International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-Jun-2026, ISSN (Online): 2395-566X

High-performance FPGA- based ALU using reversible and Quantum-inspired logic

Mr. B. Ajantha Reddy[1], Mr. A. Prasad[2], Dr. A. Ranganayakulu[3], Pagadala Ananthalakshmi[4], Jestadi Divya Jyothika[5], Vyja Swetha[6], Bathula Persis[7]

[1]Assistant Professor, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.
[2]Associate Professor, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.
[3]Professor & HOD, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.
[4,5,6,7] Student, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur

Abstract- Computing devices, mobile phones, and computers all rely on the Arithmetic Logic Unit (ALU), a critical subsystem inside processors, to carry out the arithmetic and logical operations necessary for digital system operations. There is an urgent need for more energy-efficient alternatives in digital system design to standard ALUs designed using non-reversible logic gates, which are known for their considerable power consumption. Our proposed solution to this problem is a 32-bit ALU that makes use of reversible logic gates; this will allow us to cut down on power consumption while simultaneously increasing computing performance. Our suggested 32-bit ALU uses reversible logic gates to provide a solution that reduces power consumption and increases computational efficiency; this should lead to a revolution in digital system design. Not only do we want to reduce power consumption, but we also want to increase the ALU's usefulness and adaptability in other computing contexts by adding a full set of sixteen separate operations. Our goal is to set a new standard for ALU design with this novel method, one that puts computing power and power efficiency first. Our 32-bit ALU's use of reversible logic gates is a giant leap forward in the quest for power-efficient digital computers that don't sacrifice processing capability. We seek to solve the essential demand for energy-efficient solutions in today's technology-driven world by contributing to the progress of digital system design with painstaking attention to detail and a focus on innovation. Metaphors: Arithmetic Logic Unit (ALU), Reversible Logic Gates, Digital System Design, Energy Efficiency, Versatility, Innovative Architecture, Low Power Consumption, Computational Performance.

Keywords- Arithmetic Logic Unit (ALU), Reversible Logic Gates, 32-Bit ALU, Digital System Design, Low Power Consumption, Energy Efficiency.

I. INTRODUCTION

Arithmetic and logical operations on binary integers are handled by the ALU, a basic component of the CPU. Among its many uses are the mathematical operations AND, OR, XOR, and NOT, as well as multiplication and division. Constructed from logic gates and combinational logic circuits, it handles control unit and register inputs from the central processing unit. The maximum size of binary numbers that may be processed simultaneously is defined by the bit width. Recent developments have focused on developing specialized procedures for various tasks in an effort to increase speed, efficiency, and parallelism. Modern computer efficiency relies on reducing power consumption without sacrificing performance. New approaches, such as reversible logic gates, are aiming to achieve this.

Arithmetic Logic Unit

One of the most important parts of a central processing unit (CPU) is the Arithmetic Logic Unit (ALU), which is in charge of performing logical and mathematical operations on binary values. Operations like adding, subtracting, multiplying, and dividing are all part of it, as are logical operations like AND, OR, XOR, and NOT. Inputs from the central processing unit's control unit and registers are processed by the ALU, which is made up of combinational logic circuits. The maximum size of binary integers that may be processed in a single operation is defined by its bit width. Modern ALUs use task-specific specialized operations to boost speed, efficiency, and parallelism. New methods are being developed to decrease power usage without compromising computing performance. One such method is reversible logic gates. In the end, the ALU

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-Jun-2026, ISSN (Online): 2395-566X

has a major impact on how contemporary computers work and how efficient they are.

