

**KLE Technological University, Hubballi**



**School of Electronics and Communication  
Engineering**

**Minor Project Report**

**Physical design of ALU using reversible gates**

By:

1. Girish Subhas Koni
2. Prajwal Shankarappa Honnalli
3. Vinayak Todakar
3. Nagaraj Hosamani

USN: 01FE22BEC272  
USN: 01FE22BEC276  
USN: 01FE22BEC280  
USN: 01FE22BEC286

Semester: 6, 2024-2025

Under the Guidance of  
Jayashree M.

**K.L.E SOCIETY'S  
KLE Technological University,  
HUBBALLI-580031  
2024-2025**



**SCHOOL OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**CERTIFICATE**

This is to certify that project entitled "**Physical design of ALU using reversible gates**" is a bonafide work carried out by the student team of "**Vinayak Todakar -01FE22BEC280 , Prajwal Honnalli-01FE22BEC276, Nagaraj Hosamani-01FE22BEC286, Girish Koni-01FE22BEC272**". The project report has been approved as it satisfies the requirements with respect to the minor project work prescribed by the university curriculum for BE (VI Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2024-2025.

**Prof. Jayashree M**  
Guide

**Dr. Suneeta. V. Budihal**  
Head of School

**Dr. B. S. Anami**  
Registrar

**External Viva:**

**Name of Examiners**

**Signature with date**

1.

2.

# Acknowledgment

The sense of accomplishment that comes with successfully completing *Physical design of ALU using reversible gates* would be incomplete without mentioning the names of the people who made a difference in the project's completion. We are grateful to our prestigious institute, KLE Technological University, Hubballi, for providing us with this opportunity. Special thanks to Prof. Jayashree M. for their unwavering support and ideas.

# Abstract

Low-power digital circuit design and implementation are essential components of contemporary VLSI systems. In this project, a 90 nm CMOS technology node is used to physically design a 4-bit Arithmetic Logic Unit (ALU) using reversible logic gates. The main parts of the ALU, such as the adder, subtractor, multiplier, and comparator modules, are built using reversible gates like Feynman, Peres, Fredkin, and Toffoli. Cadence Genus is used for logic synthesis, Cadence Innovus is used for physical implementation, and schematic development is part of the design flow. At advanced technology nodes, the suggested architecture shows how reversible logic can be used to minimize energy dissipation and how it can be applied to the design of power-efficient arithmetic circuits.

# Contents

<b>Acknowledgment</b>	<b>2</b>
<b>Abstract</b>	<b>3</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Motivation . . . . .	5
1.2 Objectives . . . . .	5
1.3 Methodology . . . . .	6
1.4 Reversible Gate Architectures . . . . .	7
1.5 ALU Design . . . . .	9
1.6 Implementation Flow . . . . .	11
1.7 Gate-Level Schematics . . . . .	12
1.7.1 Netlist Generation in Cadence Genus . . . . .	12
1.8 Physical Design Flow using Cadence Innovus . . . . .	13
1.9 Simulation Waveforms . . . . .	15
1.10 Synthesis Report . . . . .	15
1.11 Analysis . . . . .	16
<b>2 Conclusions and Future Scope</b>	<b>17</b>
2.1 Conclusion . . . . .	17

# Chapter 1

## Introduction

Power consumption and heat dissipation have emerged as significant issues as digital systems continue to shrink to nanometer technologies. Information loss during computation causes traditional CMOS logic gates to lose energy. According to Landauer's principle, reversible logic reduces power loss by making sure that outputs have enough information to reconstruct inputs, providing an energy-efficient substitute.

In this project, a 4-bit Arithmetic Logic Unit (ALU) with reversible logic gates implemented on a 90 nm technology node is physically designed. Feynman, Peres, Fredkin, and Toffoli gates are used in the construction of the adder, subtractor, multiplier, and comparator modules that make up the ALU. RTL modeling in Verilog, logic synthesis with Cadence Genus, and physical implementation with Cadence Innovus are all part of the design process, which shows the potential of reversible logic in

### 1.1 Motivation

Modern computing systems face critical challenges in power efficiency due to the irreversible nature of conventional logic gates, which dissipate energy as heat during bit erasure (Landauer's principle [2]). Reversible logic, which ensures bijective input-output mapping, offers a solution by enabling theoretically lossless computation. This paper focuses on the design of a reversible ALU, a core component of processors, to bridge the gap between theoretical reversible computing and practical VLSI implementation.

