JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# 32-BIT FPGA BASED ALU EMPLOYING REVERSIBLE LOGIC

**K. Rushendrababu[1], N. Lakshmi Pavani[2], P. Yaswitha[3], M. Manoj Kumar[4], Shaheda Begum[5]**

[1]K. Rushendra babu, Assistant Professor, Gudlavalleru Engineering College, Andhra Pradesh

***Abstract*** **- The Arithmetic Logic Unit (ALU) stands as a pivotal subsystem within processors, playing a crucial role in executing arithmetic and logical functions essential for digital system operations across a multitude of devices such as calculators, cell phones, and computers. However, conventional ALUs constructed using non-reversible logic gates are notorious for their substantial power consumption, presenting a pressing need for more energy-efficient alternatives in digital system design. In response to this challenge, our project proposes the development of a 32-bit ALU utilizing reversible logic gates, with the dual objective of reducing power consumption and enhancing computational performance. By embracing the principles of reversible logic gates, our proposed 32-bit ALU seeks to revolutionize digital system design by offering a solution that not only minimizes power consumption but also enhances computational efficiency. In addition to mitigating energy concerns, we aim to expand the functionality of the ALU by incorporating a comprehensive set of 16 distinct operations, further elevating its utility and versatility in various computing applications. Through this innovative approach, we strive to establish a new paradigm in ALU design, one that prioritizes both power efficiency and computational prowess. The integration of reversible logic gates into the architecture of our 32-bit ALU represents a significant step towards realizing the vision of low-power digital systems without compromising computational capabilities. With meticulous attention to detail and a focus on innovation, our project aims to contribute to the advancement of digital system design, addressing the critical need for energy-efficient solutions in today's technology-driven world.**

***Key Words:*** Arithmetic Logic Unit (ALU), Reversible Logic Gates, Low Power Consumption, Computational Performance, Digital System Design, Energy Efficiency, Versatility, Innovative Architecture.

## 1. INTRODUCTION

The ALU, a core CPU component, handles arithmetic and logical operations on binary numbers. Its functions include addition, subtraction, multiplication, division, AND, OR, XOR, and NOT. Built with combinational logic circuits, it processes inputs from the CPU's control unit and registers using logic gates. The bit width determines the size of binary numbers it can process at once. Advances aim to enhance speed, efficiency, and parallelism, with specialized operations for specific tasks. New methods like reversible logic gates aim to cut power consumption while maintaining performance, crucial for modern computing efficiency.

### 1.1 Arithmetic Logic Unit

The Arithmetic Logic Unit (ALU) is a central component of a CPU, responsible for executing arithmetic and logical operations on binary numbers. It encompasses functions such as addition, subtraction, multiplication, division, and logical operations like AND, OR, XOR, and NOT. Composed of combinational logic circuits, the ALU processes inputs from the CPU's control unit and registers through logic gates. Its bit width determines the size of binary numbers it can handle in one operation. Advances in ALU design aim to improve speed, efficiency, and parallelism, with specialized operations tailored for specific tasks. Emerging techniques, like reversible logic gates, seek to reduce power consumption while maintaining computational performance. Ultimately, the ALU significantly influences the performance and efficiency of modern computing systems.

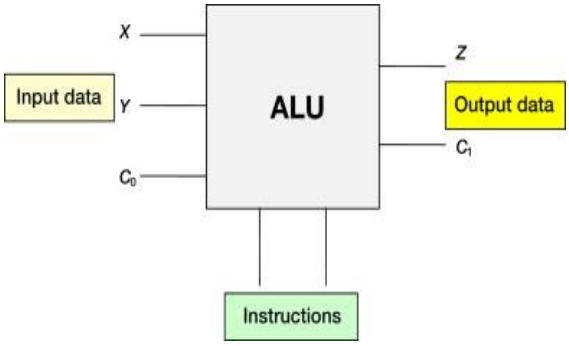


**Fig.1.1: Basic Block Diagram of ALU**

# 2. EXISTING METHOD

To design a 32-bit irreversible ALU with 16 operations, one would start by defining the supported operations, which could encompass arithmetic, logical, and comparison operations. The ALU would have two 32-bit inputs and one 32-bit output. Components for arithmetic operations (addition, subtraction, multiplication, division), logical operations (AND, OR, XOR, NOT), shifts (left, right), and comparisons (greater than, less than) would be implemented. Control logic would be devised to select operations based on input signals. Ensuring irreversibility involves employing irreversible algorithms or logic gates. Extensive testing and documentation are crucial for verifying the correctness and functionality of the ALU.

