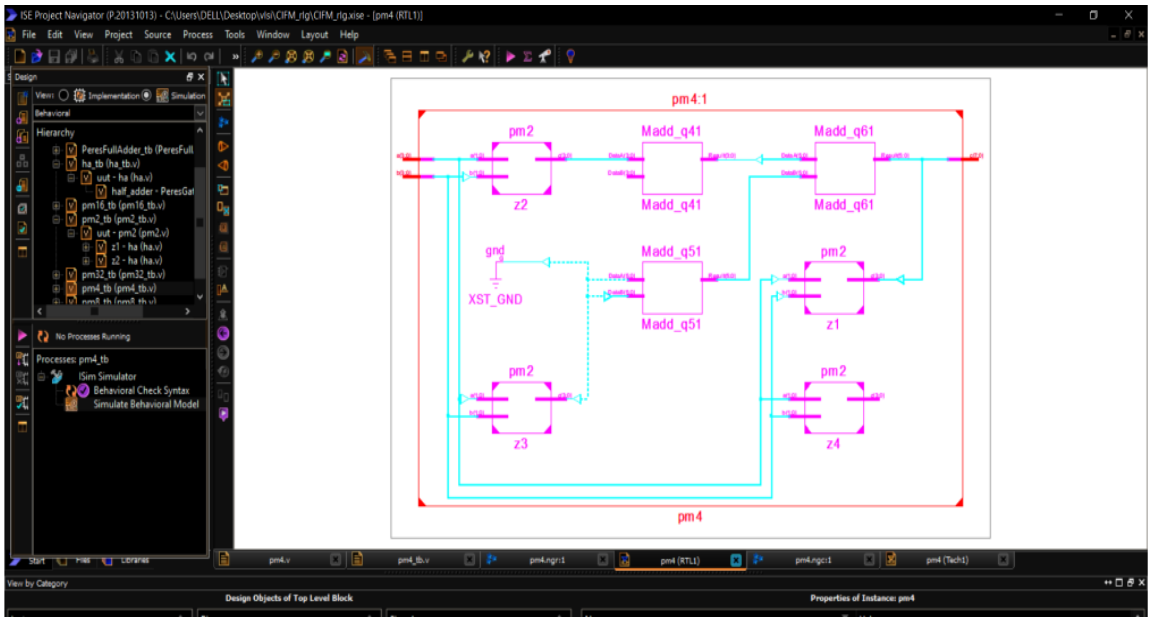


Fig.6.1 RTL Schematic of 4bit Multiplier using Peres Gate

4.5.2 OUTPUT WAVEFORM OF 4BIT MULTIPLIER

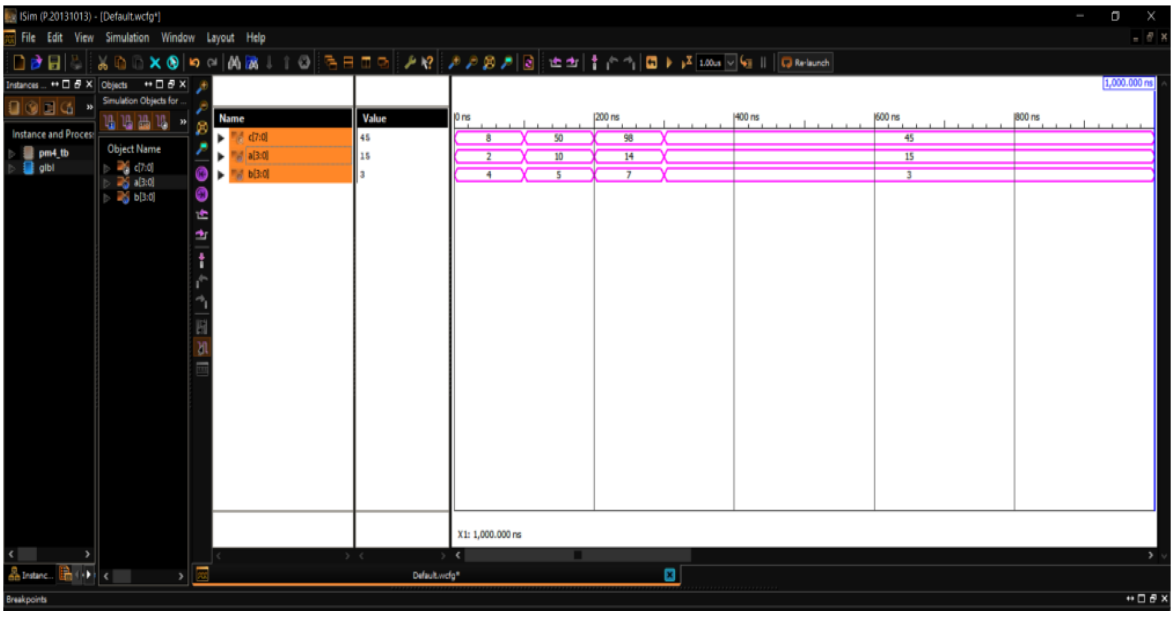


Fig.6.2 Output Waveform of 4bit Multiplier using Peres Gate

4.5.3 SYSNTHESIS REPORT OF 4BIT MULTIPLIER

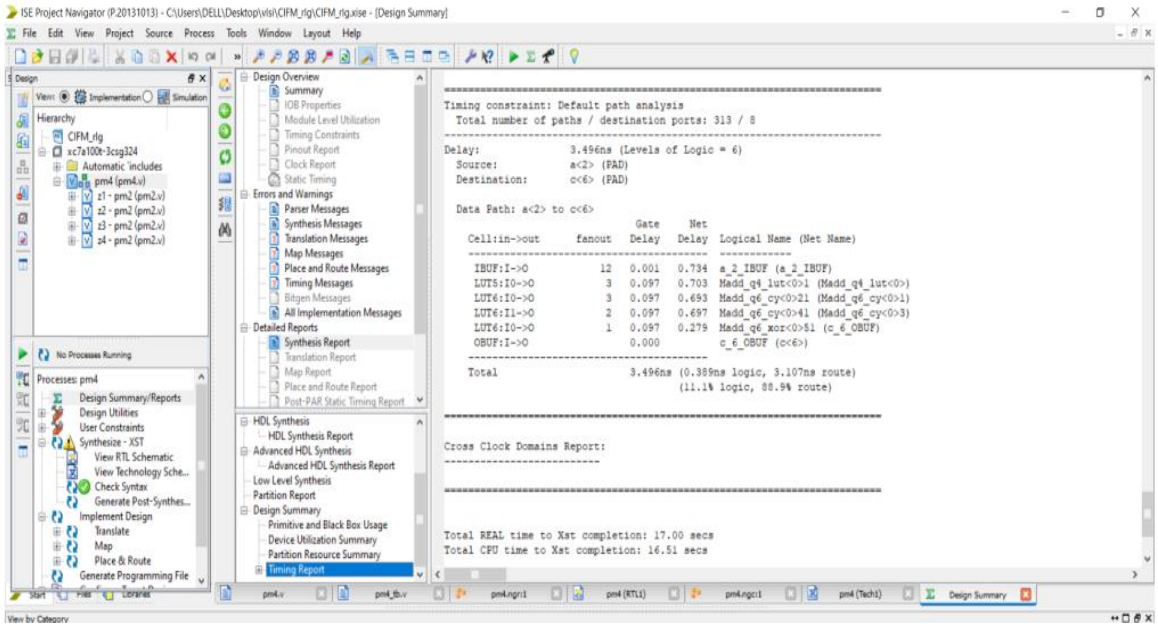


Fig.6.3 Synthesis Report of 4bit Multiplier using Peres Gate

4.6 Design of 8bit Multiplier using Peres Gate:

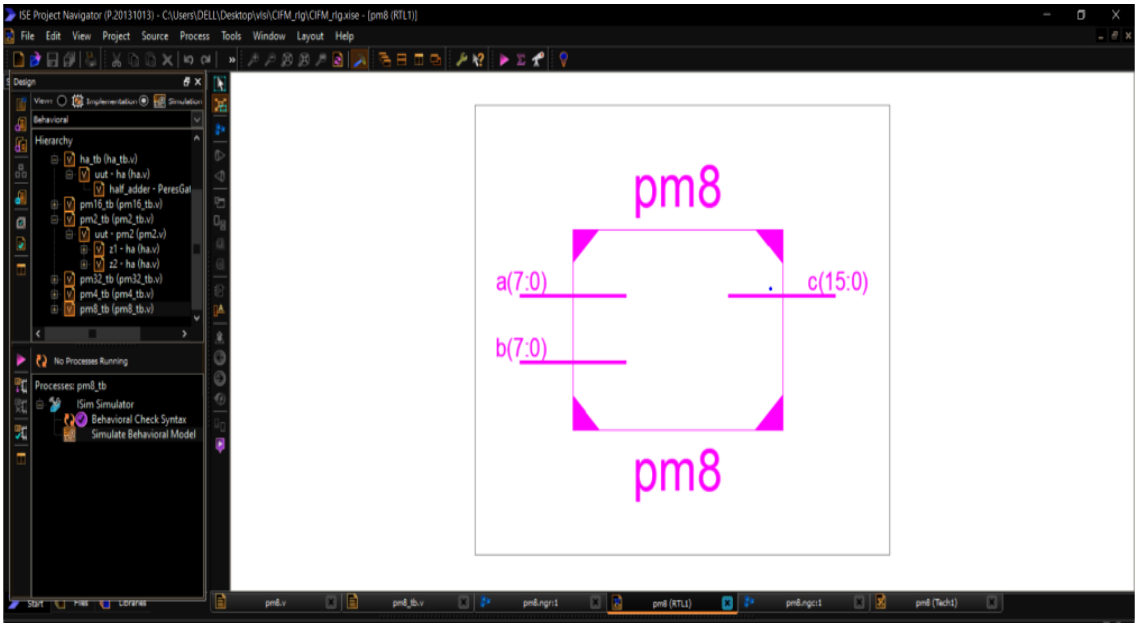


Fig.6.4 Design of 8bit Multiplier using Peres Gate

4.6.1 RTL SCHEMATIC OF 8BIT MULTIPLIER

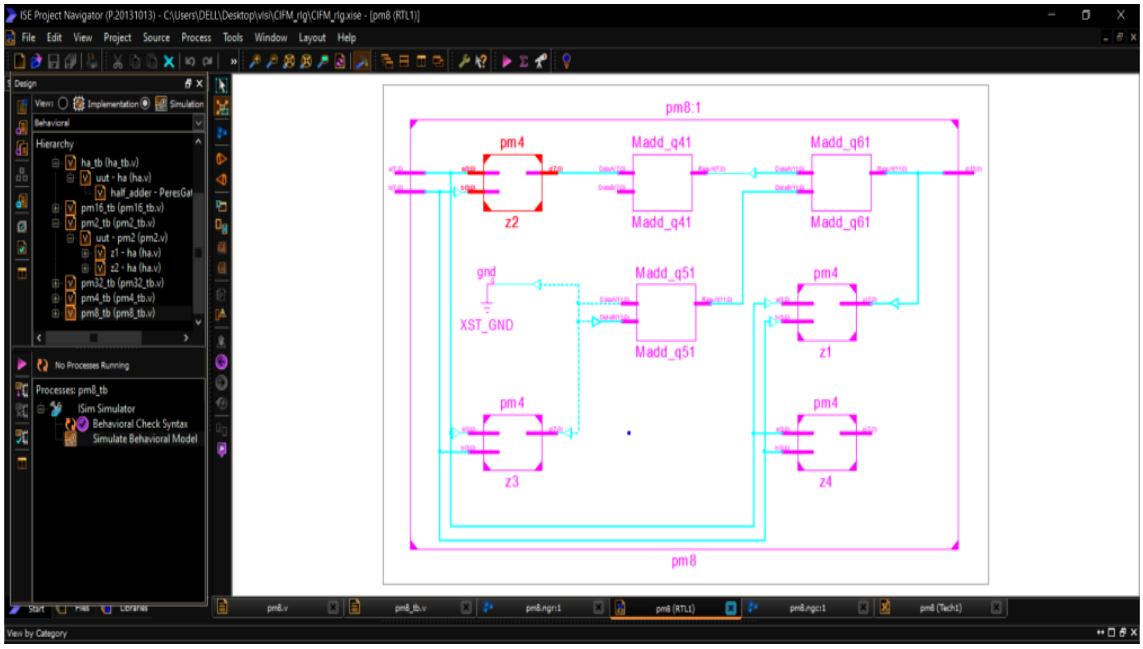


Fig.6.5 RTL Schematic of 8bit Multiplier using Peres Gate

4.6.2 OUTPUT WAVEFORM 8-BIT MULTIPLIER

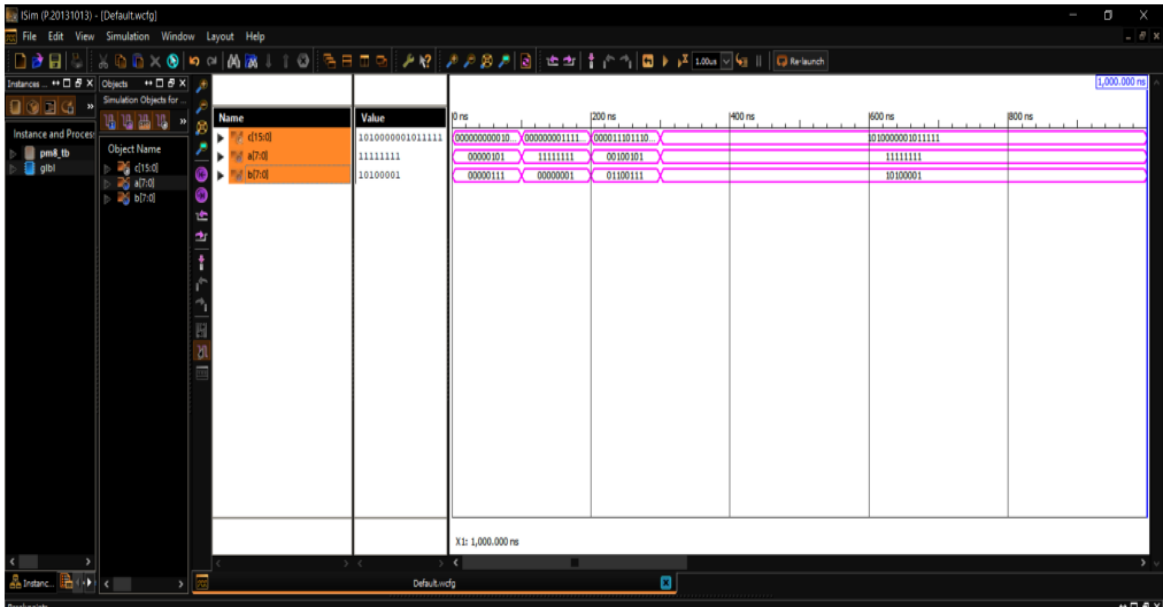


Fig.6.6 Output Waveform of 8bit Multiplier using Peres Gate

4.6.3 SYNTHESIS REPORT OF 8BIT MULTIPLIER