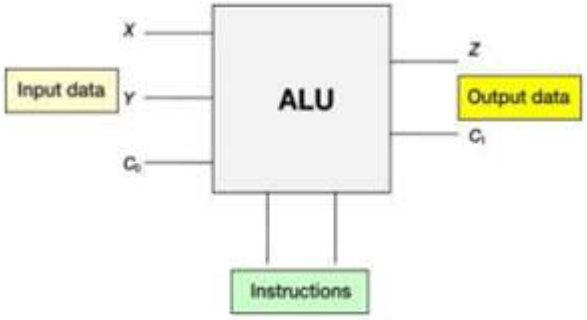

Fig.1.1: Basic Block Diagram of ALU

II. EXISTING METHOD

Determining the supported operations is the first step in designing a 32-bit irreversible ALU with 16 operations. These operations might include arithmetic, logical, and comparison operations. There would be one 32-bit output and two 32-bit inputs in the ALU. A number of mathematical and logical operations, including addition, subtraction, multiplication, and division, as well as shifts (left and right) and comparisons (greater than, less than), would be put into place. In order to choose actions depending on input signals, control logic would be developed. Using algorithms or logic gates that cannot be reversed guarantees irreversibility. To ensure the ALU is accurate and works as intended, thorough testing and documentation are required.

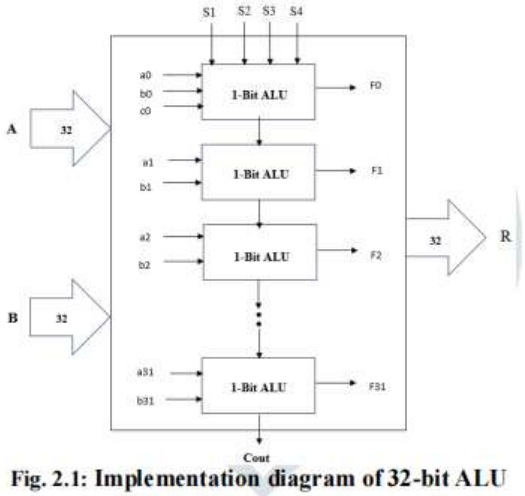

Fig. 2.1: Implementation diagram of 32-bit ALU

The 32-bit Arithmetic Logic Unit (ALU) seen in the figure is an essential part of a central processing unit (CPU) that performs logical and arithmetic operations on binary data. It handles data in 32-bit chunks and is made up of separate 1-bit ALUs. The inputs of a control signal are the operands, the control signals, and the carry-in, while the outputs are the results, the carry-out, and the status flags. It manages operations like logical AND and arithmetic with carry propagation by manipulating individual bits using bitwise operations. In order to handle data efficiently inside a computer system, this design highlights the process of decomposing complicated activities into smaller pieces.

III. PROPOSED METHOD

Reversible logic gates

A number of upcoming computer systems are beginning to rely on reversible logic, which is defined as a system in which every input vector generates an identical output vector. Quantum computing cannot be realized without its implementation. Minimizing the amount of trash outputs, circuit depth, and quantum cost are the main goals while constructing reversible logic circuits. The underlying principle of quantum computers is reversibility, which is why reversible circuits are essential. Two major limitations, however, apply to reversible logic circuits:

1. Unlike regular logic circuits, reversible logic circuits do not support fan-out, which means that it is not possible to duplicate a signal.
2. Reversible logic circuits are not designed to include feedback or loops, which brings us to our second point.
3. The following features must be guaranteed while building reversible logic circuits:
4. Minimal use of reversible gates: To maximize circuit efficiency, use the fewest reversible gates feasible.
5. Keep trash outputs to a minimum: Don't let the circuit produce too many outputs that don't help with the calculation at hand and may even hurt its performance.

Sixth, use constant inputs sparingly; doing so will simplify circuit functioning and make better use of resources.

NOT Gate

The NOT gate, which takes in only one input signal, is the most basic kind of reversible gate. When using a 1x1 gate configuration, the NOT gate incurs no quantum cost. Its primary purpose is to take an input A and produce an output P that is the inverse of A. In other words, if A = 0, then P = 1, and vice versa if A = 1, then P = 0. This gate is essential in many circuit designs because to its simplicity and absence of quantum cost.