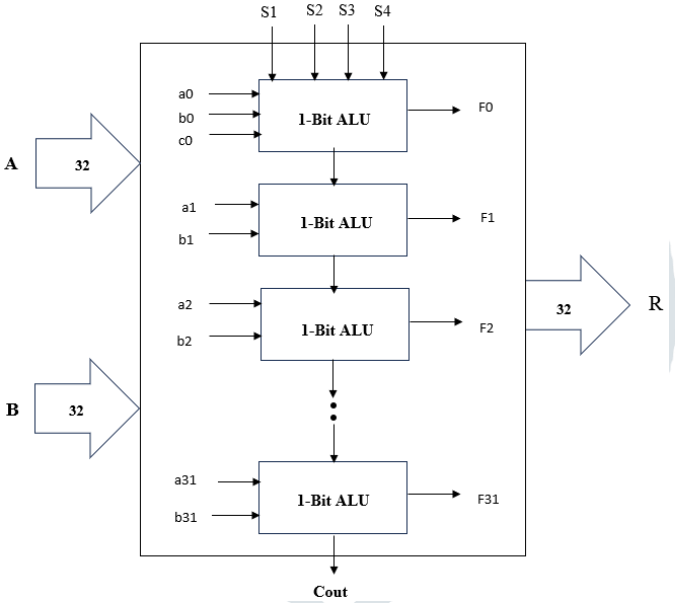


**Fig. 2.1: Implementation diagram of 32-bit ALU**

The diagram represents a 32-bit Arithmetic Logic Unit (ALU), a pivotal component of a CPU responsible for arithmetic and logical operations on binary data. Composed of individual 1-bit ALUs, it processes data in 32-bit chunks. Control signals dictate specific operations, with inputs including operands, control signals, and carry-in, and outputs comprising results, carry-out, and status flags. Through bitwise operations, it manipulates individual bits, handling tasks like logical AND and arithmetic operations with carry propagation. This design underscores how complex operations are decomposed into smaller units for efficient data processing within a computing system.

# 3.PROPOSED METHOD

## 3.1 Reversible logic gates

Reversible logic, characterized by the property that each input vector produces a unique output vector, is gaining significance in various future computing technologies. Its implementation is indispensable for realizing quantum computing. The primary objectives behind designing reversible logic circuits include reducing quantum cost, circuit depth, and the number of garbage outputs. Reversible circuits serve as the fundamental building blocks of quantum computers, given that all quantum operations are reversible. However, reversible logic circuits are subject to two key constraints:

1. Fan-out limitation: Unlike conventional logic circuits, reversible logic circuits cannot have fan-out, meaning that a signal cannot be duplicated.
2. No feedback or loops: Reversible logic circuits cannot incorporate feedback or loops in their design.
3. When designing reversible logic circuits, it's essential to ensure the following characteristics:
4. Minimal usage of reversible gates: Utilize the smallest possible number of reversible gates to optimize circuit efficiency.
5. Minimal generation of garbage outputs: Limit the production of garbage outputs, which are outputs that do not contribute to the desired computation and may degrade circuit performance.
6. Minimal reliance on constant inputs: Reduce the reliance on constant inputs to streamline circuit operation and improve resource utilization.

### 3.1.1 NOT Gate

The simplest reversible gate is the NOT gate, which operates on a single input. In a 1x1 gate setup, the NOT gate has zero quantum cost. Its function is basic: if the input is A, then the output P is the opposite of A, meaning if A is 0, then P is 1, and if A is 1, then P is 0. This gate's simplicity and lack of quantum cost make it fundamental in various circuit designs.

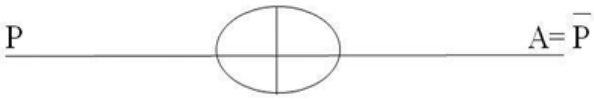


**Fig 3.1.1: Not gate**

### 3.1.2 Feynman Gate

The Feynman gate, also recognized as the Controlled NOT gate, is a 2x2 reversible gate extensively utilized for fan-out purpose in computing. With inputs labeled as A and B, it produces outputs P = A and Q = A ⊕ B (where ⊕ denotes XOR). Its quantum cost is one making it the singular 2x2 reversible gate accessible, hence frequently selected for fan-out operations across diverse circuit designs.

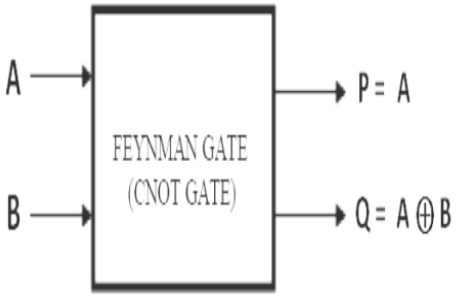


**Fig 3.1.2: Feynman Gate**

### 3.1.3 Toffoli Gate

The Toffoli gate, also known as the CCNOT gate, is a 3x3 gate with inputs labeled as A, B, and C, producing outputs P = A, Q = B, and R = A.B ⊕ C (where **'.'** represents logical AND & **"⊕"** denotes XOR). It holds a quantum cost of 5 and produces 2 garbage outputs. Invented by Tommaso Toffoli, it is recognized as a universal reversible logic gate, meaning that any reversible circuit can be constructed solely using Toffoli gates.