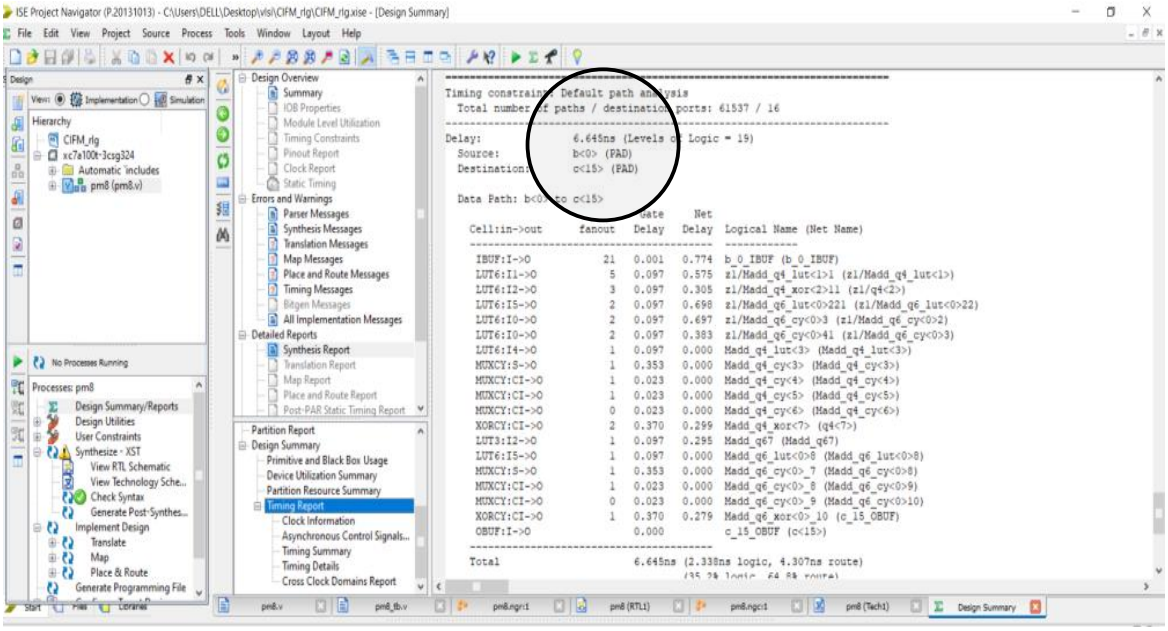


Fig.6.7 Synthesis Report of 8bit Multiplier using Peres Gate

4.7 DESIGN OF 16BIT MULTIPLIER USING PERES GATE:

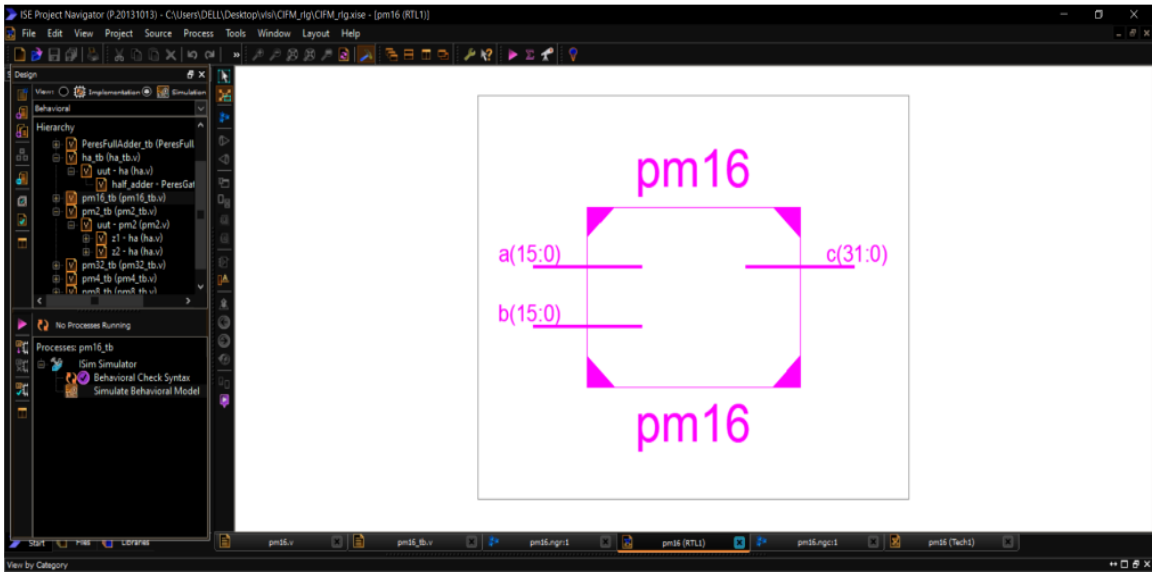


Fig.6.8 Design of 16bit Multiplier using Peres Gate

4.7.1 RTL SCHEMATIC OF 16BIT MULTIPLIER

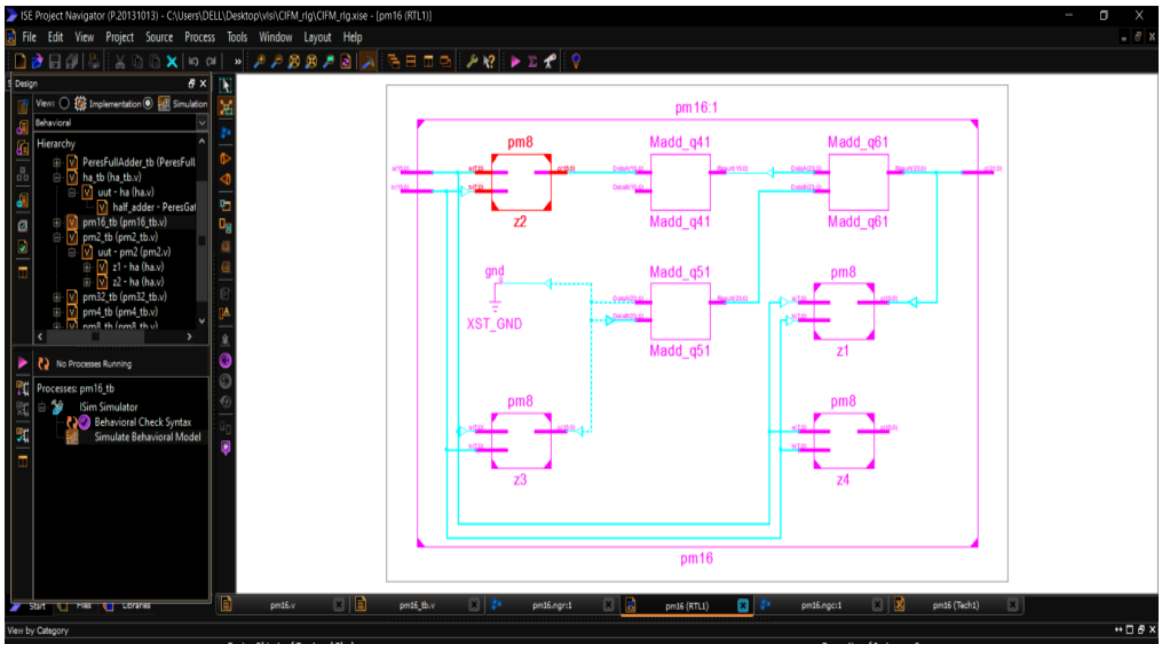


Fig.6.9 RTL Schematic of16 bit Multiplier using Peres Gate

4.7.2 OUTPUT WAVEFORM OF 16BIT MULTIPLIER

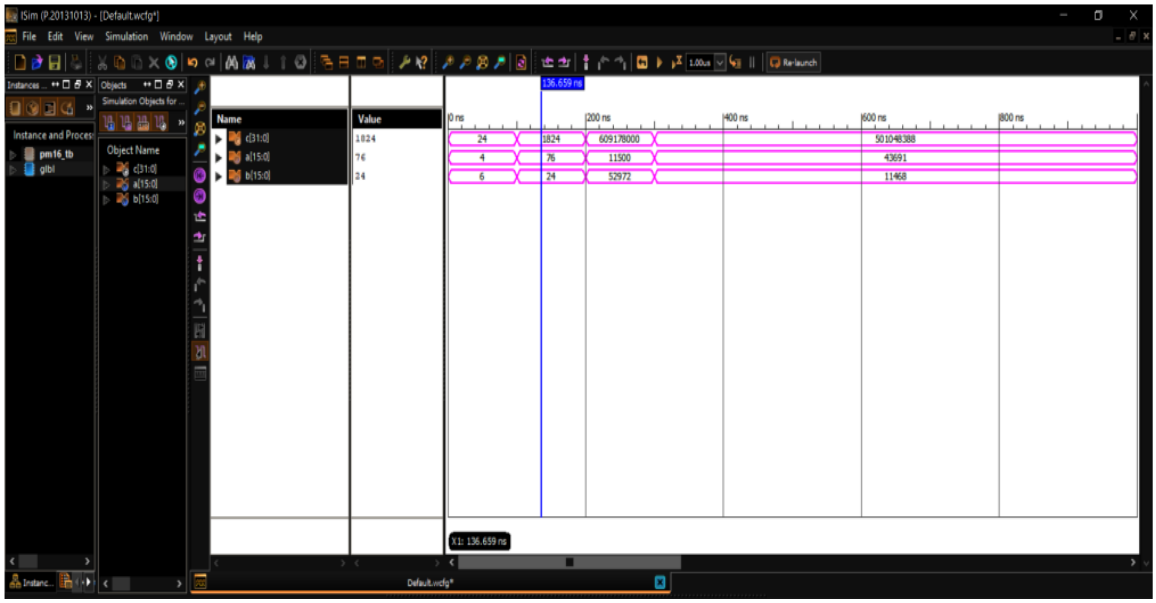


Fig.7 Output Waveform of 16bit Multiplier using Peres Gate

4.7.3 SYNTHESIS REPORT OF 16BIT MULTIPLIER

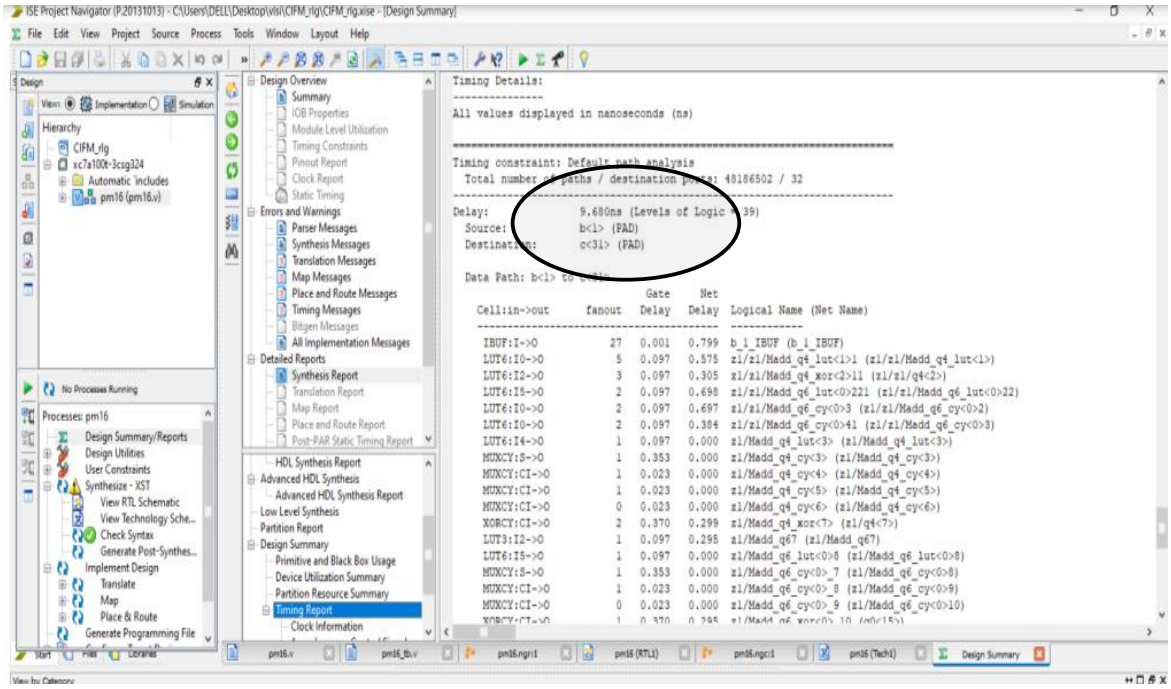


Fig.7.1 Synthesis Report of 16bit Multiplier using Peres Gate

4.8 DESIGN OF 32BIT MULTIPLIER USING PERES GATE:

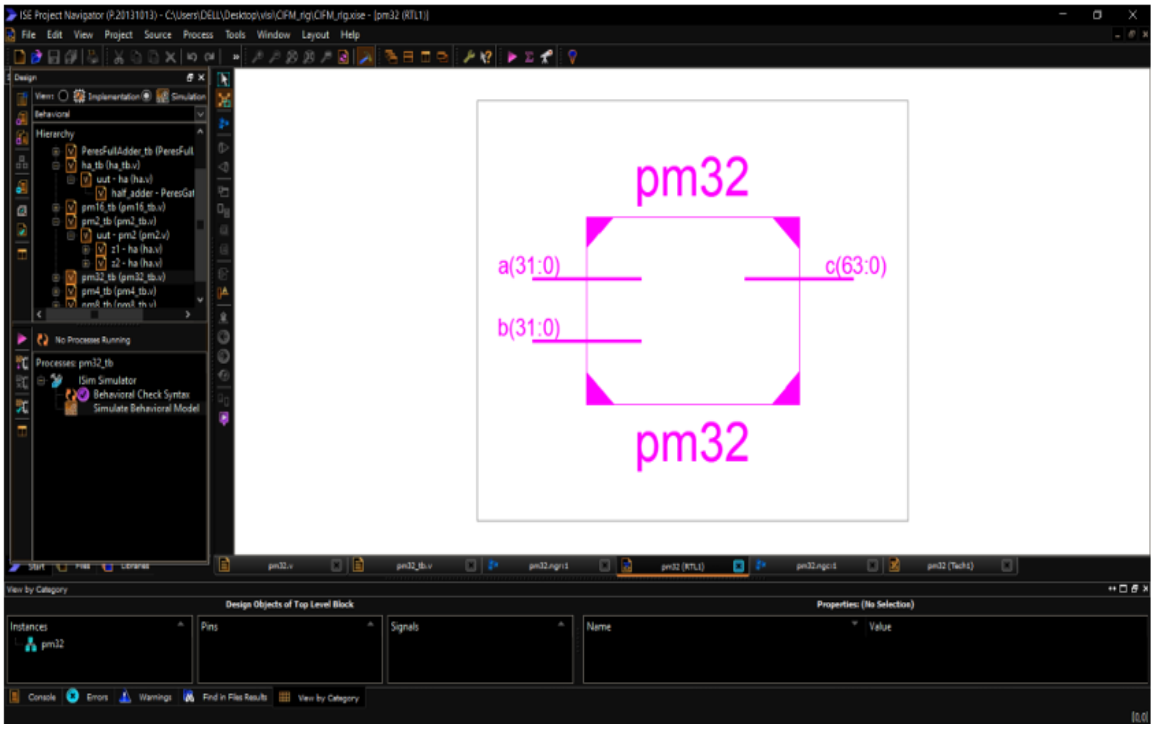


Fig.7.2 RTL Schematic of 32bit Multiplier using Peres Gate

4.8.1 RTL SCHEMATIC OF 32BIT MULTIPLIER