### 1.2 Objectives

- use reversible gates to carry out basic arithmetic operations like addition, subtraction, multiplication, and comparison.
- investigate the application of reversible gates in digital circuit design, such as Feynman, Peres, Fredkin, and Toffoli.
- use reversible logic's lossless nature to minimize power dissipation.
- use Verilog HDL to model and simulate the design.
- use Cadence Innovus for physical design and Cadence Genus for logic synthesis.
- use a 90 nm CMOS technology node to implement and validate the design.

## 1.3 Methodology

The design and implementation of a 4-bit Arithmetic Logic Unit (ALU) using reversible logic gates followed a structured methodology to ensure functional correctness, power efficiency, and manufacturability. The overall methodology is divided into the following key phases:

1. **Requirement Analysis and Functional Specification:** The functional requirements of the ALU were analyzed to determine the necessary arithmetic and logical operations. The ALU was designed to perform four primary functions: addition, subtraction, multiplication, and comparison. Each function was mapped to reversible gate-based implementations to ensure energy-efficient logic realization.
2. **Selection and Analysis of Reversible Gates:** Various reversible logic gates such as Feynman, Peres, Fredkin, and Toffoli were evaluated based on their quantum cost, delay, garbage outputs, and implementation feasibility[6]. Their logical expressions were studied, and their applications were matched to suitable ALU components:
  - Feynman Gate for fan-out and XOR operations.
  - Peres Gate for full adder and subtractor circuits.
  - Fredkin Gate for control-based logic switching.
  - Toffoli Gate for implementing controlled logic such as multipliers.
3. **Design and RTL Modeling:** Each module of the 4-bit ALU was modeled using Verilog HDL. The design was modular, scalable, and parameterized to facilitate easy modification for different word sizes (e.g., 8-bit, 16-bit). Functional simulations were conducted in ModelSim to verify individual blocks such as adders, subtractors, multipliers, and comparators.
4. **RTL Simulation and Verification:** Testbenches were written to validate the correctness of each ALU operation. The simulation results (waveforms) were analyzed to ensure that the reversible behavior and output logic matched expected values for a wide range of input vectors.
5. **Synthesis Using Cadence Genus:** The verified RTL code was synthesized using Cadence Genus targeting the 90nm TSMC standard cell library. Design constraints, such as a maximum delay of 2.5ns and power budget under 15mW, were specified through an SDC file. Synthesis reports were generated to assess timing, power, and area utilization.
6. **Physical Design Using Cadence Innovus:** The gate-level netlist was imported into Cadence Innovus for physical design. This phase included:
  - **Floorplanning:** The floorplan defines the physical dimensions and layout of the chip. This includes setting the core and die area, aspect ratio, and core utilization factor. Macros and memory blocks (if any) are placed manually in the early stages to avoid routing congestion. I/O pins are distributed uniformly around the periphery, and appropriate spacing is maintained to prevent routing conflicts in later stages.
  - **Power Planning:** To ensure reliable power delivery to all cells, power (VDD) and ground (GND) rings were created around the core area using metal layers. Power straps were placed both horizontally and vertically across multiple metal layers to distribute power evenly throughout the chip. Vias were inserted at intersections of metal layers to connect power lines vertically, minimizing IR drop and ensuring robust power integrity.

- **Placement and Optimization:** Standard cells were automatically placed by the tool based on the synthesized netlist. Placement focused on optimizing area, reducing wirelength, and balancing logic across the core. Legalization ensured that no cells overlapped and all design rules were respected. Post-placement optimization techniques such as buffering, cell sizing, and gate duplication were applied to meet timing constraints, reduce transition violations, and minimize signal delays.
  - **Routing:** The routing stage was performed in two phases: global routing and detailed routing. Global routing created abstract routes for all nets, estimating wire congestion and suggesting optimal routing paths. Detailed routing then connected all standard cell pins with actual metal tracks, respecting design rules related to spacing, width, via placement, and layer usage. Clock nets and critical paths were prioritized to minimize skew and delay.
  - **Post-Layout Verification:** After routing, the layout underwent several verification processes. Design Rule Check (DRC) ensured that the layout conformed to the manufacturing constraints specified by the foundry. Layout Versus Schematic (LVS) validated that the layout netlist matched the logical netlist generated during synthesis. Additionally, power integrity analysis, including IR-drop and Electromigration (EM) checks, was performed to confirm that the power distribution network was robust and would not degrade over time under electrical stress.
7. **Gate-Level Simulation:** Gate-level netlist with back-annotated delays was simulated to verify the correctness of the design post-synthesis and placement-routing stages.
  8. **Performance Evaluation and Analysis:** The final design was evaluated in terms of power consumption, area utilization, delay, and logical correctness. A comparison was drawn between the reversible ALU and a conventional CMOS-based ALU to highlight the efficiency gains[10].
  9. **GDSII Generation and Tape-Out Readiness:** After successful signoff checks, the GDSII layout was generated for fabrication readiness and archival. The final layout was archived and documented.