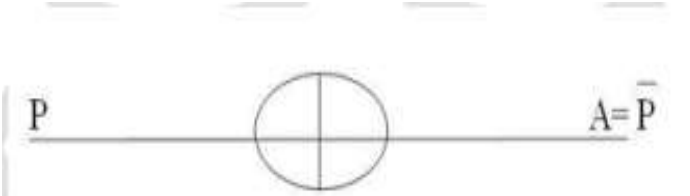

Fig 3.1.1: Not gate

Feynman Gate

A 2x2 reversible gate, the Feynman gate is often used for fan-out purposes in computers and is also known as the Controlled NOT gate. It takes two variables, A and B, as inputs and returns two sets of values, P = A and Q = A ⊕ B, where ⊕ means XOR. Because it is the only 2x2 reversible gate available for a quantum cost of one, it is often used for fan-out operations in various circuit designs.

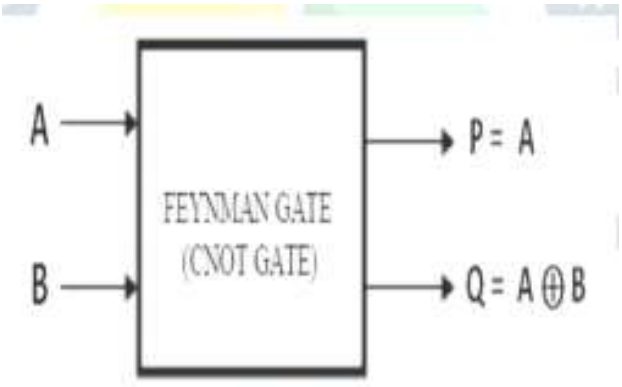

Fig 3.1.2: Feynman Gate

Toffoli Gate

The 3x3 Toffoli gate, which is also called the CCNOT gate, takes three 3x3 inputs labeled A, B, and C, and outputs P = A, Q = B, and R = A.B ⊕ C (where '.' signifies logical AND and "⊎" implies XOR). It generates 2 trash outputs and has a quantum cost of 5. Toffoli gates, invented by Tommaso Toffoli, are known as universal reversible logic gates. This means that they may be used to build any reversible circuit.

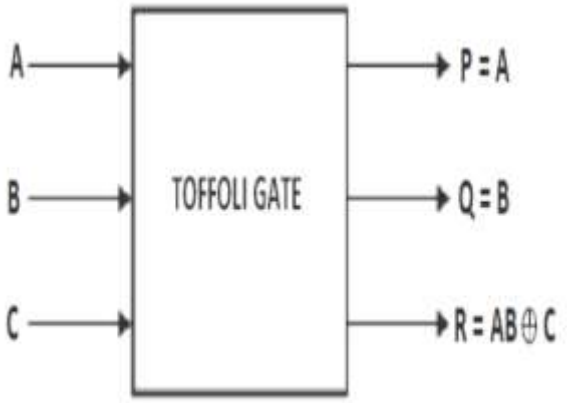

Fig 3.1.3: Toffoli gates

Peres Gate

P = A, Q = A × B, and R = A, B ⊎ C are the outputs of the Peres gate, which is also called the 3x3 reversible gate. Here, ⊎ means XOR and, indicates logical AND. The gate takes three inputs, A, B, and C. Its efficiency—as seen by its low quantum cost of 4—makes it a popular choice in many circuit designs. Because of its low quantum cost, a single Peres gate is enough to construct a half adder.