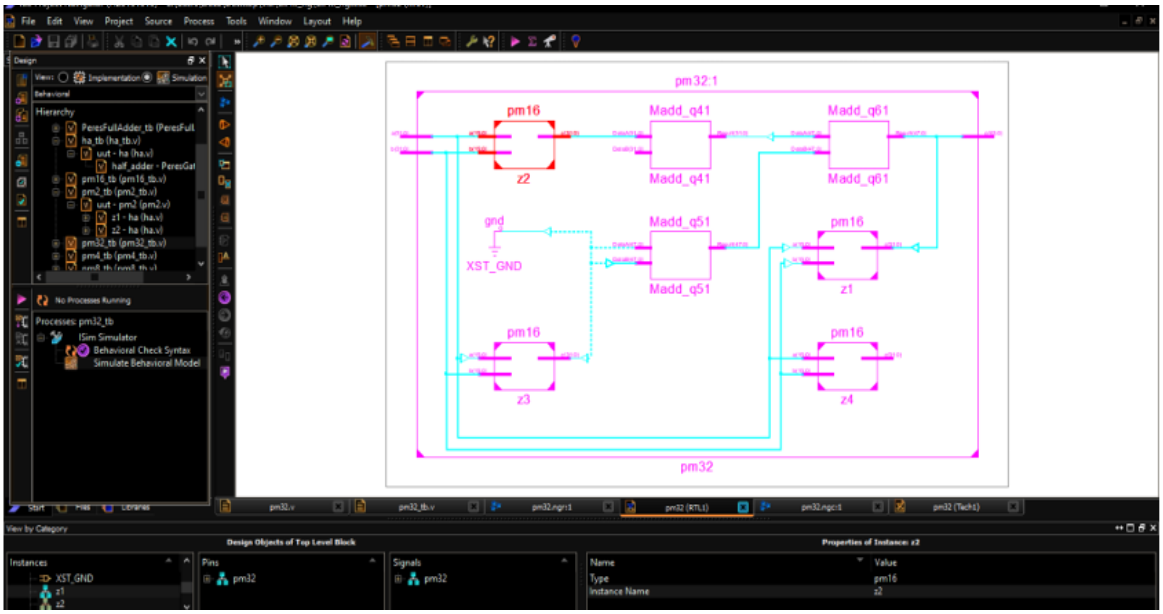


Fig.7.3 RTL Schematic of 32 bit Multiplier using Peres Gate

4.8.2 OUTPUT WAVEFORM OF 32BIT MULTIPLIER

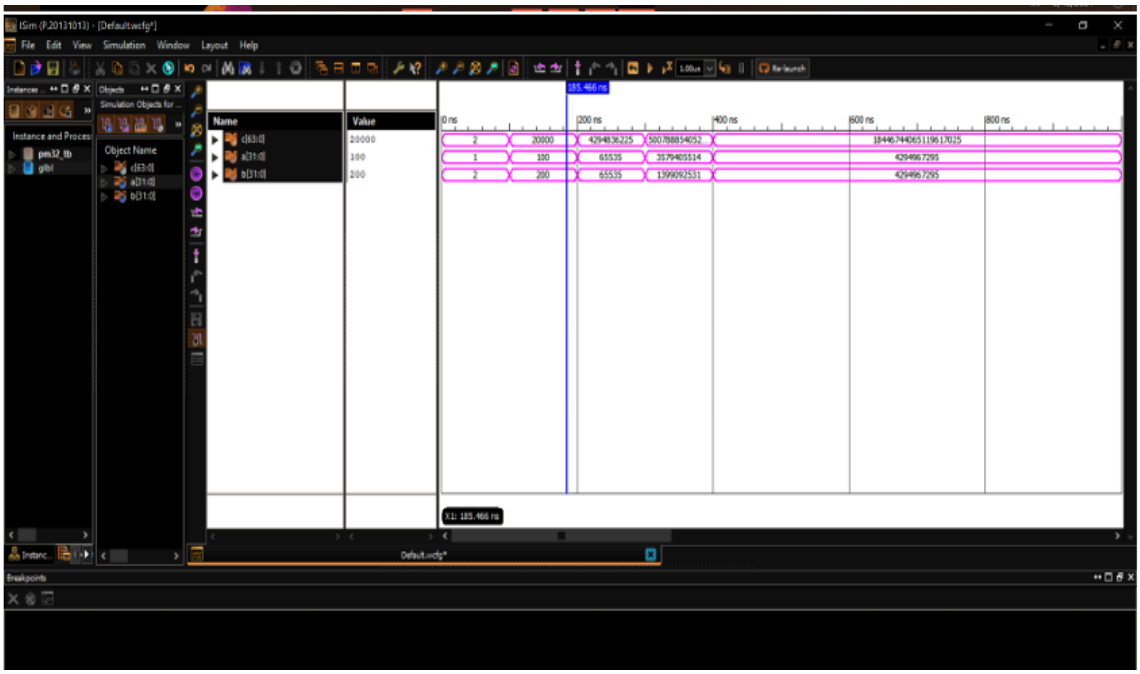


Fig.7.4 Output Waveform of 32bit Multiplier using Peres Gate

4.8.3 SYNTHESIS REPORT OF 32BIT MULTIPLIER

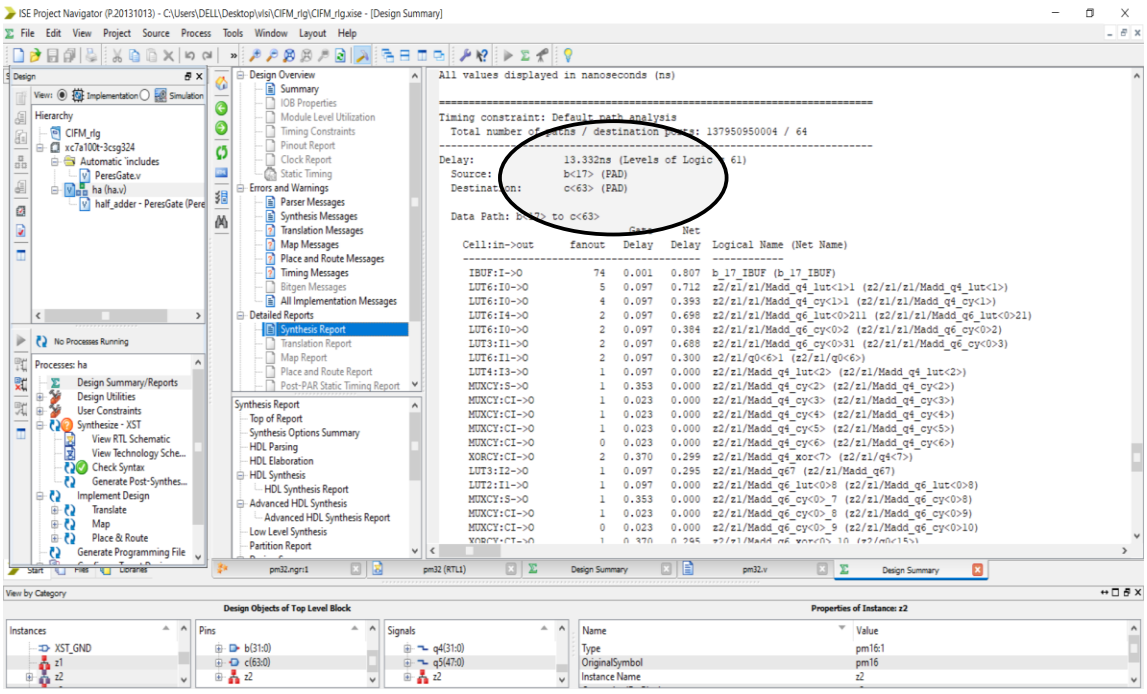


Fig.7.5 Synthesis Report of 32bit Multiplier using Peres Gate

FLOATING POINT MULTIPLIER

4.9 SIGN BIT IDENTIFICATION

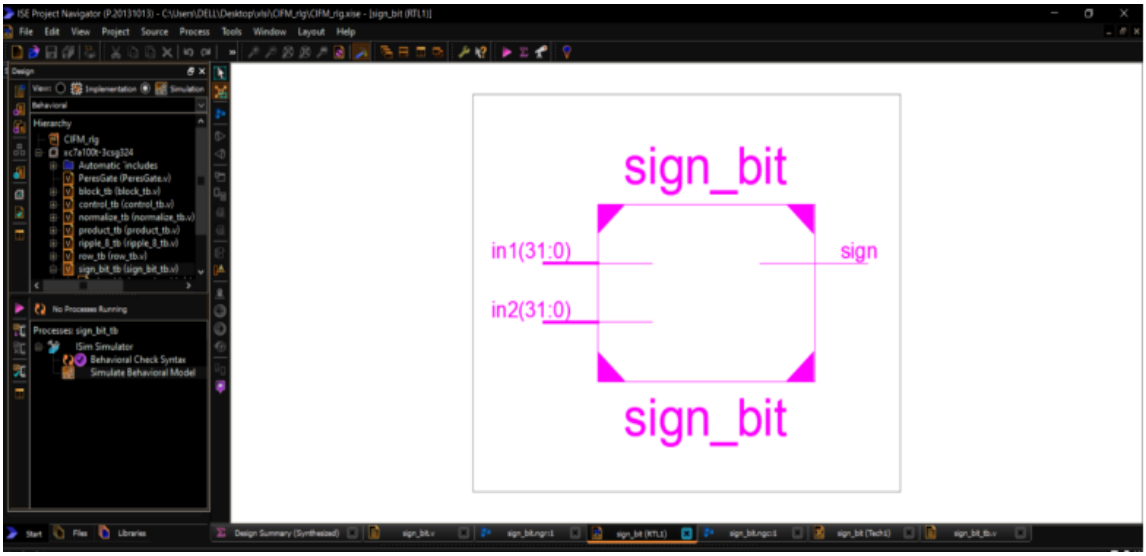


Fig.7.6 Design of sign bit

4.9.1 RTL SCHEMATIC OF SIGN BIT

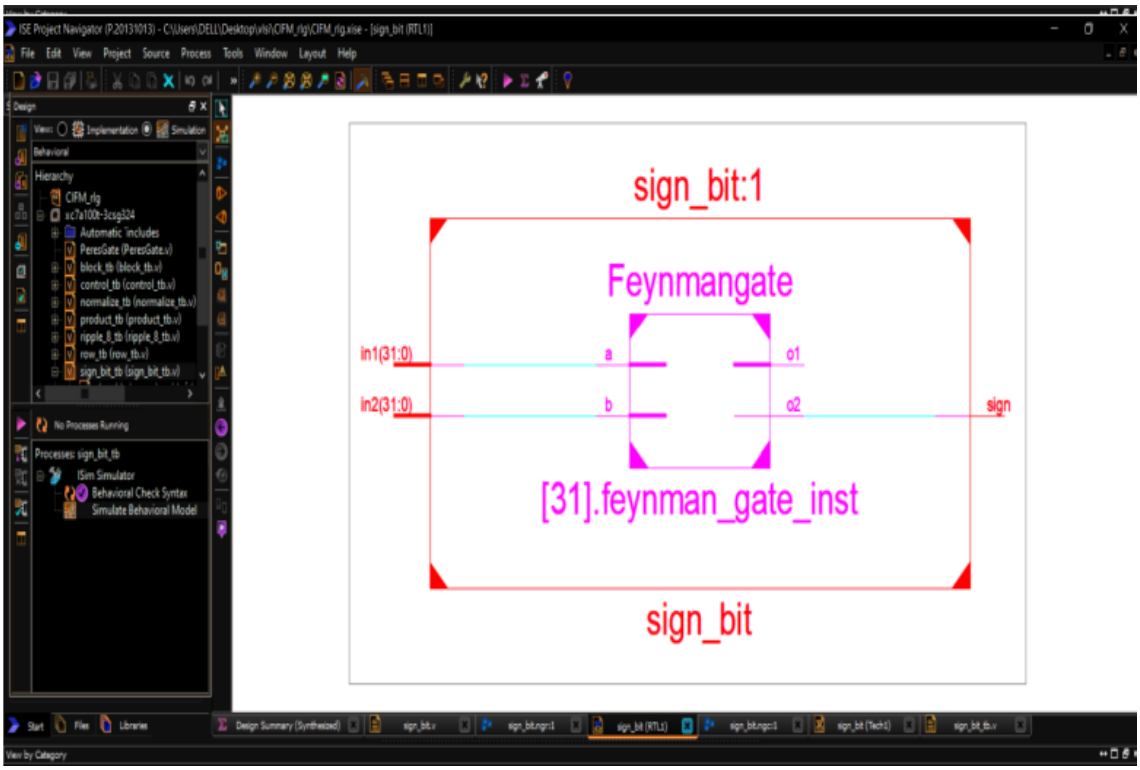


Fig.7.7 RTL schematic of sign bit

4.9.2 TECHNOLOGY SCHEMATIC OF SIGN BIT

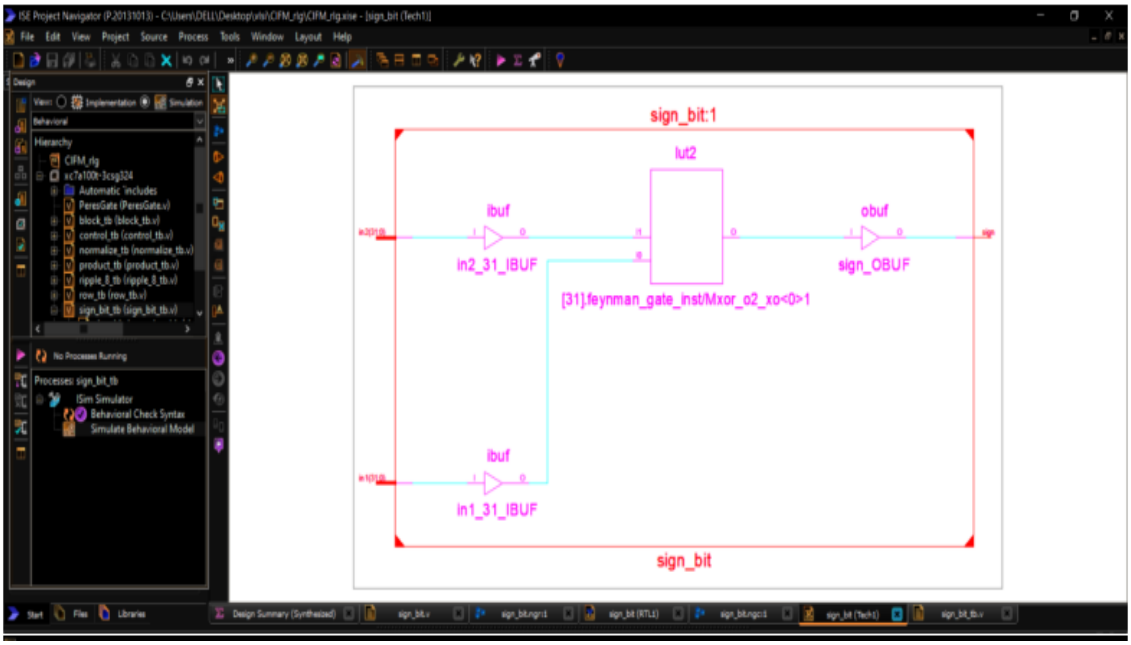


Fig.7.8 Technology schematic of sign bit

4.9.3 OUTPUT WAVEFORM OF SIGN BIT

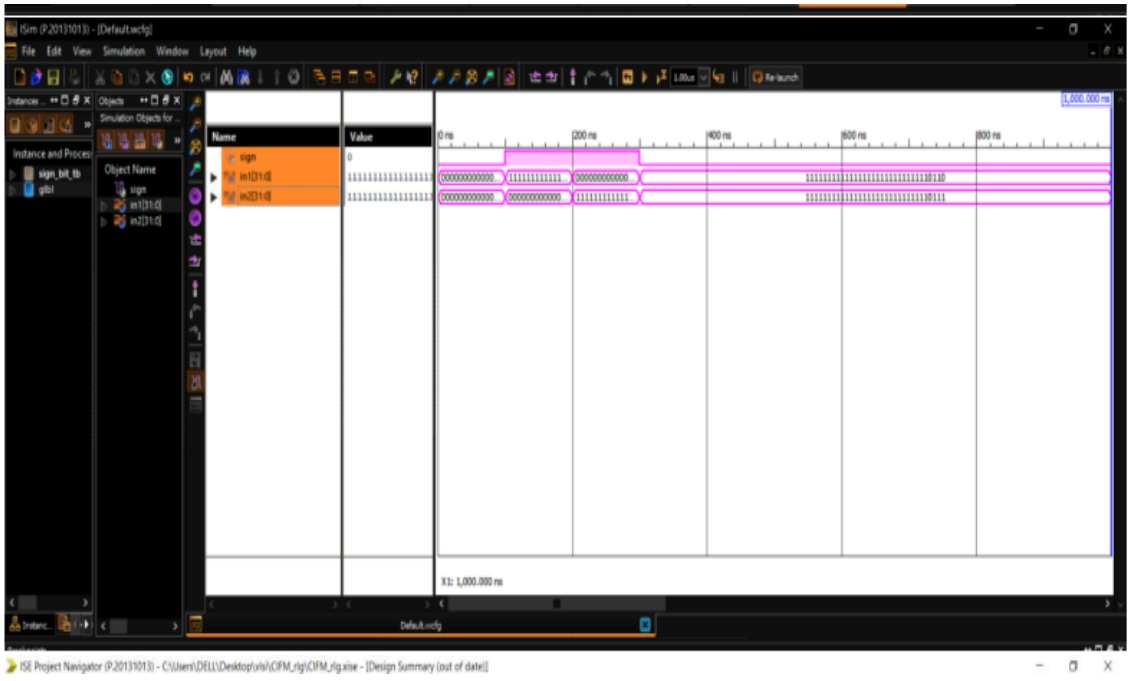


Fig.7.9 output waveform of sign bit

Adding Exponents:

4.10 FULL ADDER USING HALF ADDER

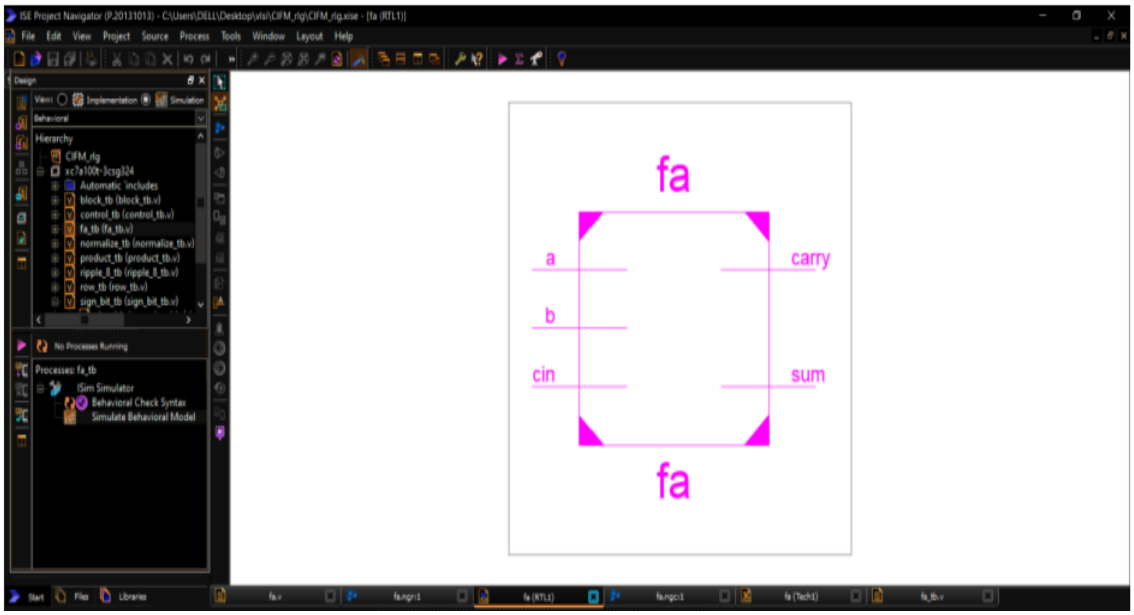


Fig.8 Design of Full adder using Half adder

4.10.1 RTL SCHEMATIC OF FULL ADDER

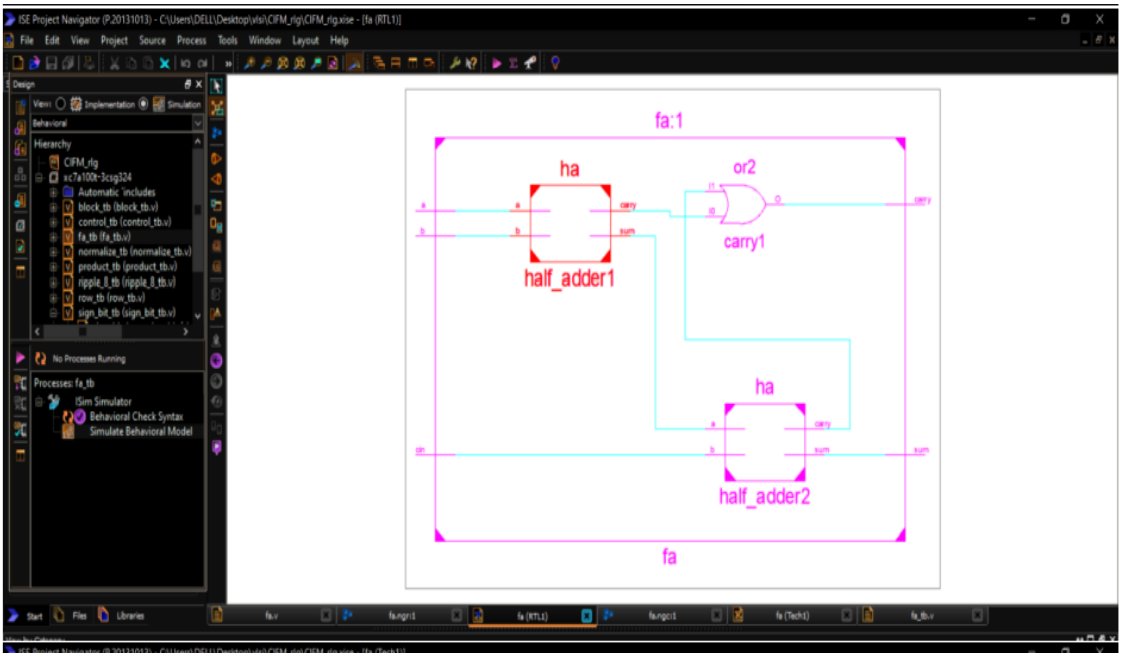


Fig.8.1 RTL schematic of Full adder

4.10.2 TECHNOLOGY SCHEMATIC OF FULL ADDER

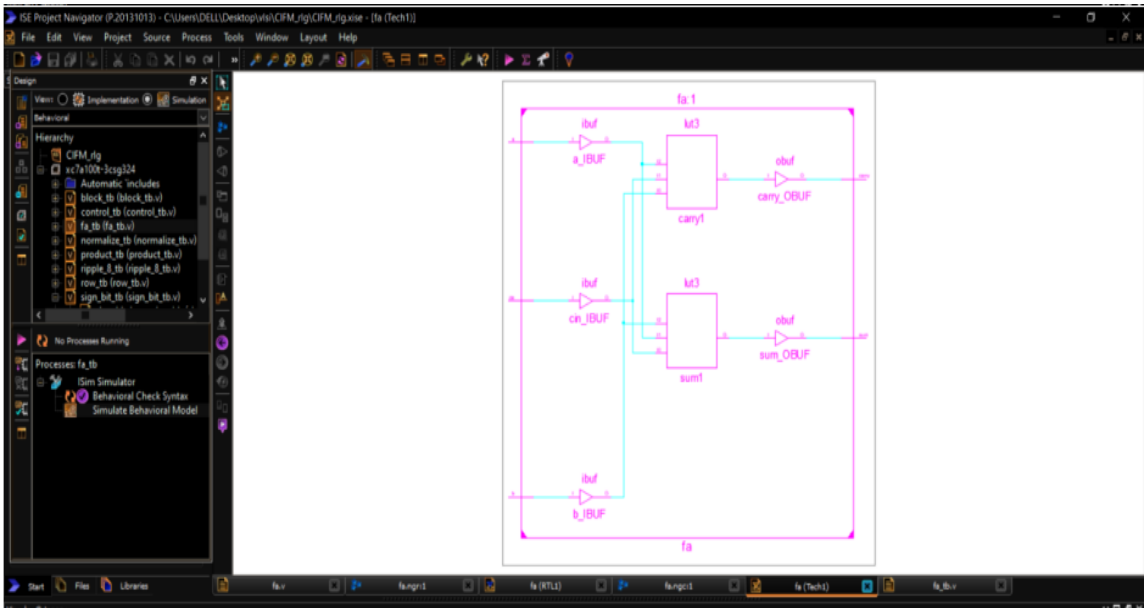


Fig.8.2 Technology Schematic of Full Adder

4.10.3 OUTPUT WAVE FORM OF FULL ADDER USING HALF ADDER

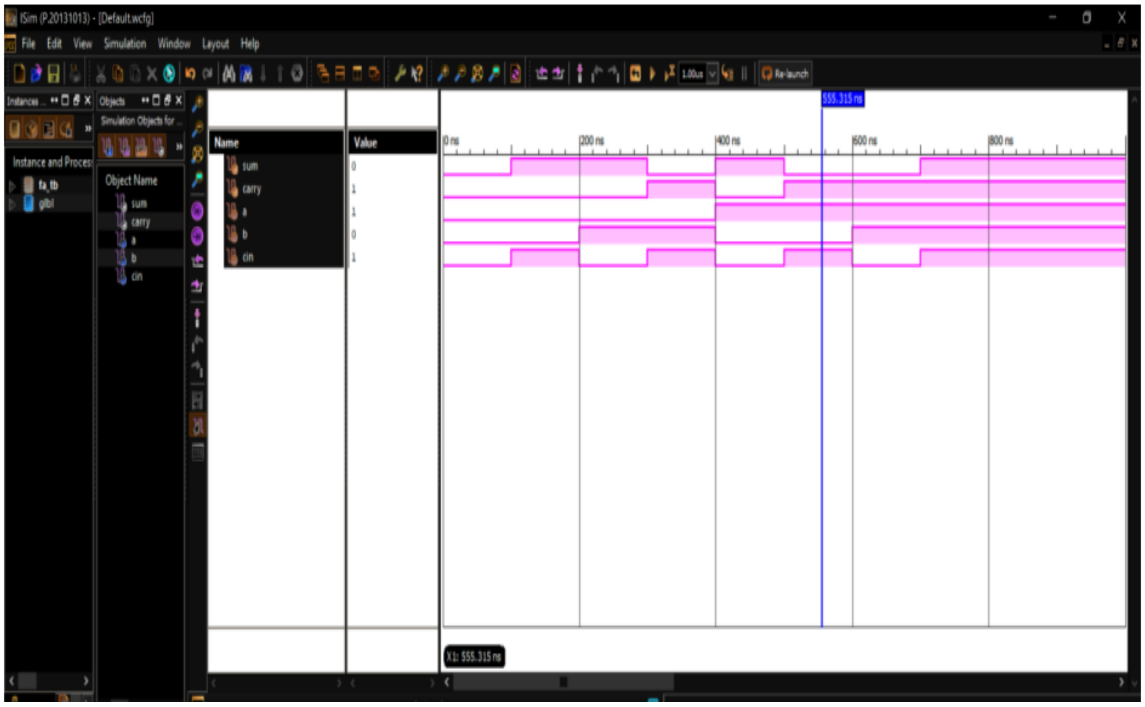


Fig.8.3 Output waveform of Full Adder

4.11 DESIGN OF 8-BIT RIPPLE CARRY ADDER

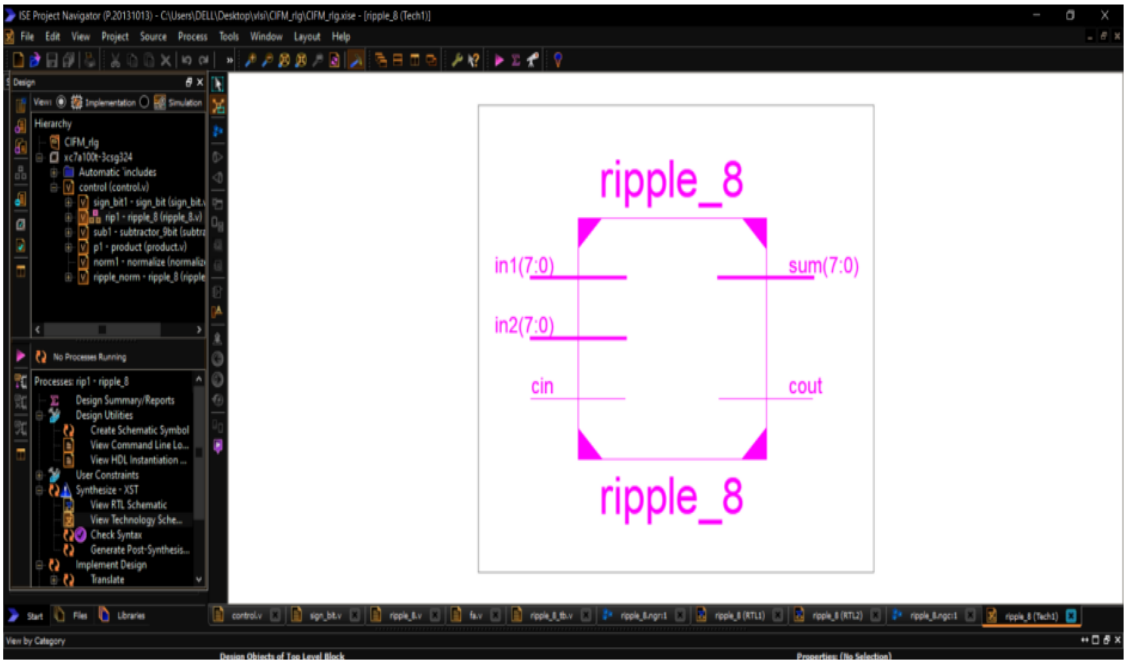


Fig.8.4 Design of 8 bit ripple carry adder

4.11.1 RTL SCHEMATIC OF RIPPLE CARRY ADDER

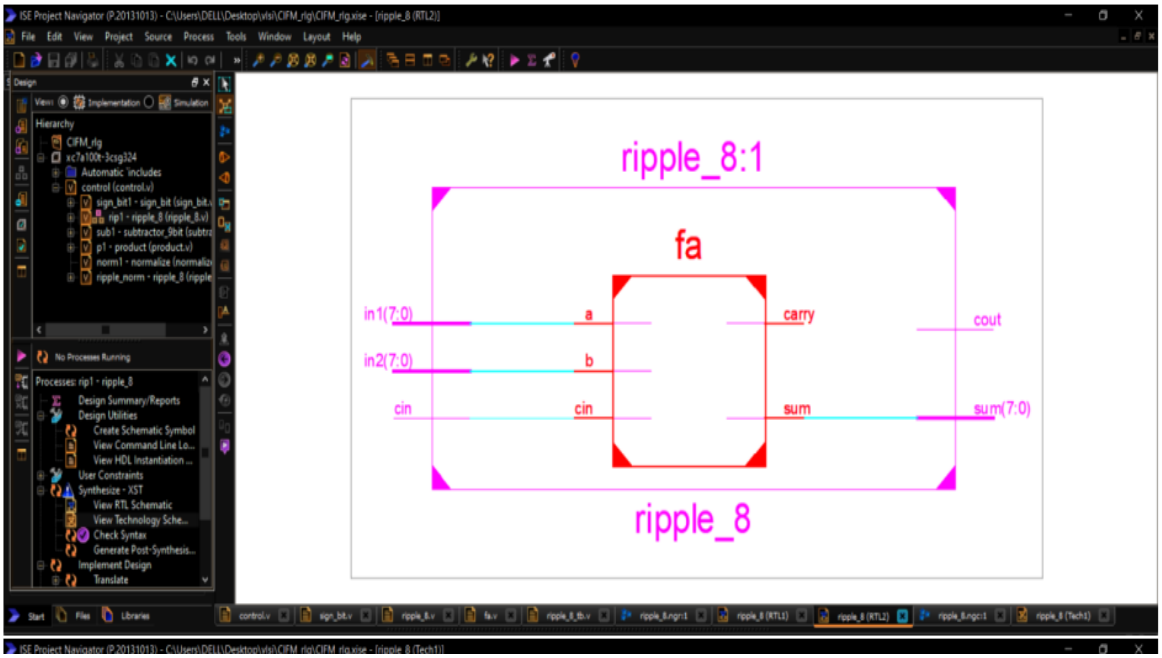


Fig.8.5 RTL schematic of ripple carry adder

4.11.2 TECHNOLOGY SCHEMATIC OF RIPPLE CARRY ADDER

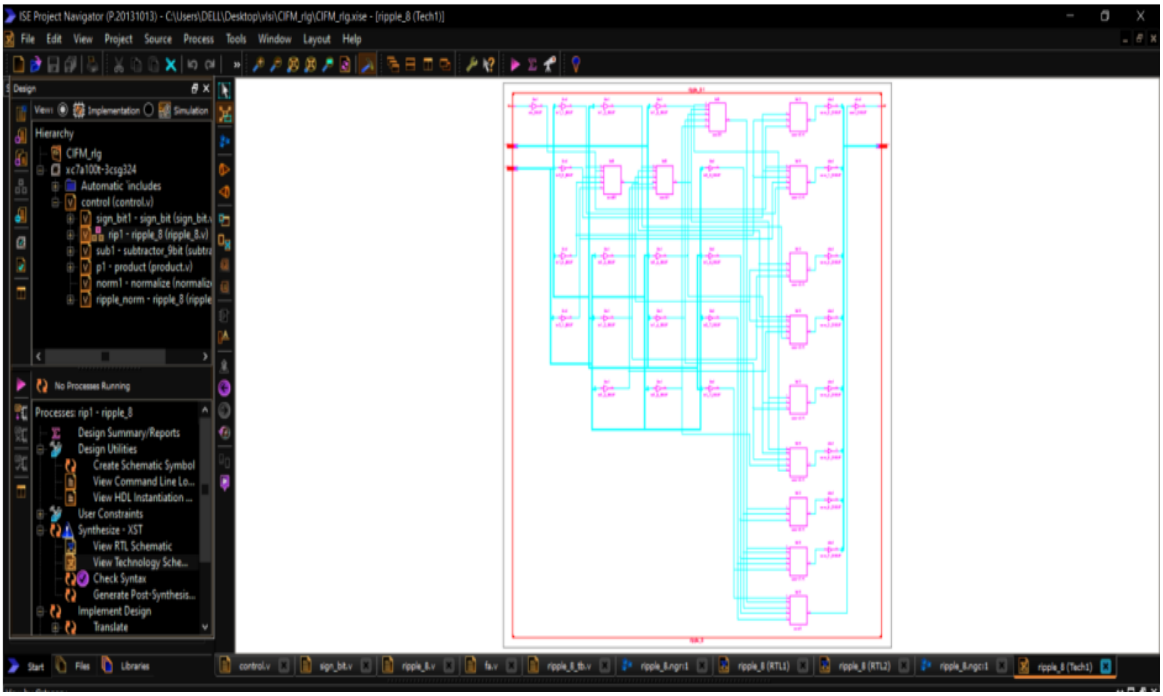


Fig.8.6 Technology schematic of ripple carry adder

4.11.3 OUTPUT WAVEFORM OF RIPPLE CARRY ADDER

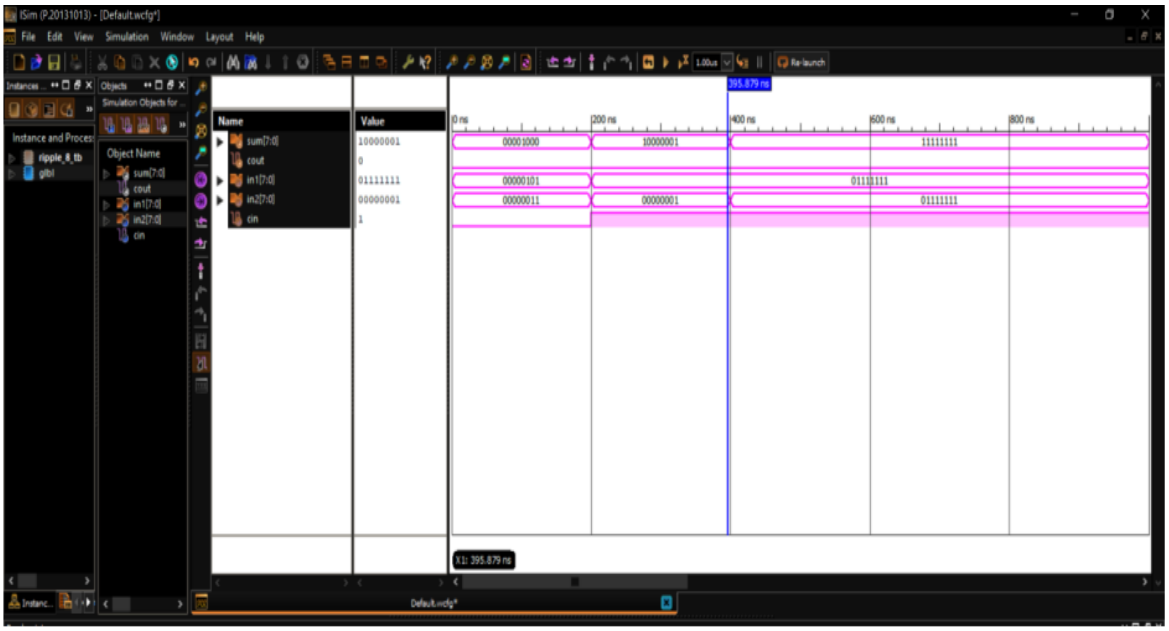


Fig.8.7 output waveform of ripple carry adder

4.12 Subtracting the Bias:

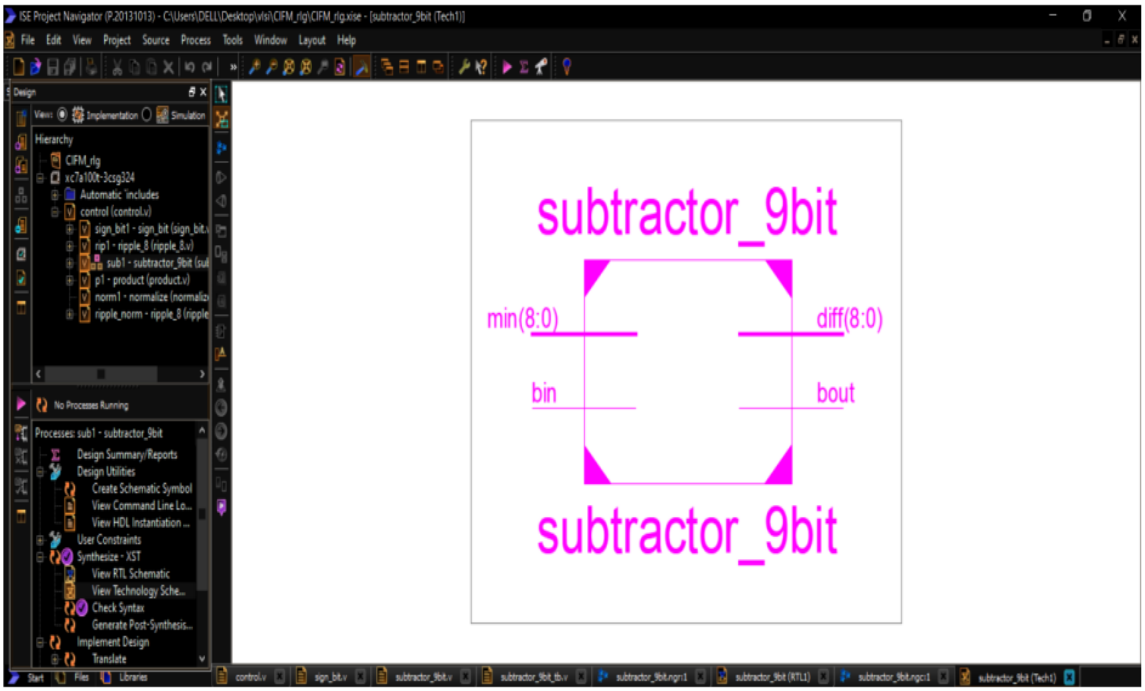


Fig.8.8 Design of subtractor

4.12.1 RTL SCHEMATIC OF SUBTRACTOR

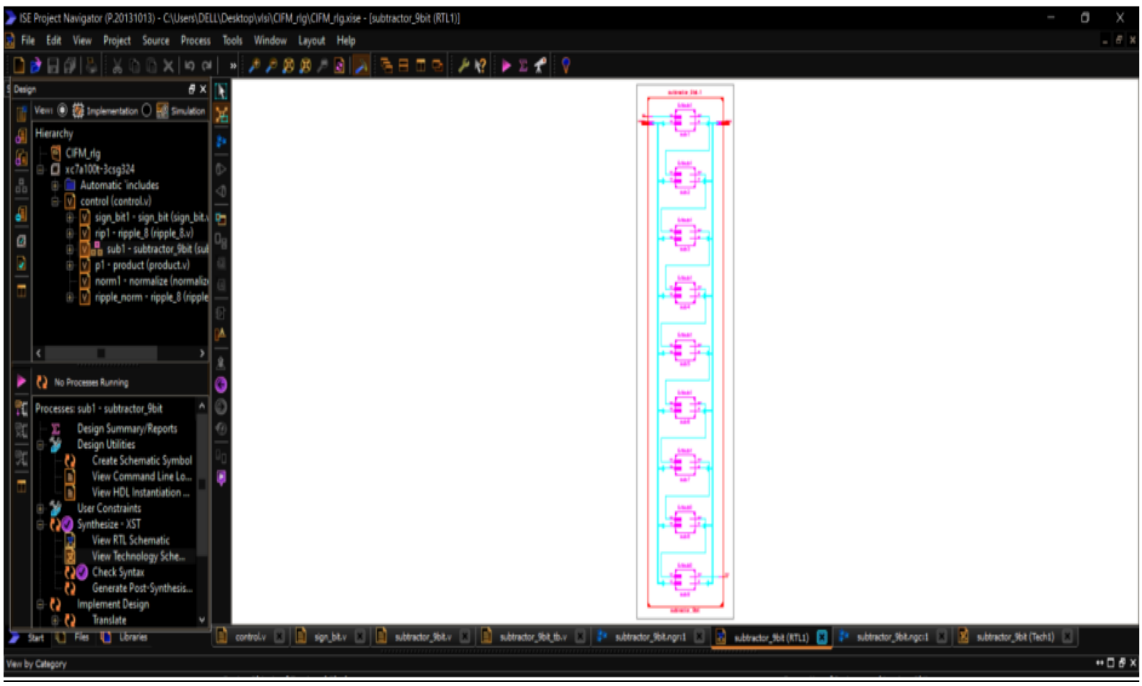


Fig.8.9 RTL Schematic of Subtractor

4.12.2 TECHNOLOGY SCHEMATIC OF SUBTRACTOR

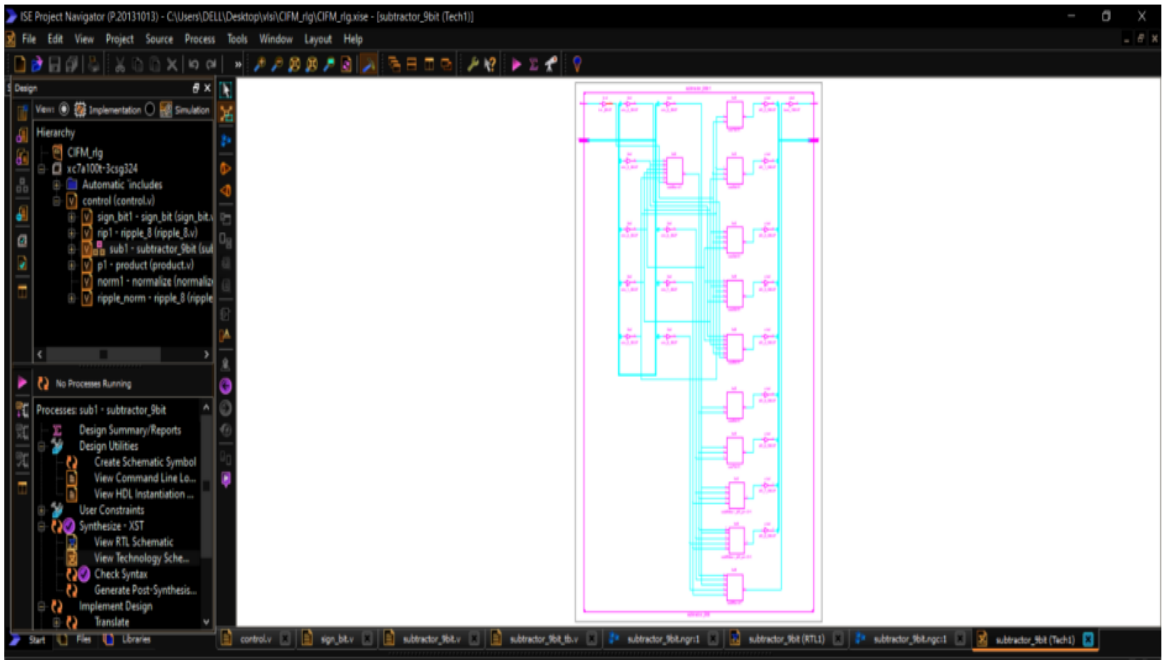