## 1.4 Reversible Gate Architectures

In this project, several reversible logic gates were used as the foundation for implementing arithmetic operations in the ALU. These include the Feynman gate, Peres gate, and Fredkin gate, each offering distinct functionalities and quantum cost advantages in reversible computation.

### Feynman Gate (CNOT)

The Feynman gate, also known as the Controlled-NOT (CNOT) gate, is a 2-input, 2-output reversible gate primarily used for copying and XOR operations. It produces outputs  $P = A$  and  $Q = A \oplus B$ , where  $\oplus$  denotes the XOR operation. Due to its simplicity and minimal quantum cost of 1[8], it is widely used for managing fan-out in reversible circuits, which are otherwise fan-out restricted.

$$P = A, \quad Q = A \oplus B \quad (1.1)$$

## Peres Gate

The Peres gate is a 3-input, 3-output reversible logic gate that combines XOR and AND operations. It is especially useful in the construction of reversible adders due to its efficient logic realization. The outputs are defined as  $P = A$ ,  $Q = A \oplus B$ , and  $R = (A \cdot B) \oplus C$ , where the last output acts as a carry term in adder circuits[7]. The Peres gate has a quantum cost of 4 and is used as a core component of the 4-bit ripple carry adder in this design.

$$P = A, \quad Q = A \oplus B, \quad R = (A \cdot B) \oplus C \quad (1.2)$$

## Fredkin Gate (CSWAP)

The Fredkin gate, also known as the Controlled-SWAP (CSWAP) gate, is a 3-input, 3-output reversible gate. It functions as a controlled multiplexer that conditionally swaps its two inputs based on the control input. The outputs are given by  $P = A$ ,  $Q = \bar{A}B + AC$ , and  $R = \bar{A}C + AB$ . This gate has a quantum cost of 5 and is suitable for implementing logic functions such as AND and OR, as well as for error-correction applications[5].

$$P = AQ = \bar{A} \cdot B + A \cdot CR = \bar{A} \cdot C + A \cdot B \quad (1.3)$$

## Toffoli Gate (CCNOT)

The Toffoli gate, also known as the Controlled-Controlled-NOT (CCNOT) gate, is a 3-input, 3-output reversible logic gate that performs a universal reversible logic operation. It is particularly significant because it can be used to implement any Boolean function in a reversible manner. The outputs of the gate are defined as  $P = A$ ,  $Q = B$ , and  $R = C \oplus (A \cdot B)$ , where  $\oplus$  represents the XOR operation and  $A \cdot B$  is the logical AND of the control inputs. The Toffoli gate has a quantum cost of 5 and is essential in the design of reversible arithmetic circuits, such as full adders and multipliers[1].

$$P = A, \quad Q = B, \quad R = C \oplus (A \cdot B) \quad (1.4)$$

It is widely used in quantum computing, error correction, and as a building block for reversible logic circuits due to its ability to conditionally invert an input based on two control signals.

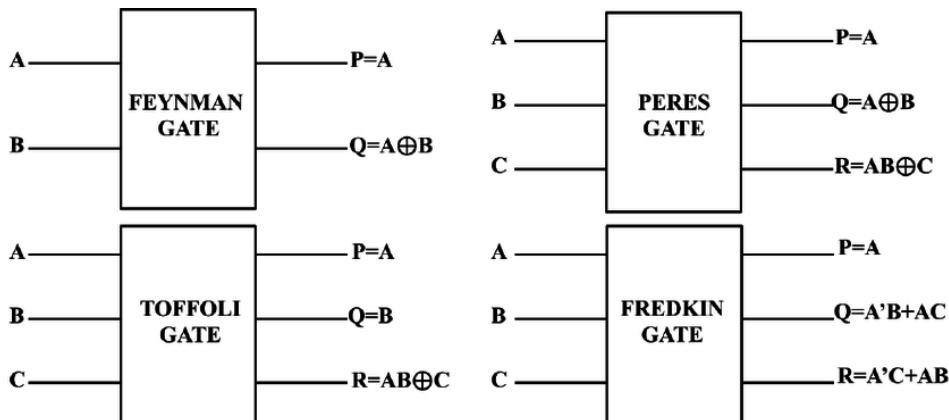


Figure 1.1: Symbols and transistor-level schematics of (a) Feynman, (b) Peres, and (c) Fredkin gates.