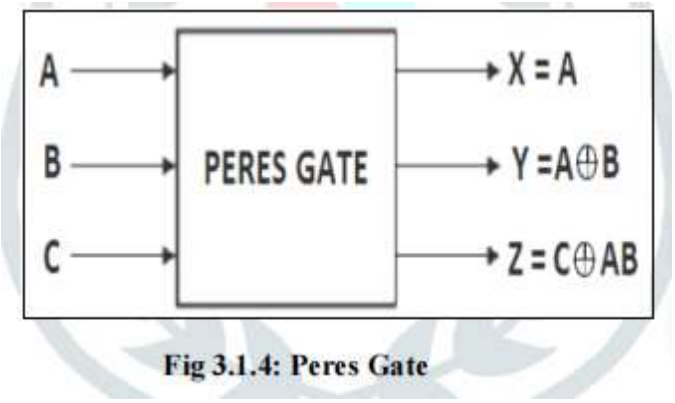

Fig 3.1.4: Peres Gate

Fredkin Gate

In quantum computing and reversible computing, there is a reversible three-bit gate called the Fredkin gate, which is also called the controlled swap gate or CSWAP gate. The state of the control bit determines how the Fredkin gate controls the swap operation it performs on its three input bits. By setting the control bit to 1, the second and third bits are swapped. When set to 0, the control bit maintains the bits' original state.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-Jun-2026, ISSN (Online): 2395-566X

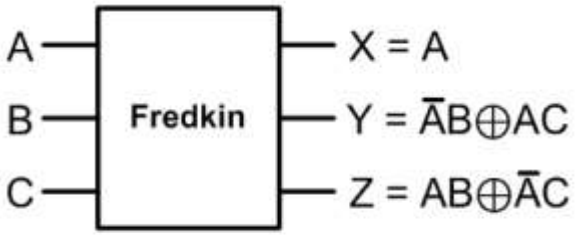

Fig 3.1.5: Fredkin Gate

bit Reversible ALU

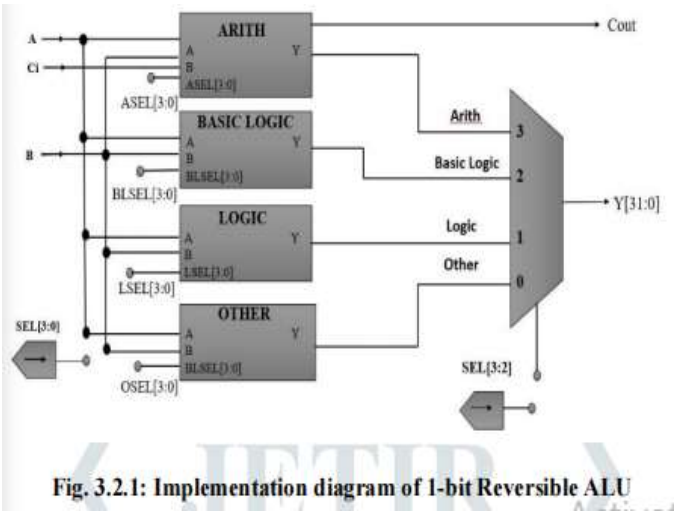

Fig. 3.2.1: Implementation diagram of 1-bit Reversible ALU

Thanks to its ability to do arithmetic, basic logic, advanced logic, and specialized operations all in one hardware unit, a 1-bit ALU exemplifies the efficiency and adaptability of digital systems. To meet a wide range of computing demands, the ARITH block adapts by carrying out arithmetic operations in response to the ASEL [1:0] control signal. You may use BLSEL [1:0] to choose basic logic operations like AND, OR, XOR, etc., and the BASIC LOGIC block handles all of that. Using LSEL as a guide, LOGIC block takes on difficult logical operations like shifting and multiplexing [1:0]. OSEL controls the OTHER block, which is responsible for specialized operations including data conversion and signal conditioning [1:0]. By allowing smooth mode changes, dynamic SEL [1:0] signals optimize resource use and minimize hardware overhead. Applications ranging from embedded systems to high-performance computers benefit greatly from this modular design's reduced complexity, increased scalability, and decreased power consumption.