Fig.9 Technology Schematic Of Subtractor

4.12.3 OUTPUT WAVEFORM OF SUBTRACTOR

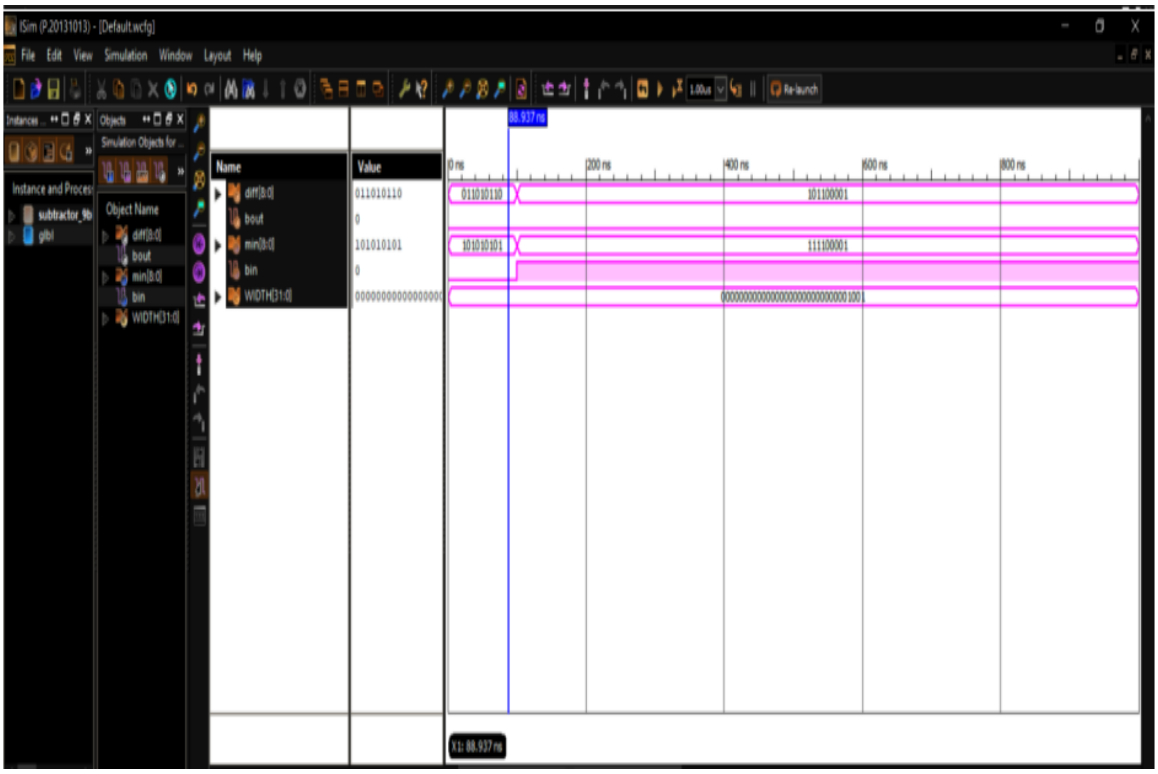


Fig.9.1 Output Waveform of Subtractor

4.13 Multiplying the Mantissa Block level

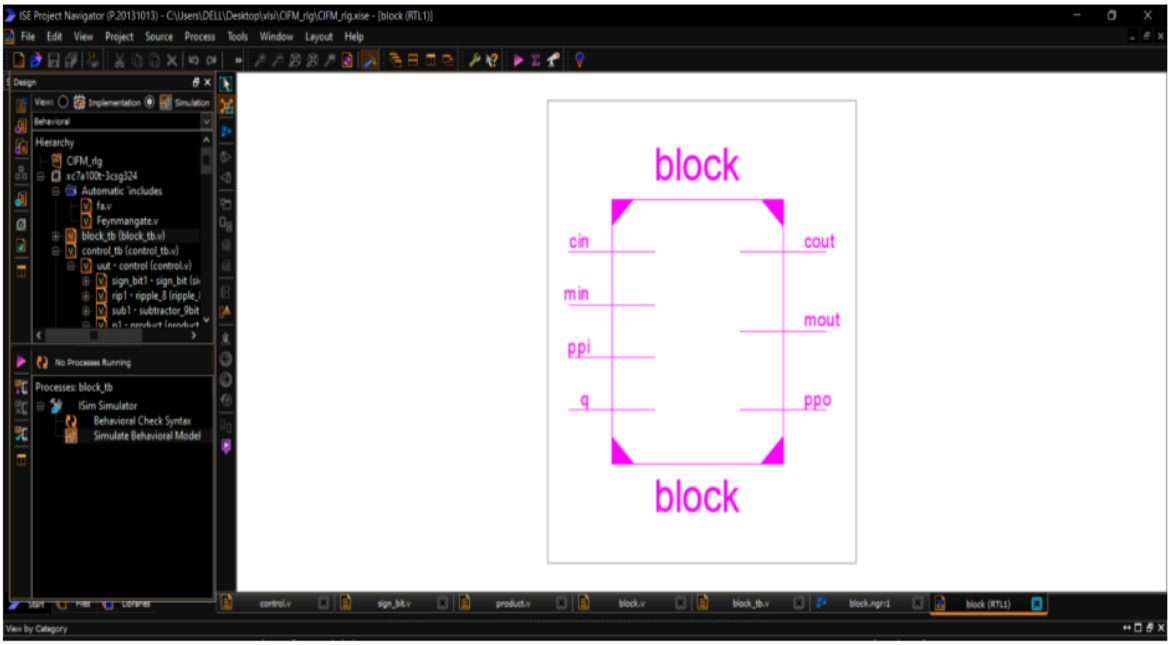


Fig.9.2 Design of BLOCK

4.13.1 RTL SCHEMATIC OF MANTISSA BLOCK LEVEL

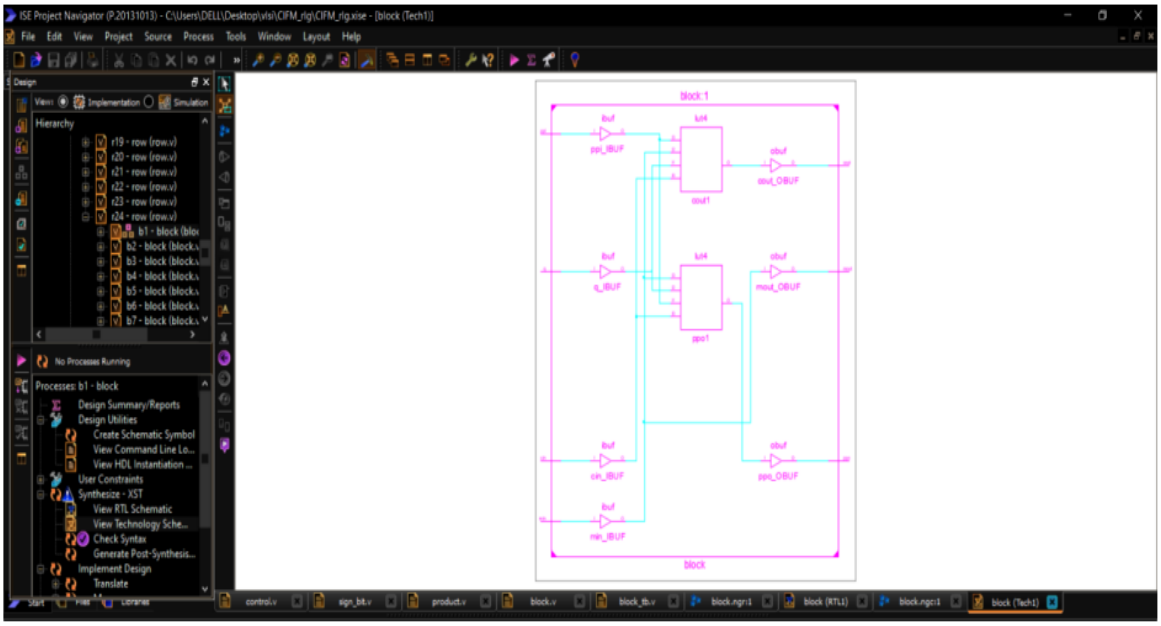


Fig.9.3 RTL Schematic of Mantissa

4.13.2 OUTPUT WAVEFORM OF MANTISSA BLOCK LEVEL

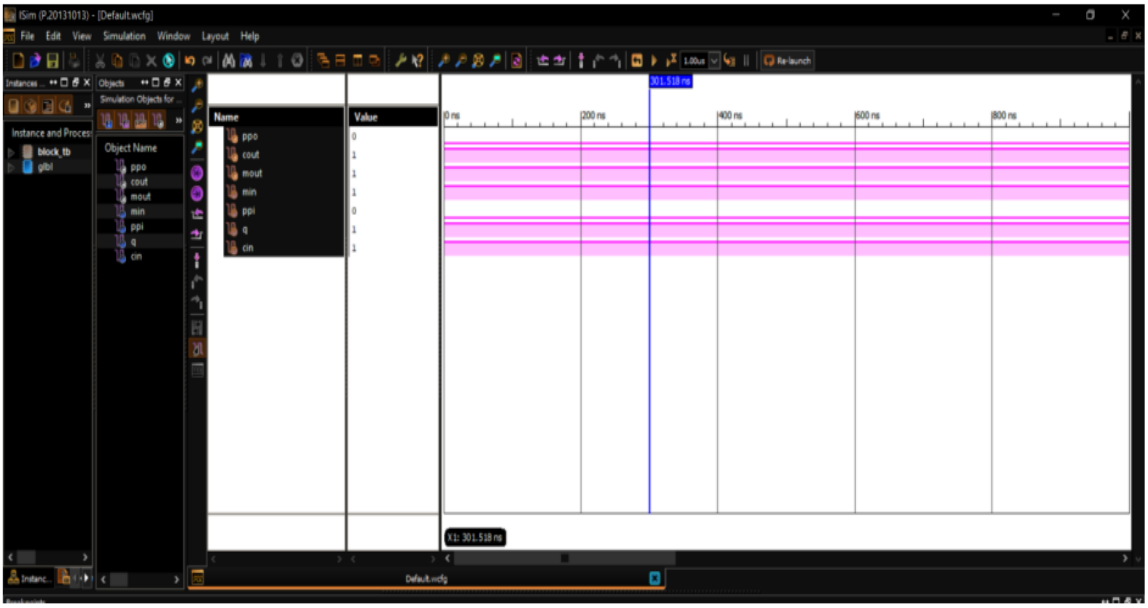


Fig.9.4 Waveform of Mantissa

4.14 DESIGN OF ROW LEVEL MULTIPLICATION

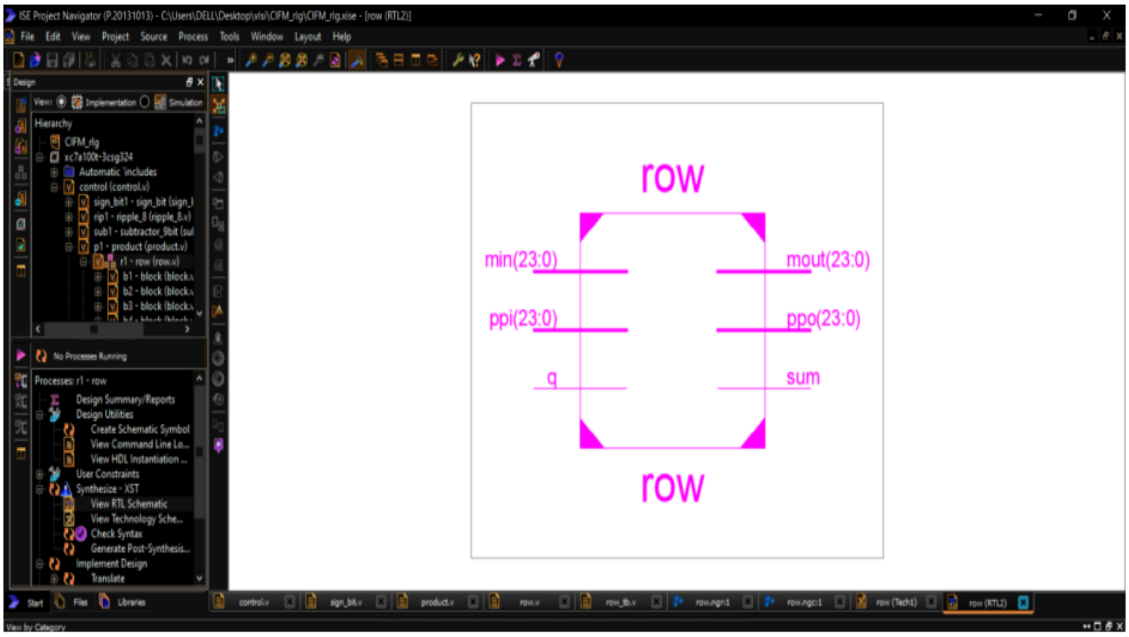


Fig.9.5 Design of Row level Multiplication

4.14.1 RTL SCHEMATIC OF ROW LEVEL

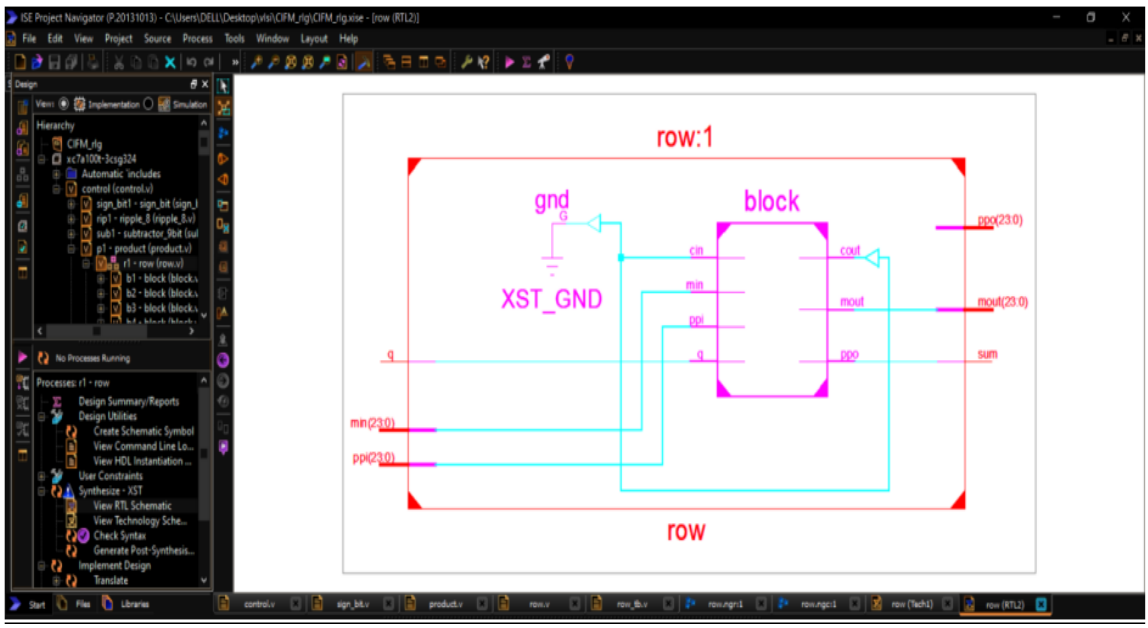


Fig.9.6 RTL Schematic of Row Level Multiplication

4.14.2 TECHNOLOGY SCHEMATIC OF ROW LEVEL MULTIPLICATION

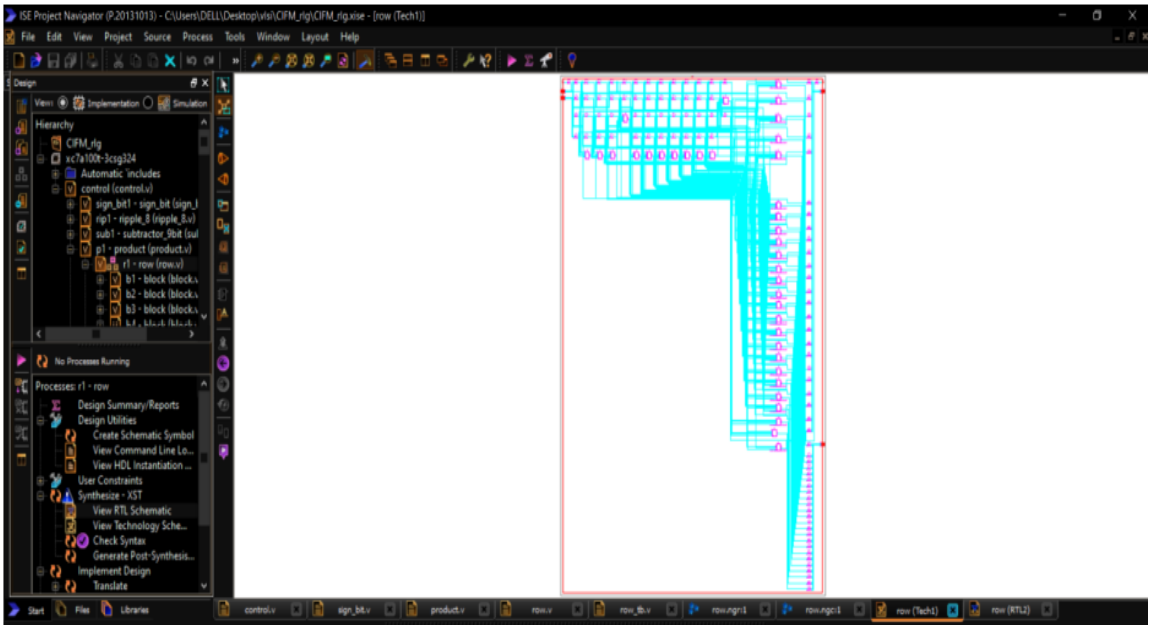


Fig.9.7 Technology Schematic of Row Level Multiplication

4.14.3 OUTPUT WAVEFORM OF ROW LEVEL MULTIPLICATION

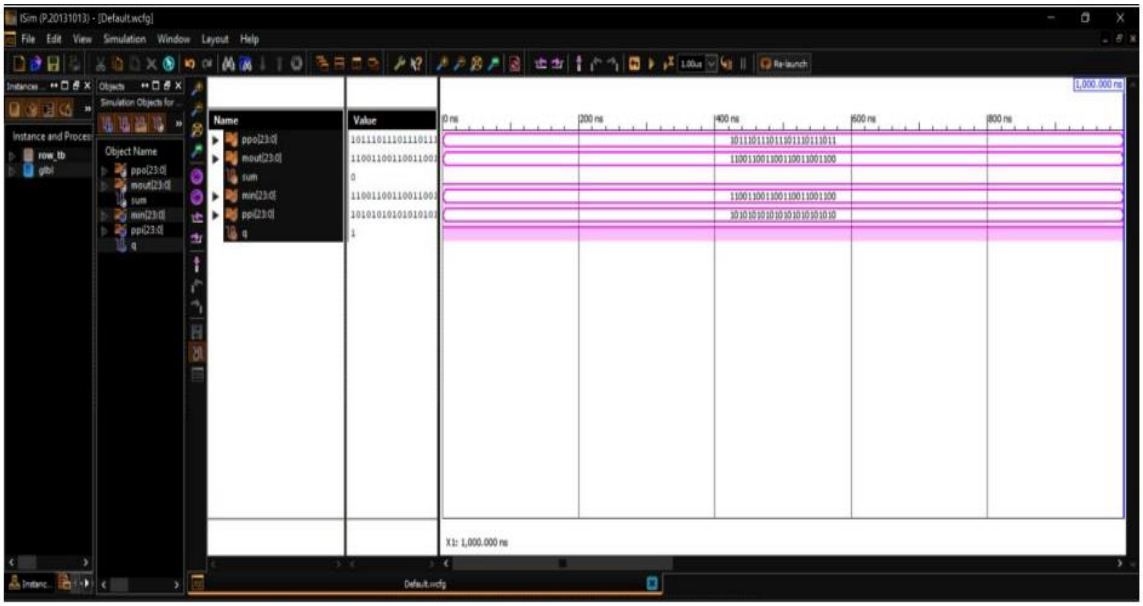


Fig.9.8 Waveform of Row Level Multiplication

4.15 DESIGN OF FINAL PRODUCT

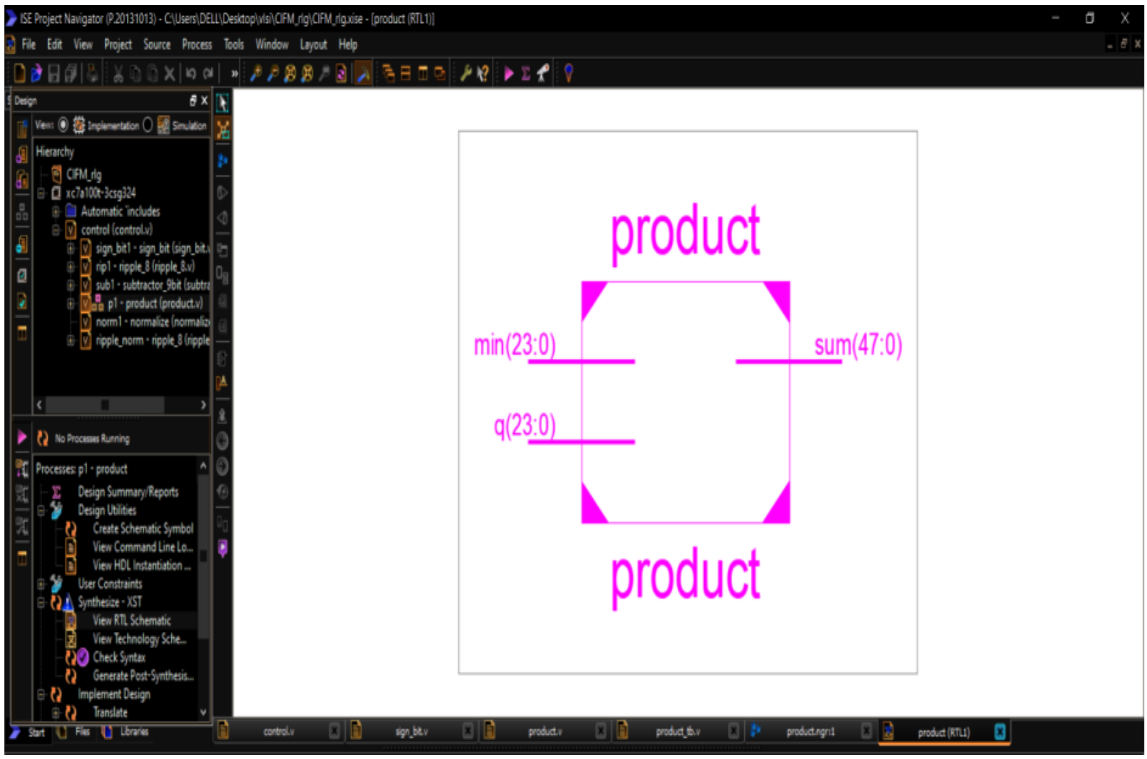


Fig.9.9 Design of Final Product

4.15.1 RTL SCHEMATIC OF PRODUCT

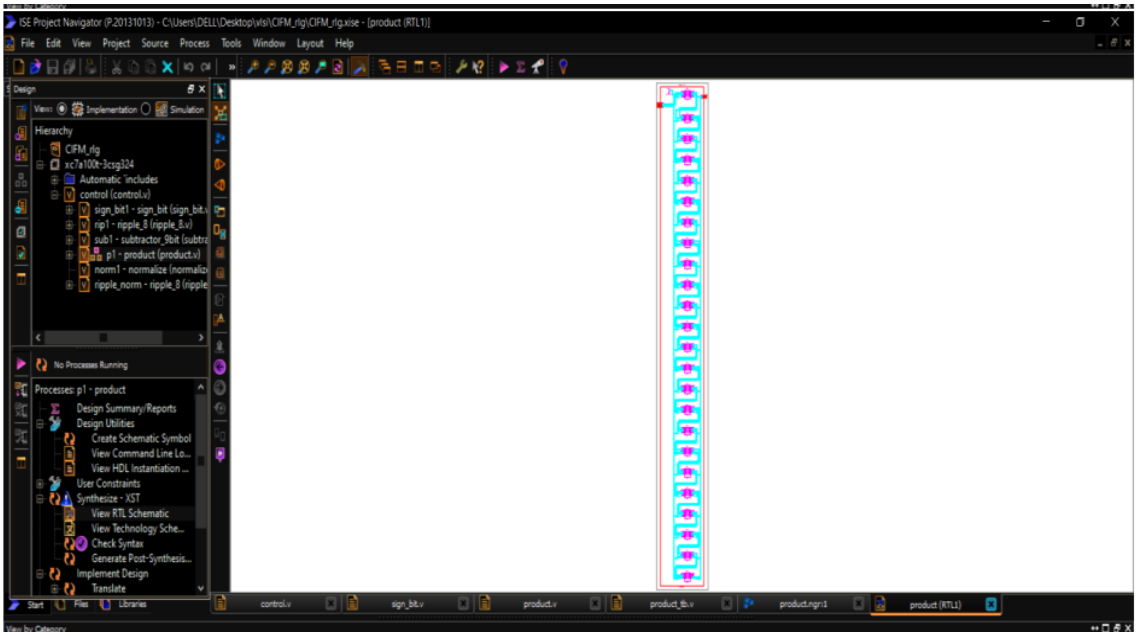


Fig.10 RTL of Product

4.15.2 TECHNOLOGY SCHEMATIC OF PRODUCT