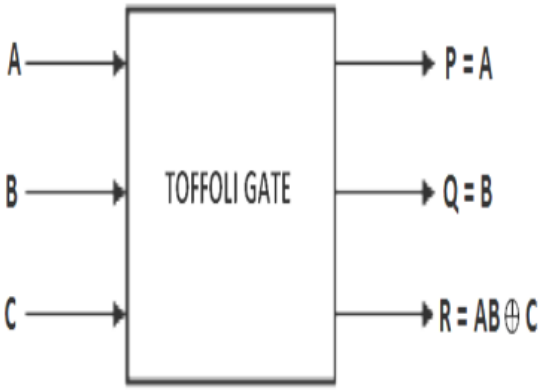


**Fig 3.1.3: Toffoli gates**

### 3.1.4 Peres Gate

The Peres gate, also referred to as the 3x3 reversible gate, operates on inputs A, B, and C, yielding outputs P = A, Q = A ⊕ B, and R = A, B ⊕ C (where ⊕ signifies XOR and, represents logical AND). With a quantum cost of 4, it stands out for its efficiency, often making it a preferred choice in various circuit designs. Notably, a single Peres gate is sufficient to design a half adder due to its minimal quantum cost.

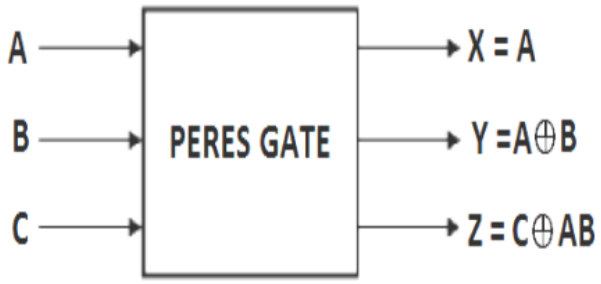


**Fig 3.1.4: Peres Gate**

### 3.1.5 Fredkin Gate

The Fredkin gate, also known as the controlled swap gate or CSWAP gate, is a reversible three-bit gate in quantum computing and reversible computing. The Fredkin gate performs a controlled swap operation on its three input bits, depending on the state of the control bit. If the control bit is set to 1, it swaps the second and third bits. If the control bit is 0, it leaves the bits unchanged.

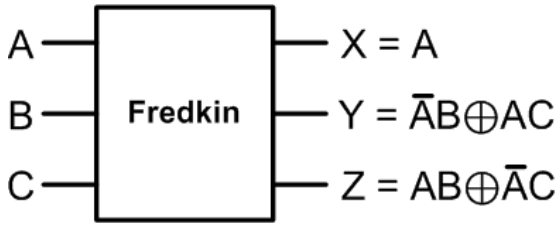


**Fig 3.1.5: Fredkin Gate**

### 3.2 32-bit Reversible ALU

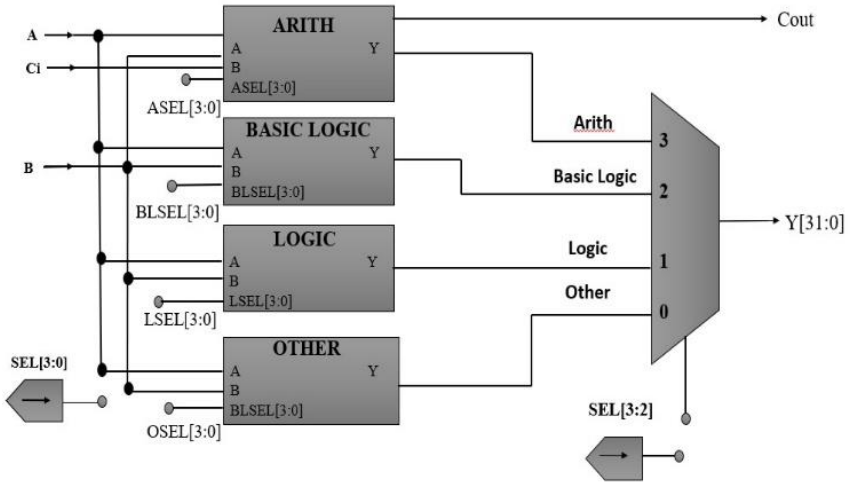


**Fig. 3.2.1: Implementation diagram of 1-bit Reversible ALU**

1-bit ALU embodies flexibility and efficiency in digital systems, handling arithmetic, basic logic, advanced logic, and specialized operations within a single hardware unit. ARITH block executes arithmetic operations based on ASEL [1:0] control signal, adapting to diverse computational needs. BASIC LOGIC block manages fundamental logic functions like AND, OR, XOR, etc., selected via BLSEL [1:0]. LOGIC block tackles complex logical tasks like multiplexing and shifting, guided by LSEL [1:0]. OTHER block caters to specialized functions such as signal conditioning and data conversion, controlled by OSEL [1:0]. Dynamic SEL [1:0] signals enable seamless mode transitions, optimizing resource usage and minimizing hardware overhead. This modular design reduces complexity, power consumption, and enhances scalability, ideal for various applications from embedded systems to high-performance computing.