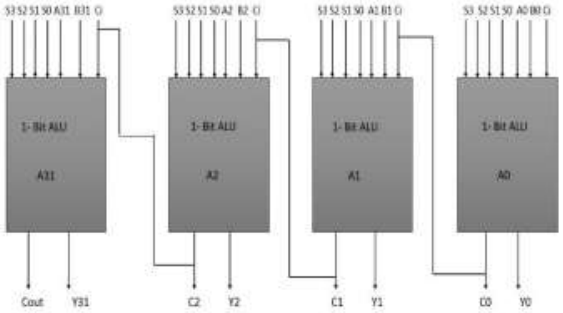

Fig 3.2.2: Implementation diagram of 32-bit Reversible ALU

The 32-bit ALU is an expert at performing logical and mathematical operations on binary data; it is composed of 32 separate 1-bit ALUs. Every one-bit ALU takes in two bits of data as input and processes them according to the instructions given by function signals. In order to perform logical and arithmetic operations involving several bits, the carry-out from one ALU is carried over to the next. The ALUs' combined outputs are sent to the 32-bit bus. The 32-bit ALU is optimized for calculations with a fixed width, but the multi-function digital logic unit is flexible enough to handle a wide variety of operations. When designing a digital system, both units are essential.

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

Two 32-bit input lines, "A" and "B," are available to the ALU. There are separate 1-bit ALUs for each of the A and B input bits (a0–a31 and b0–b31, respectively). In all, four control signals are present. These are the most important factors that

define the ALU's functioning.The ALU uses four selection lines and two 32-bit inputs to perform sixteen operations using reversible logic gates. Nevertheless, the tasks are detailed in the table.

Digital systems, such as Arithmetic Logic Units (ALUs) or comparable processing units, are typically represented by the control signals and their associated operations in the above table. A quick rundown of each procedure is this:

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.

IV. DESIGN METHODOLOGY

Using reversible logic gates, the project's methodology systematically designs, implements, and evaluates a 32-Bit ALU. The first step is to learn what is already known about reversible logic gates, ALU design, FPGA implementation, and related subjects by doing a comprehensive literature research. After that, we outline the precise needs and specifications for the design, including the supported logical and arithmetic operations, the input-output formats, and the performance goals like latency, power consumption, and speed. After these requirements are met, the 32-Bit ALU's logic design may begin, with an emphasis on creating reversible logic-compatible individual functional blocks such as adders, multipliers, and logical gates. After that, the ALU that was built is put into action on an FPGA platform by use of synthesis tools and hardware description languages (HDL). Subsequently, comprehensive testing and verification processes are implemented, which include both software and hardware testing, in order to confirm functionality and evaluate performance indicators. Power consumption, dissipation, speed, and latency are some of the metrics used to compare the reversible 32-Bit ALU's performance to those of conventional, non-reversible implementations. In order to make the reversible ALU design more efficient and effective, optimization strategies are investigated based on the assessment findings. Detailed documentation is kept throughout the process, culminating in a report that explains the project's approach, results, and conclusions. This report is intended to be sent to the right people and will help develop digital system design and low-power computing.

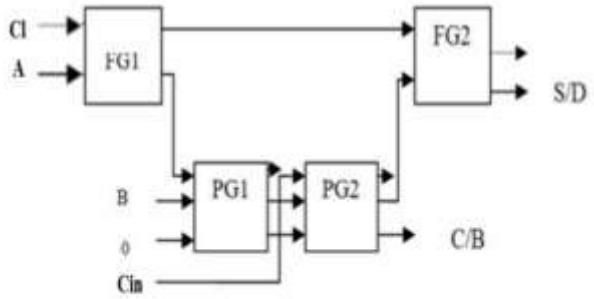

Fig 4.1: Full adder and Full subtractor

A complete adder/subtractor is shown in Figure 4.1 by use of Feynman gates and Perse gates. The control line dictates whether operation is used, addition or subtraction. When the control signal is turned off, the circuit adds two numbers. When set to 1, the control signal causes the circuit to subtract.