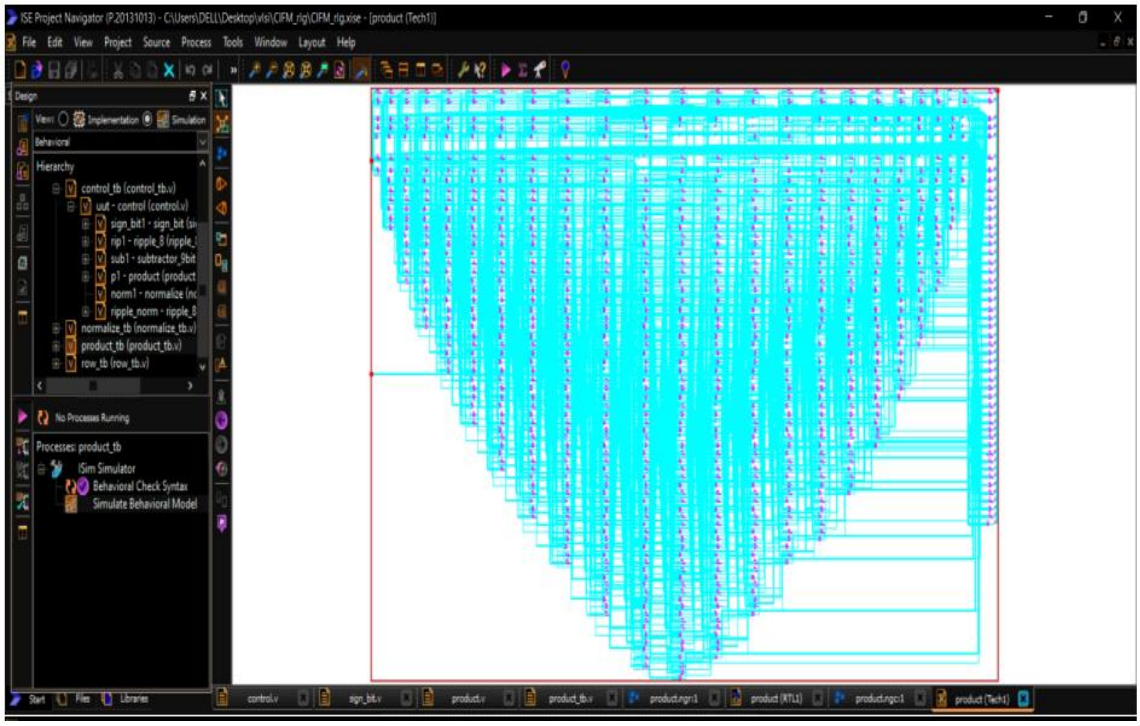


Fig.10.1 Technology Schematic of Product

4.15.3 OUTPUT WAVEFORM OF PRODUCT

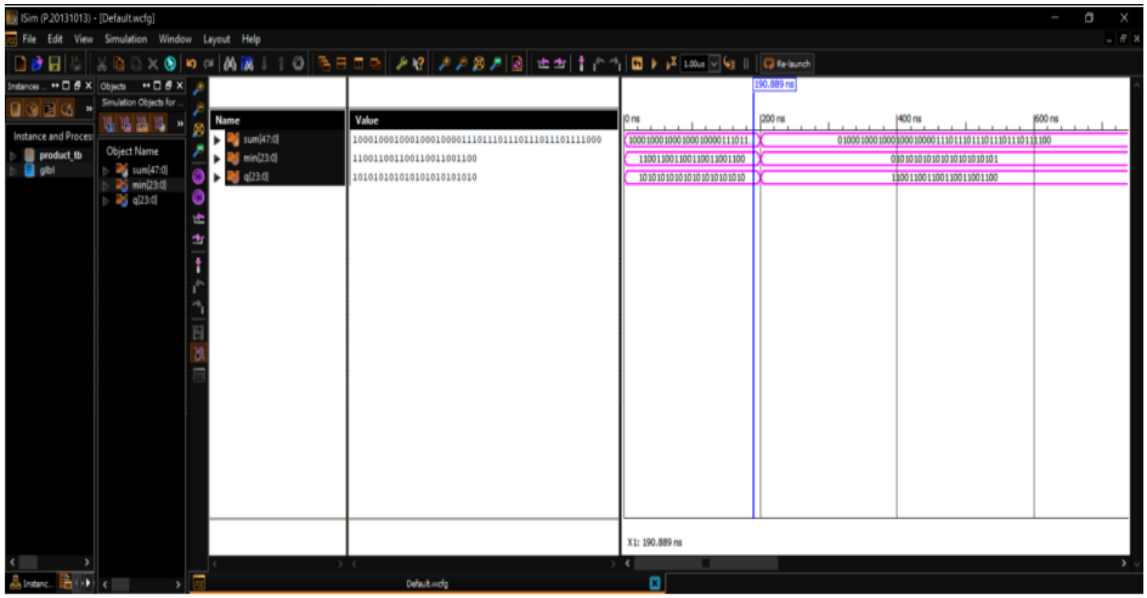


Fig.10.2 Waveform of Product

4.16 DESIGN OF NORMALIZER

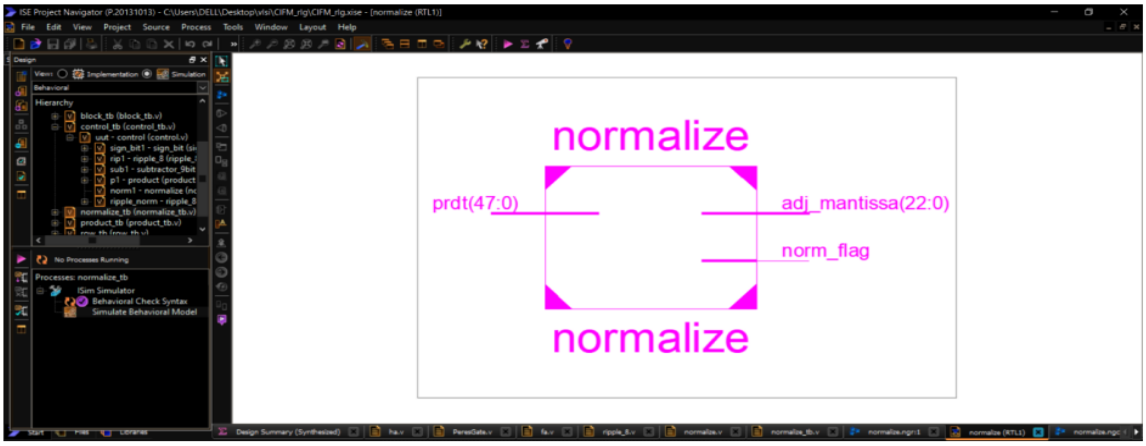


Fig.10.3 Design of Normalizer

4.16.1 RTL SCHEMATIC OF NORMALIZER

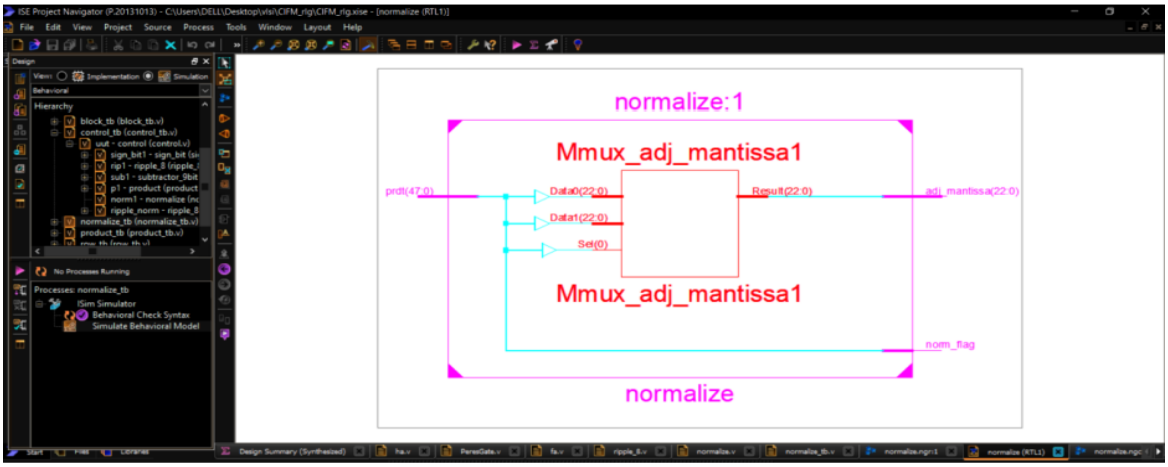


Fig.10.4 RTL of Normalizer

4.17 Floating Pointing Multiplier

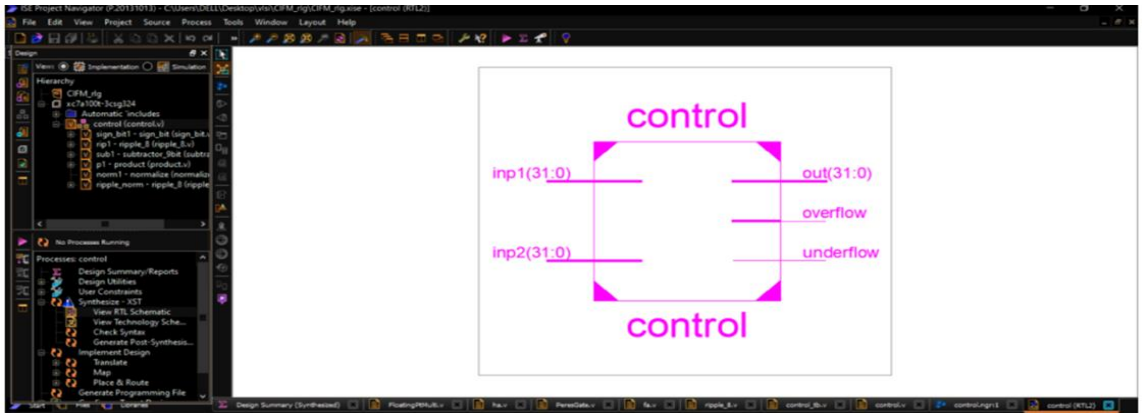


Fig.10.5 Design of Floating point Multiplier

4.17.1 RTL SCHEMATIC OF FLOATING POINT MULTIPLIER

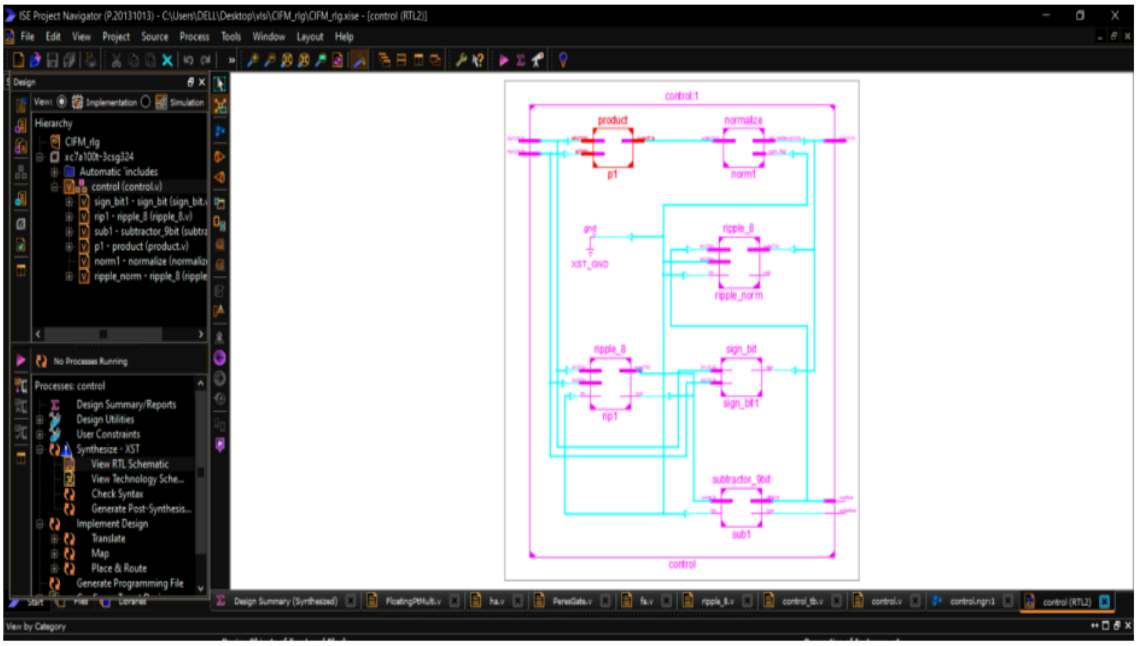


Fig.10.6 RTL Schematic of Floating Point Multiplier

4.17.2 TECHNOLOGY SCHEMATIC OF FLOATING POINT MULTIPLIER

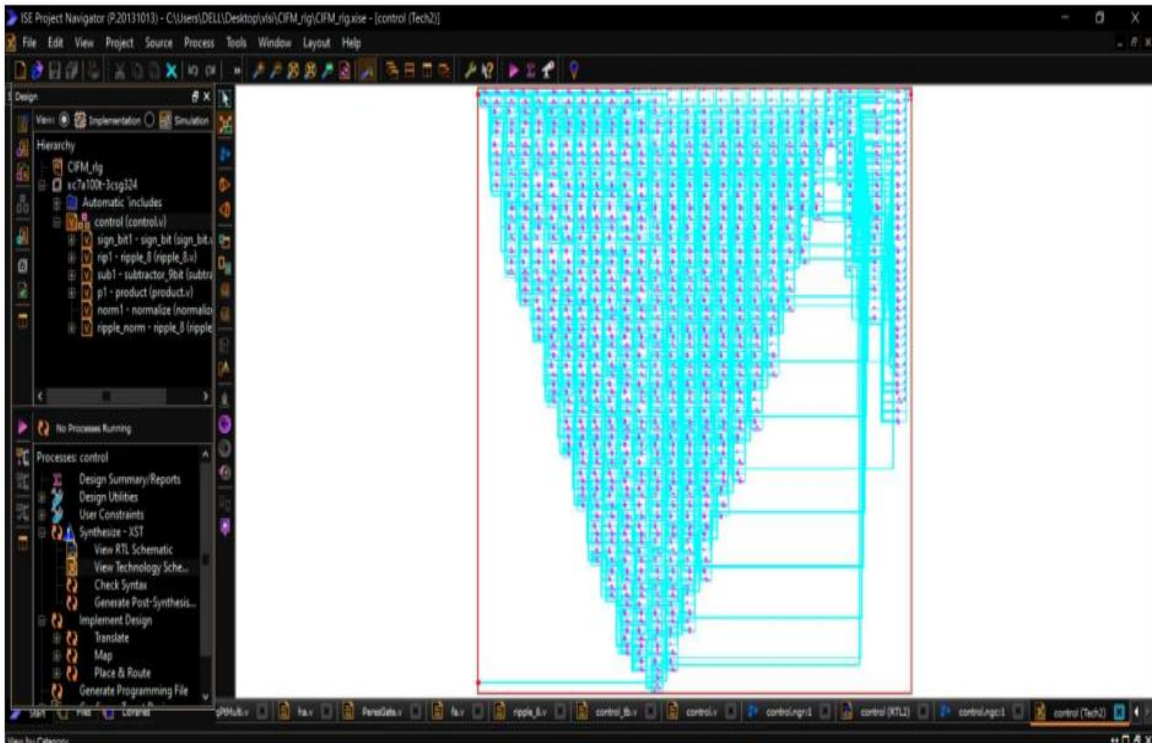


Fig.10.7 Technoogy Schematic of Floating point Multiplier

4.17.3 OUTPUT WAVEFORM OF FLOATING POINT MULTIPLIER

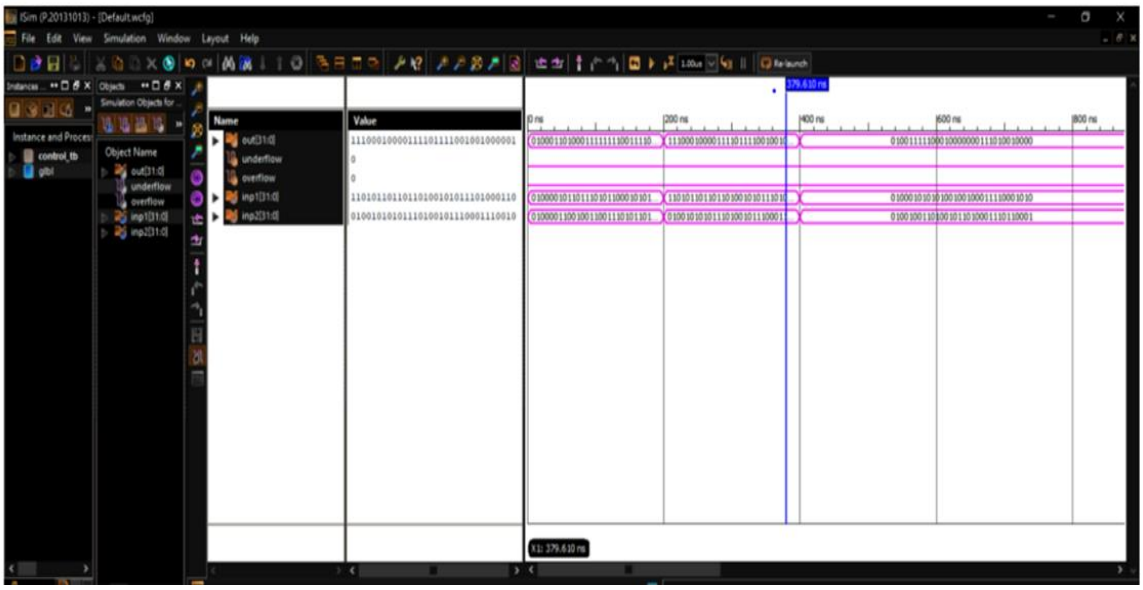


Fig.10.8 Waveform of Floating Point Multiplier

4.18 Design of 32bit Combined integer and Floating Point Multiplier :

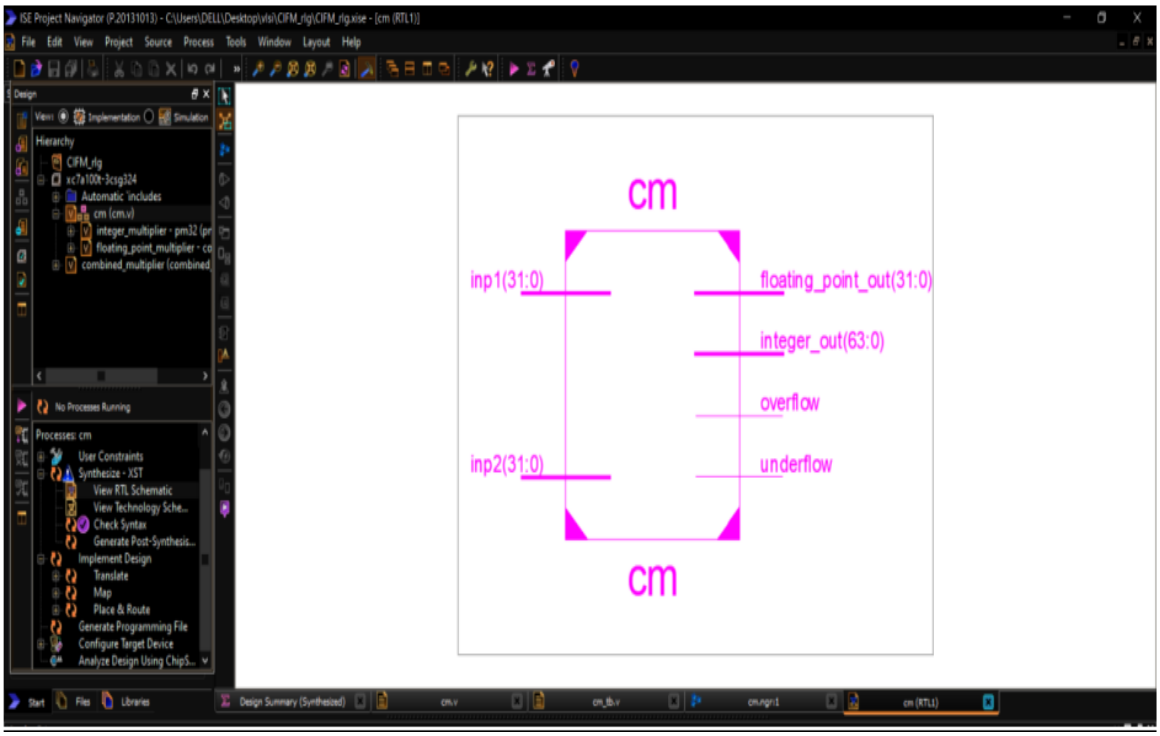


Fig.10.9 Design of 32bit Combined integer and Floating Point Multiplier

4.18.1 OUTPUT WAVEFORM OF 32BIT MULTIPLIER

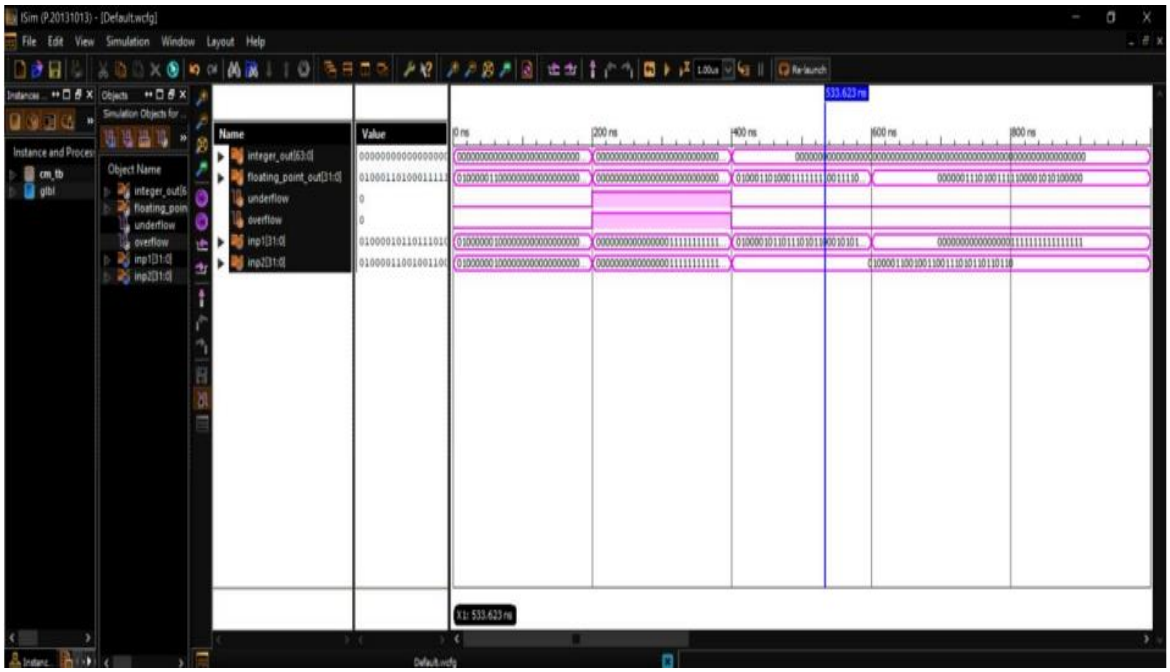


Fig.11 Output Waveform of 32bit Combined Integer and Floating Point Multiplier