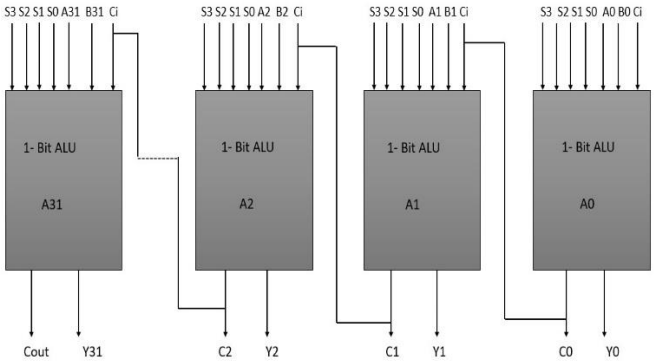


**Fig 3.2.2: Implementation diagram of 32-bit Reversible ALU**

The 32-bit ALU specializes in arithmetic and logical operations on binary data, consisting of 32 individual 1-bit ALUs. Each 1-bit ALU processes a pair of input bits and performs operations determined by function signals. Carry-out from each ALU propagates to the next, enabling multi-bit arithmetic and logical operations. Results from all ALUs are consolidated into the 32-bit output bus. In contrast, the multi-function digital logic unit offers flexibility in handling diverse operations, while the 32-bit ALU focuses on fixed-width computations. Both units serve as fundamental components in digital system design.

**Table 1.1 Truth Table of 32-bit ALU**

| Sno | Selection line | Operation |
|---|---|---|
| 0 | 0000 | A+B |
| 1 | 0001 | A-B |
| 2 | 0010 | A*B |
| 3 | 0011 | A \| B |
| 4 | 0100 | ~A |
| 5 | 0101 | R0 |
| 6 | 0110 | ~(A^B) |
| 7 | 0111 | ~ (A & B) |
| 8 | 1000 | Cl |
| 9 | 1001 | ~ (A \| B) |
| 10 | 1010 | A & B |
| 11 | 1011 | IO STS |
| 12 | 1100 | INC |
| 13 | 1101 | B*1 |
| 14 | 1110 | A^B |
| 15 | 1111 | A*1 |

The ALU has two 32-bit input lines labelled “A” and “B”. Each bit from the A and B inputs (a0 to a31 and b0 to b31) is connected to individual 1-bit ALUs. There are four control signals. These likely determine the operation performed by the ALU.The ALU features two 32-bit inputs and employs four selection lines for executing 16 operations using reversible logic gates. However, the table outlining these operations.

The provided table outlines a set of control signals and their corresponding operations within a digital system, likely part of an Arithmetic Logic Unit (ALU) or a similar computational unit. Here's a brief explanation of each operation:

**0000 (sum):** Adds two operands, a and b, to produce their sum.
**0001 (a - b):** Subtracts operand b from operand a to yield the result.
**0010 (a *b):** Bit by Bit Multiplication which yields the product as a result.
**0011 (a | b):** Performs a bitwise OR operation between operands a and b.
**0100 (~a):** Computes the bitwise complement of operand a.
**0101 (R0):** Returns the value stored in register R0.
**0110 (~ (a ^ b)):** Performs a bitwise XNOR operation between operands a and b.
**0111 (~(a&b)):** Performs a bitwise NAND operation between operands a and b.
**1000 (Clear):** Returns a 32-bit zero value.
**1001 (~ (a | b)):** Performs a bitwise NOR operation between operands a and b.
**1010 (a&b):** Performs a bitwise AND operation between operands a and b.
**1011 (IO_STS):** Returns the value of the IO_STS register.
**1100 (INC):** Increments the value of a by 1.
**1101 (b*1):** Multiplies the contents of b by one time.
**1110 (a ^ b):** Computes the bitwise XOR of operands a and b.
**1111 (a*1):** Multiplies the contents of b by one time.