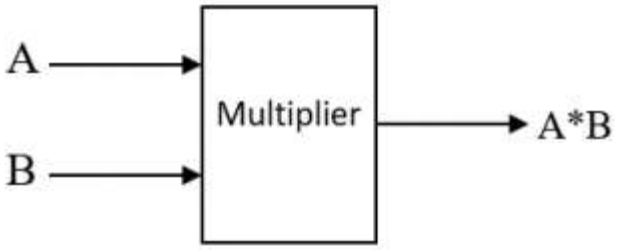

Fig 4.2: Multiplier

As shown in Figure 4.2, the multiplier block takes two parameters, A and B, and uses them to perform the

multiplication operation. This block performs a multiplication by bits.

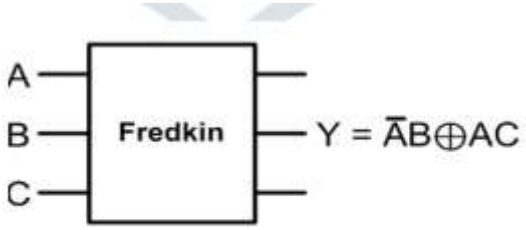

Fig 4.3: OR Operation

As shown in Figure 4.3, a Fredkin gate is used to carry out the logical OR operation in an OR operation block. With two inputs denoted as A and B, the Fredkin gate executes the OR operation on them.

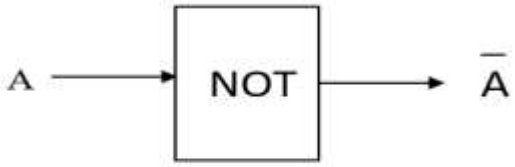

Fig 4.4: NOT Operation

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

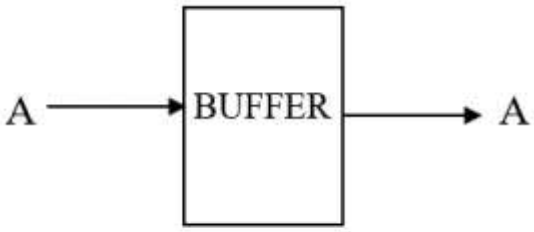

Fig 4.5: Buffer

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

Fig 4.6: AND Operation

A Peres gate is used to conduct the logical AND operation by passing input c as zero in Figure 4.6, which displays an AND operation block. The three inputs labeled A, B, and C are used to execute the AND operation via the Peres gate.

Fig 4.7: XOR Operation

In Figure4.7, we can see an XOR operation block in action, with the logical XOR operation carried out by means of a Peres gate. The Peres gate does an XOR operation on the three inputs that are designated A, B, and C.

V. SCHEMATIC VIEWS

Fig 5.1: RTL schematic of 32-bit reversible ALU

The RTL schematic of a 32-bit reversible ALU is shown in Figure 5.1. The use of reversible decoder-controlled combinational circuits in a 32-bit reversible ALU is advantageous and offers many benefits. Compared to traditional designs, it significantly reduces dynamic power consumption by 1.6 times, uses just 3% of FPGA memory, and conserves a remarkable 91% of space. In conclusion, the ALU's RTL design provides improved computational efficiency with

the use of reversible logic, efficient decoding, and optimum power usage.