4.18.2 SYNTHESIS REPORT OF COMBINED INTEGER AND FLOATING POINT MULTIPLIER

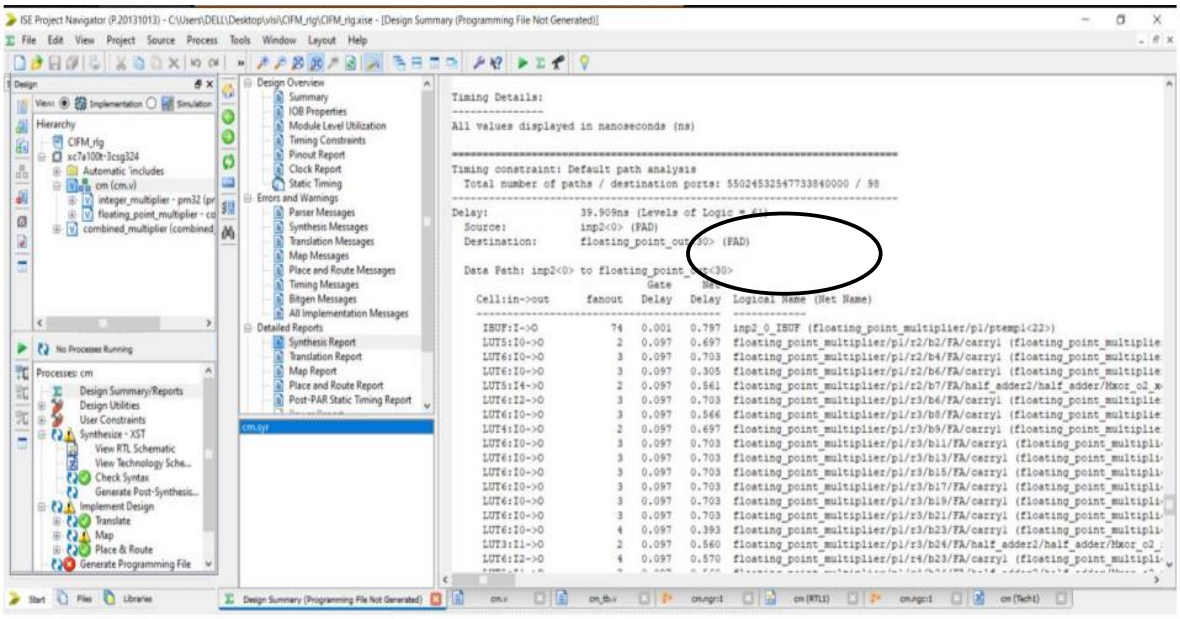


Fig.11.1 Delay obtained for Combined Integer and Floating Point Multiplier

COMPARISON TABLE:

Logic Gates

CIRCUITS	DELAY
Logic gates	9.23ns
Full Adder	9.033ns
2bit Multiplier	9.073ns
4bit Multiplier	21.53ns
8bit Multiplier	27.696ns
16bit Multiplier	36.789ns
32bit Multiplier	46.507ns

Reversible Gates

CIRCUITS	DELAY
Peres Gate	0.889ns
Feynman Gate	0.733ns
TS Gate	0.943ns
Full Adder	0.949ns
2bit Multiplier	0.948ns
4bit Multiplier	3.496ns
8bit Multiplier	6.645ns
16bit Multiplier	9.680ns
32bit Multiplier	13.332ns
Floating Point Multiplier	40.390ns
Combined integer and floating point multiplier	39.909ns

Comparison of Performance Characteristics in Terms of Delay

Name of the Multiplier	Vendor	Delay(n Sec)
Proposed 4bit Reversible Vedic Multiplier (Reference Paper 1-4244-0173)	Xilinx	5.353ns
4bit multiplier using Reversible logic	Xilinx	3.496ns

APPLICATIONS:

Quantum Computing: Reversible logic is fundamental to quantum computing, where operations must be reversible to maintain coherence and avoid energy loss. Our multiplier could serve as a key component in quantum circuits for arithmetic operations, enabling more efficient quantum algorithms for tasks such as factoring large numbers and simulating quantum systems.