# 4. DESIGN METHODOLOGY

The methodology for the project involves a systematic approach to design, implement, and evaluate a 32-Bit ALU utilizing reversible logic gates. Initially, a thorough literature review is conducted to understand the current state of the art in reversible logic gates, ALU design, FPGA implementation, and related topics. Subsequently, specific design specifications and requirements are defined, encompassing supported arithmetic and logical operations, input-output formats, and performance targets such as speed, power consumption, and delay. With these specifications in place, the logic design of the 32-Bit ALU is developed, focusing on designing individual functional blocks like adders, multipliers, and logical gates while ensuring compatibility with reversible logic principles. Following this, the designed ALU is implemented on an FPGA platform using hardware description languages (HDL) and synthesis tools. Rigorous verification and testing procedures are then carried out, involving both simulation and hardware testing, to validate functionality and assess performance metrics. Comparative performance evaluation is conducted, comparing the

reversible 32-Bit ALU with traditional non-reversible designs in terms of power consumption, dissipation, speed, and delay. Based on the evaluation results, optimization techniques are explored to enhance the efficiency and effectiveness of the reversible ALU design. Throughout the process, comprehensive documentation is maintained, peaking in a detailed report outlining the methodology, findings, and conclusions of the project, aimed at distribution to relevant stakeholders and contributing to advancements in low-power computing and digital system design.

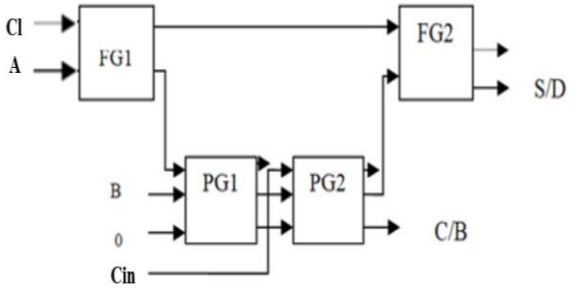


**Fig 4.1: Full adder and Full subtractor**

In Figure 4.1, a full adder/subtractor is designed using Feynman gates and Perse gates. The selection between addition and subtraction operations is determined by the control line. If the control signal is set to 0, the circuit performs addition. Conversely, if the control signal is set to 1, the circuit performs subtraction.

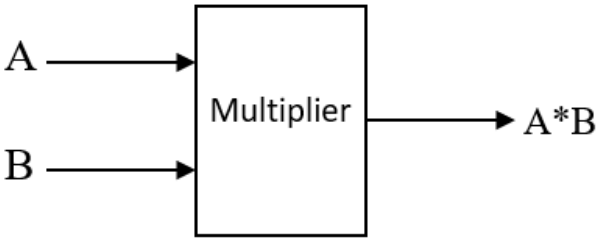


**Fig 4.2: Multiplier**

The multiplier block depicted in Figure 4.2 accepts two inputs, labeled as A and B, and executes the multiplication operation based on these inputs. A bit-by-bit multiplication takes place in this block.

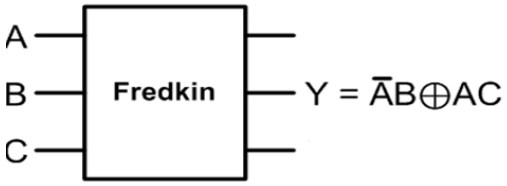


**Fig 4.3: OR Operation**

Figure 4.3 illustrates an OR operation block, where a Fredkin gate is employed to execute the logical OR operation. The Fredkin gate, configured with two inputs labeled A and B, performs the OR operation on these inputs.

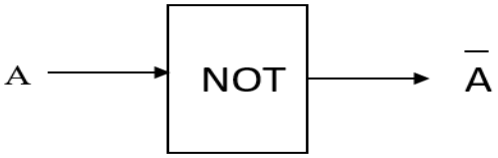


**Fig 4.4: NOT Operation**

Figure 4.4 shows Not operation. It takes single input A. It performs not operation on input A.

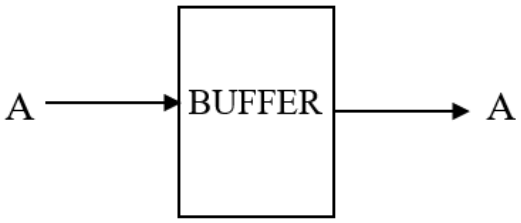


**Fig 4.5: Buffer**

Figure 4.5 shows Buffer operation. It takes single input A. It performs read operation on input A.

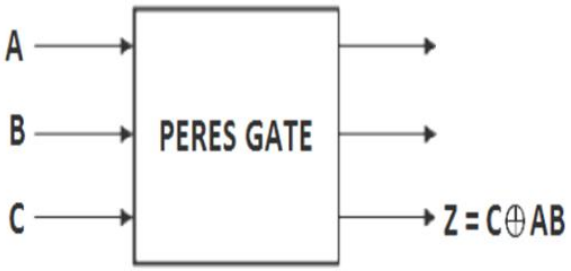


**Fig 4.6: AND Operation**

Figure 4.6 illustrates an AND operation block, where a Peres gate is employed to execute the logical AND operation by passing input c as zero. The Peres gate, configured with three inputs labeled A, B and C, performs them AND operation on these inputs.