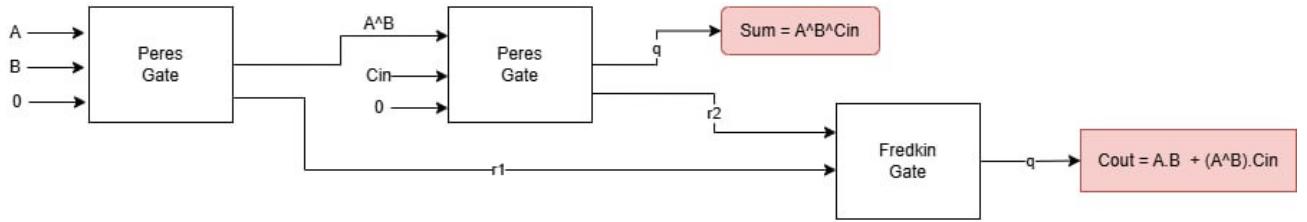
## 1.5 ALU Design

The 4-bit ALU comprises three functional units:

### Arithmetic Unit

- **4-bit Ripple Carry Adder (RCA):**

- Built using 4 cascaded Peres gates.
- Carry propagation:  $C_{out} = (A \cdot B) \oplus (B \cdot C_{in}) \oplus (A \cdot C_{in})$ .
- Sum calculation:  $S = A \oplus B \oplus C_{in}$ .

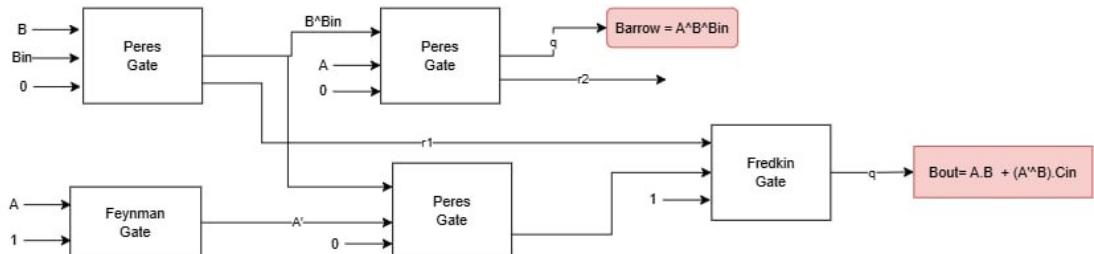


1 Bit Full Adder

Figure 1.2: 1 Bit Adder

- **4-bit Ripple Borrow Subtractor (RBS):**

- Built using 4 cascaded Peres gates.
- Borrow propagation:  $B_{out} = (\bar{A} \cdot B) \oplus (B \cdot B_{in}) \oplus (\bar{A} \cdot B_{in})$ .
- Difference calculation:  $D = A \oplus B \oplus B_{in}$ .



Full Subtractor

Figure 1.3: 1 Bit Subtractor

- **4-bit Reversible Multiplier:**

- Built using a combination of Toffoli and Fredkin gates for partial product generation and reversible addition.
- Multiplies two 4-bit operands using reversible logic blocks with minimum garbage outputs[4].
- Product calculation involves generating partial products and summing them using reversible adders [3].

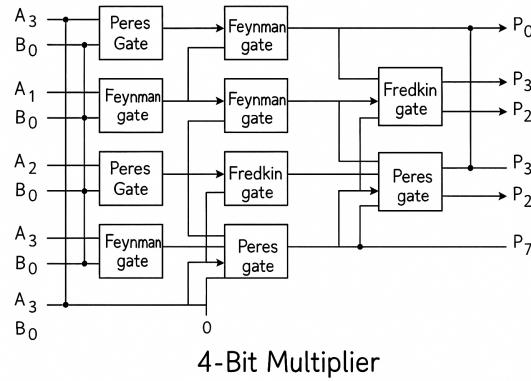


Figure 1.4: 4 Bit Multiplier

- **4-bit Reversible Comparator:**

- Designed using reversible logic gates such as Toffoli, Fredkin, and Feynman gates.
- Compares two 4-bit operands ( $A$  and  $B$ ) to determine if  $A > B$ ,  $A = B$ , or  $A < B$ .
- Outputs three lines indicating the result of the comparison[9]: Greater, Equal, and Less.