Graphics Processing Units (GPUs): GPUs are essential for rendering complex graphics and performing parallel computations in applications like computer-aided design (CAD), gaming, and virtual reality. Our multiplier can improve the efficiency of GPU architectures by optimizing the multiplication operations required for graphics rendering and general-purpose computing tasks.

Machine Learning and Artificial Intelligence: Many machine learning algorithms, such as neural networks, rely heavily on matrix multiplication operations, which often involve both integer and floating-point calculations. Our multiplier can accelerate the training and inference processes of machine learning models, leading to faster and more efficient AI systems.

Low-Power Computing: Reversible logic gates have the potential to reduce energy consumption in conventional computing systems. By incorporating our multiplier into low-power devices such as smartphones, wearables, and IoT sensors, we can extend battery life and reduce environmental impact without sacrificing computational performance.

Digital Communication Systems: Digital communication systems, including wireless communication, data transmission, and networking, rely on efficient signal processing algorithms for encoding, decoding, and error correction. Our multiplier can accelerate these algorithms by efficiently processing integer and floating-point operations, leading to improved data throughput and reliability.

Digital Signal Processing (DSP): DSP algorithms often involve complex mathematical operations where both integer and floating-point calculations are required. Our multiplier can accelerate signal processing tasks such as filtering, modulation, and spectral analysis by efficiently handling both types of operations.

CHAPTER 5

CONCLUSION

In conclusion, this project effectively addressed the complex task of designing a 32-bit reversible combined integer and floating-point multiplier. Through systematic development, a foundation was laid using fundamental reversible logic gates such as Peres and Feynman Gates. Starting from basic components like half adders and full adders and progressing to higher-level multipliers, a robust integer multiplier capable of handling 32-bit operands was constructed. Transitioning to floating-point arithmetic, components were devised to manage the intricacies of floating-point representation, including sign bit calculation, mantissa calculation and exponent calculation.These components were seamlessly integrated to form a comprehensive floating-point multiplier capable of accurately processing floating-point inputs while preserving precision and range. The integration of both integer and floating-point multiplication functionalities ensures versatility and efficiency in handling various input scenarios. The system adeptly produces integer output when dealing with integer inputs, floating-point output when both inputs are floating-point numbers, and correctly computes floating-point output when one input is an integer and the other is a floating-point number.

Overall, the project demonstrates a meticulous fusion of theoretical concepts and their practical application in designing an advanced and adaptable multiplier. This multiplier adeptly meets the demands of modern computational challenges, showcasing its adaptability and effectiveness.

Future Scope: This work can be extended by exploring the integration of additional arithmetic operations beyond multiplication, such as division, square root, and exponentiation, into the existing multiplier architecture. This expansion would create a more comprehensive arithmetic processing unit capable of handling a wider range of computational tasks within digital signal processing systems. Furthermore, enhancing the versatility and adaptability of the multiplier unit by incorporating configurable parameters and modes of operation could significantly broaden its applicability across diverse application domains. For instance, implementing configurable precision settings for floating-point operations or supporting various integer arithmetic formats would enable the multiplier to cater to a broader spectrum of computational requirements.

APPENDIX:

 # PERES Gate:

```
module PeresGate (input a,input b,input c,output o1,output o2,output o3);
assign o1 = a;
assign o2 = a^b;
assign o3 = ((a&b)^c);
endmodule
```

Test Bench code:

```
`timescale 1ns / 1ps
module PeresGate_tb;
reg a;
reg b;
reg c;
wire o1;
wire o2;
wire o3;
PeresGate uut (.a(a),.b(b),.c(c),.o1(o1),.o2(o2),.o3(o3));
initial begin
 a = 0; b = 0; c = 0;#100;
 a = 0; b = 0; c = 1;#100;
 a = 0; b = 1; c = 0;#100;
 a = 0; b = 1; c = 1;#100;
 a = 1; b = 0; c = 0;#100;
 a = 1; b = 0; c = 1;#100;
 a = 1; b = 1; c = 0;#100;
 a = 1; b = 1; c = 1;#100;
end
endmodule
```

FEYNMAN GATE

```
module Feynmangate (input a,input b,output o1,output o2);
assign o1 = a;
assign o2 = a^b;
endmodule
```

 # FULL ADDER

```
`include "ha.v"
 module fa (input a,input b,input cin, output sum,output carry);
 wire h1_sum, h1_carry, h2_carry;
```

```
 ha half_adder1 (.a(a),.b(b), .sum(h1_sum),.carry(h1_carry));
 ha half_adder2 (.a(h1_sum),.b(cin), .sum(sum),.carry(h2_carry));
 Ror carry_or_gate ( .a(h1_carry), .b(h2_carry), .o1(temp_carry), .o2(carry));
endmodule
```

 # 32 bit Multiplier

```
  module pm32(a,b,c);
   input [31:0]a,b;
   output [63:0]c;
   wire [31:0]q0,q1,q2,q3,q4,temp1;
   wire [63:0]c;
   wire [47:0]q5,q6,temp2,temp3,temp4;
   pm16 z1(a[15:0],b[15:0],q0[31:0]);
   pm16 z2(a[31:16],b[31:16],q1[31:0]);
   pm16 z3(a[15:0],b[31:16],q2[31:0]);
   pm16 z4(a[31:16],b[31:16],q3[31:0]);
   assign temp1 ={16'b0,q0[31:16]};
   assign q4 = q1[31:0]+temp1;
   assign temp2 ={16'b0,q2[31:0]};
   assign temp3 ={q3[31:0],16'b0};
   assign q5 = temp2+temp3;
   assign temp4={16'b0,q4[31:0]};
   assign q6 = temp4 + q5;
   assign c[15:0]=q0[15:0];
   assign c[63:16]=q6[47:0];
  endmodule
```

#FLOATING POINT MULTIPLIER

Underflow /overflow checking and Control module for operating other modules:

```
module control(input wire [31:0] inp1,input wire [31:0] inp2,
output wire [31:0] out,output wire underflow,
output wire overflow);
wire sign;wire [7:0] exp1; wire [7:0] exp2;
wire [7:0] exp_out; wire [7:0] test_exp;
wire [22:0] mant1; wire [22:0] mant2;
wire [22:0] mant_out;
sign_bit sign_bit1(sign, inp1, inp2);
wire [7:0] temp1;
wire dummy; //to connect unused cout ports of adder wire carry
```

```
wire [8:0] sub_temp;
ripple_8 rip1(temp1, carry, inp1 [30:23], inp2 [30:23],1'b0);
subtractor_9bit sub1 (sub_temp, underflow, {carry, temp1}, 1'b0);
and (overflow, sub_temp [8], 1'b1); //if the exponent has more than 8 bits: overflow
wire [47:0] prdt;
product p1 (prdt, {1'b1, inp1 [22:0]},{1'b1, inp2[22:0]});
wire norm_flag;
wire [22:0] adj_mantissa;
normalize norm1(adj_mantissa, norm_flag,prdt);
ripple_8 ripple_norm(test_exp, dummy,sub_temp [7:0], {7'b0, norm_flag}, 1'b0);
assign out [31] = sign;
assign out [30:23] = test_exp;
assign out [22:0] = adj_mantissa;
endmodule
```

combined multiplier

```
module cmulti(
   input wire [31:0] int_input1, // Integer input 1
   input wire [31:0] int_input2, // Integer input 2
   input wire [31:0] fp_input1,  // Floating-point input 1
   input wire [31:0] fp_input2,  // Floating-point input 2
   output wire [63:0] int_output, // Integer output
   output wire [31:0] fp_output,  // Floating-point output
   output wire int_mode          // Integer mode select
);
   wire [63:0] int_out;
   wire [31:0] fp_out;
   pm32 int_mult(
      .a(int_input1),
      .b(int_input2),
      .c(int_out) );
  control fp_mult(
      .inp1(fp_input1),
      .inp2(fp_input2),
      .out(fp_out),
      .underflow(),
      .overflow() );
   assign int_output = int_out;
   assign fp_output = fp_out;
   assign int_mode = (int_input1 !== 32'b0 && int_input2 !== 32'b0) ? 1'b1 : 1'b0;
endmodule
```

References

[1] Syed Farah Naz and Ambika Prasad Shah ," Reversible Gates: A Paradigm Shift in Computing", in IEEE Open Journal Of Circuits And Systems , 15 August 2023 ,vol. 4, pp.241 257.

[2] Anekant Jain and Rakhee Jain ," Design of Reversible Single Precision and Double Precision Floating Point Multipliers", in IEEE open journal Of Circuits , RGPV, Bhopal ,India 2018 .

[3] Mr. Ashish K. Thakre , Prof. Sujata S. Chiwande and Mr. Sumit D. Chafale ," Design of Low Power Multiplier Using Reversible Logic Gate", in IEEE Journal , Nagpur, India 2014.

[4] Anamika and Rockey Bhardwaj, " Reversible Logic Gates and its Performances ",in IEEE journal , Chandigarh University Gharuan, India 2018./

[5] Himanshu Thapliyal, Hamid R. Arabnia and A.P Vinod," Combined Integer and Floating Point Multiplication Architecture (CIFM) for FPGAs and Its Reversible Logic Implementation", in journal, IIIT Hyderabad, India 2006.

[6] Young-Min Jun And In-Chan Choi , " Optimal Multi-Bit Toffoli Gate Synthesis" , in *IEEE Access* , 9 February 2023, vol. 11, pp. 27342-27351.

[7] Song-Hyeon Kuk , Seungmin Han, Dong Hyun Lee, Bong Ho Kim , Joonsup Shim , Min Hyuk Park , Jae-Hoon Han and Sang-Hyeon Kim ," Logic And Memory Ferroelectric Field-Effect-Transistor Using Reversible And Irreversible Domain Wall Polarization" , in *IEEE Electron Device Letters*, vol. 44, No. 1, January 2023 ,pp.36-39.

[8] Hai-Sheng Li , "The Optimization and Application of 3-Bit Hermitian Gates and Multiple Control Toffoli Gates" , in *IEEE Transactions On Quantum Engineering* , 29 September 2022,vol. 3, pp. 3102715- 3102715.

[9] Hieu Nguyen And Linh H. Tran ," Synthesis of Reversible and Quantum Circuit Using ROCBDD and Mixed-Polarity Toffoli Gate" ,in *IEEE Access* , September 29, 2021, vol. 9 , pp. 135432- 135439.

[10] Sithara Raveendran , Pranose J. Edavoor , Y. B. Nithin Kumar And M. H. Vasantha ," Inexact Signed Wallace Tree Multiplier Design Using Reversible Logic" , in *IEEE Access* , July 28, 2021, vol. 9, pp. 108119- 108130.

[11] Timothée Goubault De Brugière ,Marc Baboulin, Benoît Valiron, Simon Martiel, and Cyril Allouche ," Reducing the Depth of Linear Reversible Quantum Circuits" , in *IEEE Transactions On Quantum Engineering* ,July 7, 2021 ,vol. 2, pp. 3102422- 3102444.

[12] Chetan Kumar Dabhi , Ananda S. Roy , Lucy Yang, and Yogesh Singh Chauhan ," Anomalous Gidl Effect with Back Bias in FINFET: Physical Insights and Compact Modeling", in *IEEE Transactions On Electron Devices,* vol. 68, No. 7 , pp. 3261-3267.

[13] Ahmed Moustafa and Ahmed Younes, "Efficient Synthesis of Reversible Circuits Using Quantum Dot Cellular Automata" ,in *IEEE Access* ,May 25, 2021, vol. 9, pp. 76662-76673.

[14] Kevin D. Osborn and Waltraut Wustmann ," Reversible Fluxon Logic with Optimized CNOT Gate Components" ,in *IEEE Transactions On Applied Superconductivity,* vol. 31, No. 2, March 2021, pp. 1300213- 1300225.

[15] Seong-Min Cho , Aeyoung Kim , Dooho Choi , Byung-Soo Choi and Seung-Hyun Seo, " Quantum Modular Multiplication" , in *IEEE Access* , November 18, 2020, vol.8 ,pp. 213244-213252.

[16] Sithara Raveendran , Pranose J. Edavoor , Nithin Y. B. Kumar and M. H. Vasantha ," An Approximate Low-Power Lifting Scheme Using Reversible Logic ", in *IEEE Access* , October 6, 2020, vol.8,pp. 183367- 183377.

[17] Jiaqing Jiang, Xiaoming Sun, Yuan Sun, Kewen Wu and Zhiyu Xia ," Structured Decomposition For Reversible Boolean Functions" , in *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, ,* October 2020 ,vol. 39, No. 10, pp.2410-2412.

[18] Bikash Debnath , Jadav Chandra Das , Debashis De , Ferial Ghaemi , Ali Ahmadian and Norazak Senu ," Reversible Palm Vein Authenticator Design with Quantum Dot Cellular Automata for Information Security in Nanocommunication Network", in *IEEE Access* , September 22, 2020 ,vol. 8,pp. 174821-174832.

[19] Jeferson F. Chaves , Marco A. Ribeiro , Frank Sill Torres , and Omar P. Vilela Neto , "Designing Partially Reversible Field-Coupled Nanocomputing Circuits ", in *IEEE Transactions On Nanotechnology,* vol. 18,2019, pp.589-597.

[20] Sajjad Parvin and Mustafa Altun ," Perfect Concurrent Fault Detection in CMOS Logic Circuits Using Parity Preservative Reversible Gates ", in *IEEE Access* , November 4, 2019, vol. 7, pp. 163939- 163947.

[21] Hari Mohan Gaur, Ashutosh Kumar Singh, And Umesh Ghanekar ," Design of Reversible Arithmetic Logic Unit With Built-In Testability " , in *IEEE Design & Test*, September/October 2019.

[22] A. N. Nagamani, S. N. Anuktha, N. Nanditha and Vinod Kumar Agrawal, "A Genetic Algorithm-Based Heuristic Method for Test Set Generation in Reversible Circuits" in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, No. 2, February 2018 , pp.324-336.

[23] Arman Roohi , Ramtin Zand, Shaahin Angizi , and Ronald F. Demara, "A Parity-Preserving Reversible QCA Gate with Self-Checking Cascadable Resiliency" , in *IEEE Transaction On Emerging Topics in Computing*, 21 July 2016,vol.6, No. 4,pp.450-459.

[24] Kamalika Datta, Indranil Sengupta, and Hafizur Rahaman , "A Post-Synthesis Optimization Technique for Reversible Circuits Exploiting Negative Control Lines" , in *IEEE Transactions on Computers*, vol. 64, No. 4, April 2015,pp. 1208-1214.

[25] Mozammel H A Khan, " Design Of Reversible Synchronous Sequential Circuits Using Pseudo Reed-Muller Expressions" , *in IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, vol. 22, No. 11, November 2014 ,pp. 2278-2286.

[26] Shu Nakaharai, Tomohiko Iijima, Shinichi Ogawa, Song-Lin Li, Kazuhito Tsukagoshi, Shintaro Sato, And Naoki Yokoyama ," Electrostatically Reversible Polarity of Dual-Gated Graphene Transistors" , in *IEEE Transactions on Nanotechnology*, vol. 13, No. 6, November 2014 , pp.1039-1043.

[27]Purnima Sethi And Sukhdev Roy, "All-Optical Ultrafast Switching in 2 × 2 Silicon Microring Resonators and its Application to Reconfigurable Demux/Mux and Reversible Logic Gates" , in *Journal of Lightwave Technology*, vol. 32, No. 12, June 15, 2014 , pp. 2173- 2180.

[28] Himanshu Thapliyal , Nagarajan Ranganathan, and Saurabh Kotiyal," Design of Testable Reversible Sequential Circuits" , in IEEE *Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, No. 7, July 2013,pp.1201-1209.

[29] Oleg Golubitsky and Dmitri Maslov, "A Study of Optimal 4-Bit Reversible Toffoli Circuits and their Synthesis" ,in *IEEE Transactions on Computers*, vol. 61, No. 9, September 2012 ,pp. 1341-1353.

[30] Tanay Chattopadhyay ," All-Optical Modified Fredkin Gate" , in *IEEE Journal Of Selected Topics in Quantum Electronics*, vol. 18, No. 2, March/April 2012 ,pp. 585-592.

[31] Sk. Noor Mahammad and Kamakoti Veezhinathan ," Constructing Online Testable Circuits Using Reversible Logic ", in *IEEE Transactions on Instrumentation and Measurement*, vol. 59, No. 1, January 2010 , pp .101-109.

[32] Himanshu Thapliyal And Nagarajan Ranganathan, "Reversible Logic-Based Concurrently Testable Latches for Molecular QCA" , in *IEEE Transactions on Nanotechnology*, vol. 9, No. 1, January 2010 ,pp.62-69.

[33] Pallav Gupta, Abhinav Agrawal, and Niraj K. Jha," An Algorithm for Synthesis of Reversible Logic Circuits" , in *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, No. 11, November 2006 ,pp. 2317-2330.

[34] Dmitri Maslov, Gerhard W. Dueck, and D. Michael Miller," Toffoli Network Synthesis with Templates" , in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, No. 6, June 2005 ,pp. 807-817.