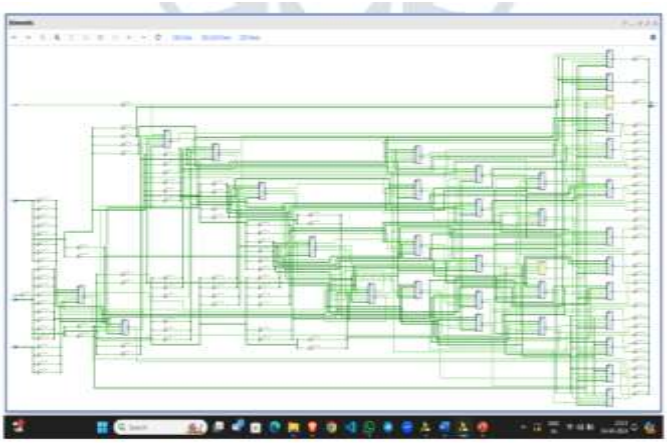

Fig 5.2: Technology schematic of 32-bit Reversible ALU

A 32-bit reversible ALU is seen in Figure 4.2, which is the technology schematic. You can see the layout and connections of the components of a 32-bit Reversible ALU in the technical diagram. In order to carry out tasks efficiently, it incorporates decoder-controlled circuits and reversible logic concepts. While decoders choose operations according to control signals, gates such as Toffoli and Fredkin make invertible operations possible. Optimization at the gate level and careful layout design minimize power consumption, area, and delay, ensuring efficiency. Memory and area optimization techniques lessen the strain on FPGA resources, making them more efficient and smaller. Optimization of performance is achieved by rigorous modeling and verification procedures, with an emphasis on computing speed, reliability, and scalability.

6.SIMULATION RESULTS

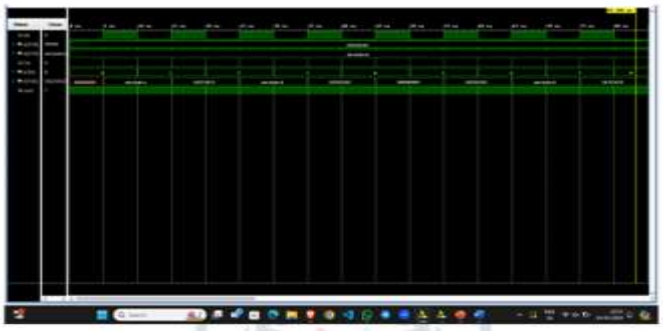

Fig 6.1: Output waveform of first 8 operations.

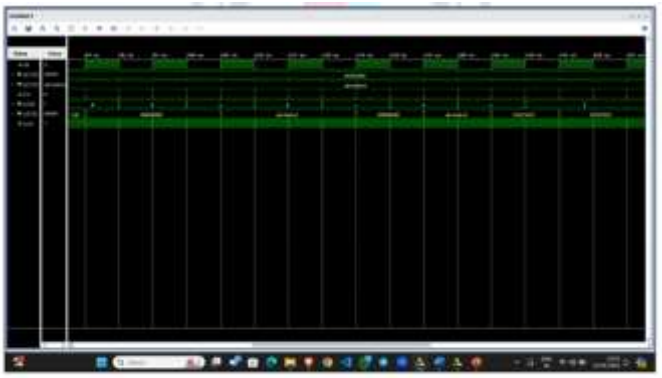

Fig 6.2: Output waveform of next 8 operations.

An overview of the control signals utilized in digital systems, each related to unique operations, may be seen in the simulated waveform. To aid in configuring the behavior of components like the ALU, these signals are represented using binary codes. Engineers can't build successful digital systems without first comprehending these signals, which allow them to generate unique instructions and efficiently regulate data flow. To put it simply, the table is a basic reference for digital system designers looking to implement and understand control signals.

VI. CONCLUSION

Using the Peres, Fredkin, Feynman, and Toffoli gates, we developed a 32-bit reversible ALU in this study. Enhancing ALU's performance is the primary goal of this work. The most recent design not only compares the delay and power dissipation of reversible and irreversible gates, but it also beats earlier designs in terms of power efficiency. Surprisingly, it shows a great deal of area optimization, using just 3% of the overall area of the FPGA. The significant reduction in power consumption from 22.488W to 1.136W and latency from 10.636ns to 5.255ns are particularly noticeable in irreversible gate logic.

Based on fundamental reversible gates, this design can be easily applied to encoders and other circuit designs. Reversible logic's adaptability and versatility can be further demonstrated by utilizing gates such as CNOT, DNG, and HNG to further reduce power consumption, delay, and area. It demonstrates how basic ideas may be extended imaginatively to different levels of circuit complexity, leading to new solutions in logic design.

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 GateBased 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.