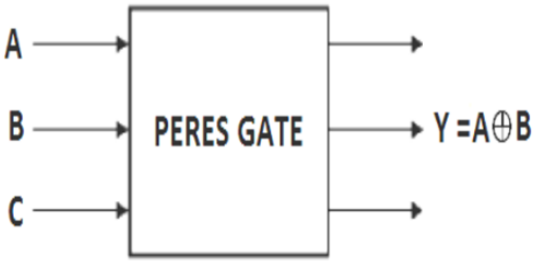


**Fig 4.7: XOR Operation**

Figure 4.7 illustrates an XOR operation block, where a Peres gate is employed to execute the logical XOR operation. The Peres gate, configured with three inputs labeled A, B and C, performs them XOR operation on these inputs.

# 5.SCHEMATIC VIEWS

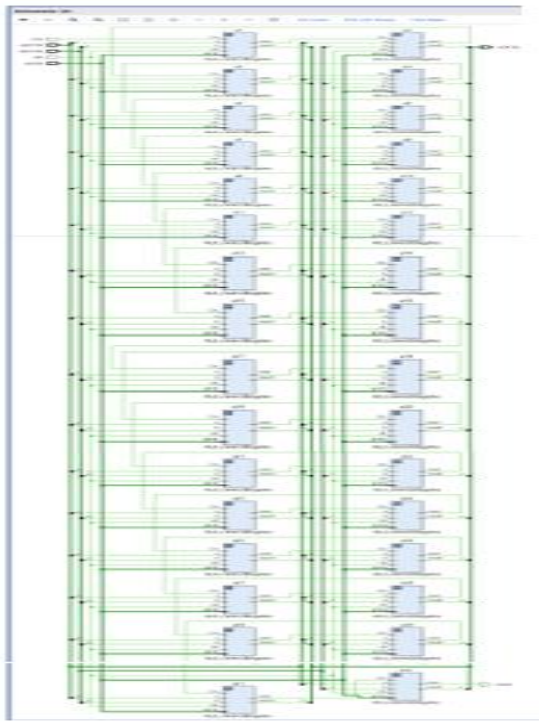

**Fig 5.1: RTL schematic of 32-bit reversible ALU**

Figure 5.1 shows RTL Schematic of 32-bit reversible ALU. A 32-bit reversible ALU involves employing reversible decoder-controlled combinational circuits, presenting significant advantages. Notably, it slashes dynamic power consumption by 1.6 times, utilizes merely 3% of FPGA memory, and conserves a substantial 91% of space in contrast to conventional designs. To summarize, the integration of reversible logic, efficient decoding, and optimized power utilization enhances computational efficiency in the ALU's RTL schematic.

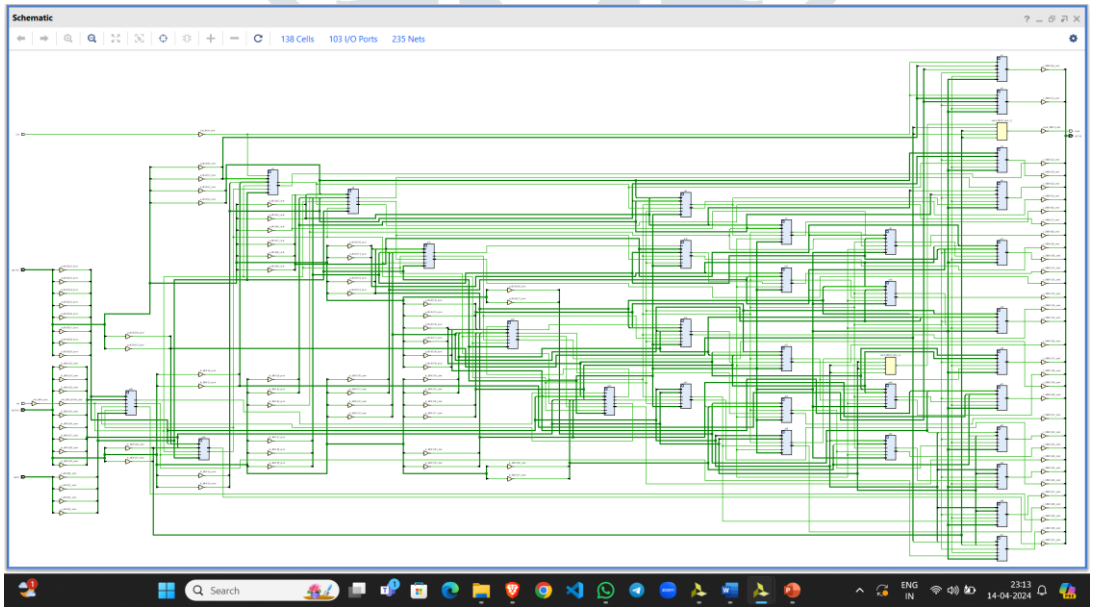


**Fig 5.2: Technology schematic of 32-bit Reversible ALU**

Figure 4.2 shows Technology schematic of 32-bit reversible ALU. The technology schematic of a 32-bit Reversible ALU illustrates how its components are laid out and interconnected. It integrates reversible logic principles and decoder-controlled circuits to perform operations efficiently. Gates like Toffoli and Fredkin enable invertible operations, while decoders select operations based on control signals. Efficiency is a priority, with gate-level optimization and layout design

minimizing power, area, and latency. Techniques to optimize memory and area usage reduce FPGA resource consumption, enhancing compactness and efficiency. Thorough simulation and verification processes ensure performance enhancement, focusing on computational speed, reliability, and scalability.

# 6.SIMULATION RESULTS


**Fig 6.1: Output waveform of first 8 operations.**

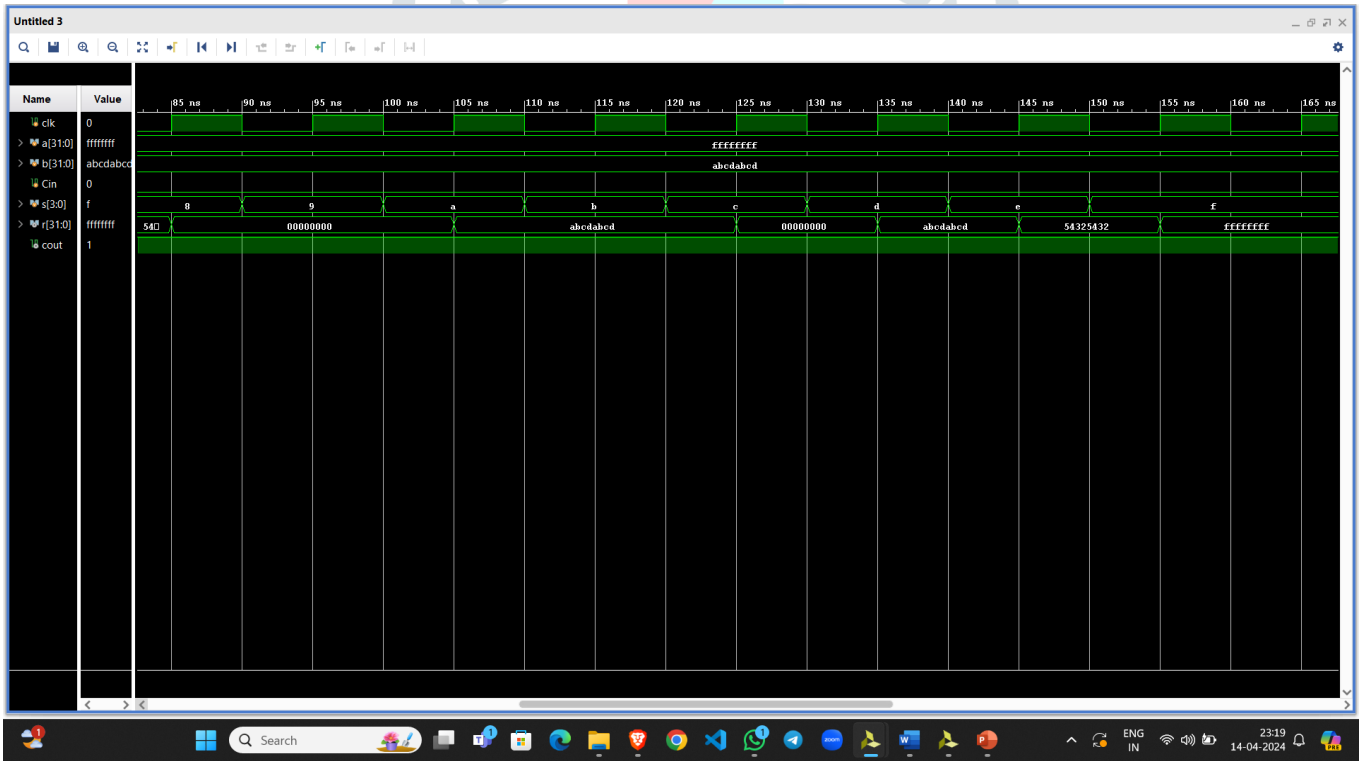


**Fig 6.2: Output waveform of next 8 operations.**

The simulation waveform represents an overview of control signals used in digital systems, each corresponding to specific operations. Binary codes are employed to represent these signals, helping configure component behavior such as the ALU. Understanding these signals is essential for designing efficient digital systems, enabling engineers to create custom instructions and manage data flow effectively. In essence, the table serves as a fundamental guide for implementing and interpreting control signals in digital system design.

# 6.CONCLUSION

In this paper we had implemented a 32-bit Reversible ALU using Peres, Fredkin, Feynman and Toffoli Gates. The main moto of this paper is to improve the performance of ALU. The latest design outperforms previous ones in power efficiency and conducts a comparative analysis of reversible and irreversible gates' delay and power dissipation. Remarkably, it utilizes just 3% of the FPGA's total area, demonstrating significant area optimization. Notably, irreversible gate logic shows a substantial decrease in delay from 10.636ns to 5.255ns and power dissipation from 22.488W to 1.136W.

**1.2 Comparison Table of Reversible and Irreversible ALU Parameters**

| Parameters | 32-bit ALU Irreversible gates | 32-bit ALU Reversible gates |
|---|---|---|
| Power | 22.488W | 1.136W |
| Delay | 10.636ns | 5.255ns |
| Area(LUT'S) | 243 | 7 |

As this design is based on basic reversible gates, furthermore this approach will extend to other circuit designs like encoders, and also the gates like CNOT, DNG, HNG can be further used in order to reduce the power, delay and Area further showcases the adaptability and versatility of reversible logic. It highlights how fundamental principles can be creatively applied to various levels of circuit complexity, paving the way for innovative solutions in logic design.

# 7.REFERENCES

[1] Tiwari, M. K., & Kumar, D. A Review on Reversible Logic Gates and Their Applications in Digital System Design. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(5),632-636. (2018).

[2] Saeedi, S., & Zamani, M. A Survey of Reversible Arithmetic Logic Units. ACM Transactions on-Automation of Electronic Systems (TODAES), 26(2), 1-29. (2020).

[3] Li, J., & Khalid, F. FPGA Implementation of a Reversible 8-bit ALU. In Proceedings of the International Conference on Field-Programmable Technology (FPT) (pp. 234-239). (2019).

[4] Gupta, R., & Jain, S. FPGA Implementation of Reversible 4-bit ALU Using DKG Gate. International Journal of Computer Applications, 134(13), 1-5. (2016).

[5] Khosrozadeh, A., et al. A Comprehensive Review on Designing 16-Bit Arithmetic Logic Unit Using Reversible Logic Gates. International Journal of Computer Applications, 169(6), 36-42. (2017).

[6] S.M. Swamynathan[1], Dept. of ECE, SNS College of Technology, Coimbatore. India. V. Banumathi [2], Dept. of ECE, Anna University Regional Centre, Coimbatore. India. Design and Analysis of FPGA Based 32 Bit ALU Using Reversible Gates.

[7] Soumya Sen, Piyali Saha, Souvik Saha," FPGA-Supported HDL Approach to Implement Reversible Logic Gate-Based ALU", International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON) 1-4. (2023)

[8] Noel R. Strader, Phillip E. Allen, and Randall L. Geiger." VLSI Design Techniques for Analog and Digital Circuits":612.

[9] Chandni N. Naik, Vaishnavi M. Velvani, Pooja J. Patel, Khushbu G. Parekh, "VLSI Based 16 Bit ALU with Interfacing Circuit", International Journal of Innovative and Emerging Research in Engineering Volume 2, Issue 3, 2015.

[10] Arvind Rajput, Anil Goyal, "Design and comparison of low power & high speed 4-bit ALU", Proc. of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008), RIMT-IET, Mandi Gobindgarh. March 29, 2008.

[11] Dhanabal R,Bharathi, V,Saira Salim, "design of 16-bit low power ALU- dbgpu", Dhanabal Ret.al / International Journal of Engineering and Technology (IJET), ISSN: 0975-4024, Vol.5 No.3, Jun-Jul 2013.

[12] G. Moore, "Cramming more components onto integrated circuits", Electronics Magazine, 19 April, 1965.

[13] Jarrod D. Luker and Vinod B. Prasad, "RISC System Design in an FPGA", Bradley University,

IEEE, pp.532-536. 2001

[14] J.L. Hennessy, "VLSI Processor Architecture", IEEE Trans.Computers, vol. C-33, no. 12, pp. 1221-1246, 1984.

[15] Nilam Patel, Prof. J.H.Patil, "FPGA Based Implementation of 16-bit RISC Microcontroller",International Journal of scientific Research, Volume 4, Issue1, January 2015.

[16] Paul P. Chu, Deepak R.Mithani, "32 bit extended function Arithmetic-logic unit on single chip", U.S. Patent Document, Vol. 3, Issue 7, July 1984.

[17] Rahul R.Balwaik, Yogesh M. Jain, Amutha Jeyankar, "VLSI design of 16-bit processor", International Journal of VLSI and Embedded Systems-IJVES ISSN-2249, Vol04, June 2013.

[18] S. de Pablo, J.A. Cebrián, L.C. Herrero, A.B. Rey (2006), "A very simple 8-bit RISC processor for FPGA", RISCuva1 FPGA world 2006.