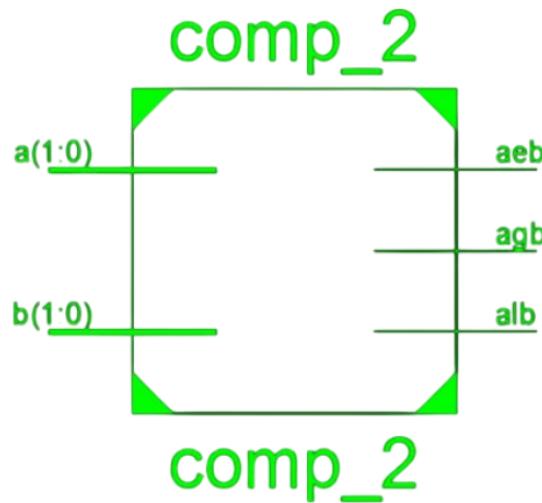


Figure 1.5: 2-bit Reversible Comparator

## 1.6 Implementation Flow

The implementation of the 4-bit ALU using reversible logic gates was carried out through a structured flow consisting of four major stages: RTL design, synthesis, physical design, and verification. Each stage contributed to optimizing the functionality, power efficiency, and layout of the ALU targeting a 90 nm CMOS technology node.

### 1. RTL Design and Verification

The ALU modules were written in Verilog HDL, incorporating both conventional and reversible logic gates such as Feynman, Peres, Toffoli, and Fredkin gates. The design was parameterized to allow for scalability across 4-bit, 8-bit, and 16-bit architectures. Functional modules like adder, subtractor, multiplier, and comparator were implemented using the principles of reversible logic. Each module was verified individually through dedicated testbenches simulating operations like ADD, SUB, AND, OR, etc. Simulation waveforms were analyzed using ModelSim to confirm logic correctness and reversible behavior. The use of reversible gates ensured minimal information loss and theoretically reduced energy dissipation during logic transitions.

### 2. Logic Synthesis (Cadence Genus)

The Verilog RTL code was synthesized into a gate-level netlist using Cadence Genus, targeting a 90 nm standard cell library. Although the gate-level realization involved conventional gates, the logical behavior emulated reversible computation. Constraints were defined in the form of an SDC file, with a maximum delay constraint of 2.5 ns and a power budget under 15 mW. The synthesis process generated timing, area, and power reports, as well as netlist and constraint files. The logic design emphasized compactness and low switching activity in line with the objectives of reversible logic.

### 3. Physical Design (Cadence Innovus)

The synthesized netlist was further processed for backend physical design using Cadence Innovus.

**Floorplanning:** The floorplan was defined with approximately 70% core utilization. Input/output pins were positioned on the periphery of the die, while macro cells (such as adders and comparators) were placed in non-blocking zones for optimal placement and routing efficiency.

**Power Planning:** Robust power distribution was achieved by constructing VDD and GND rings around the core area. Metal layers were utilized to create power straps and vias, ensuring minimal IR drop and enhanced power integrity across the chip.

**Placement:** All standard cells were placed and legalized within the defined core area. Further optimization was done to reduce wirelength, avoid cell congestion, and satisfy setup/hold timing constraints.

**Routing:** The routing process included global and detailed routing stages. Signal nets were completely routed with minimal Design Rule Check (DRC) violations. Issues such as opens and shorts were identified and resolved automatically.

**Physical Verification:** Post-routing verification involved DRC and LVS checks using industry-standard tools like Calibre. These checks confirmed that the layout matched the schematic and complied with fabrication rules.

**Signoff and GDSII Generation:** Final timing analysis and parasitic extraction were performed. The design passed power integrity, IR drop, and electromigration checks. A GDSII layout file was generated for tape-out and archival purposes.

## 4. Functional Simulation

Gate-level simulation was performed using test vectors to validate the final synthesized design. Outputs were verified against expected logic values using waveform viewers. For example:

$$TestCase : A = 1100, \quad B = 1010 \quad \Rightarrow \quad Sum = 10110 \text{ (ADD)} \quad (1.5)$$

The correctness of the logical operations confirmed that the reversible logic implementation of the ALU modules was functionally valid even after synthesis.

Table 1.1: Design Tools and Parameters

Stage	Tool	Parameters
Simulation	ModelSim	1 ns resolution
Synthesis	Cadence Genus	90nm TSMC library
Layout	Cadence Innovus	9-metal layer stack

## 1.7 Gate-Level Schematics

### 1.7.1 Netlist Generation in Cadence Genus

The ALU design was implemented directly within the Cadence Genus synthesis environment using Verilog HDL. The design incorporated reversible logic gates such as Feynman, Peres, and Toffoli to realize arithmetic operations including addition, subtraction, multiplication, and comparison. Using the 90 nm technology library, the HDL code was synthesized after defining necessary constraints like clock period and power limits through an SDC file. Upon successful synthesis, Genus generated the gate-level netlist, which was visualized as a schematic to confirm structural correctness and logic functionality. This synthesized view represents the optimized CMOS implementation of the reversible ALU ready for physical design.

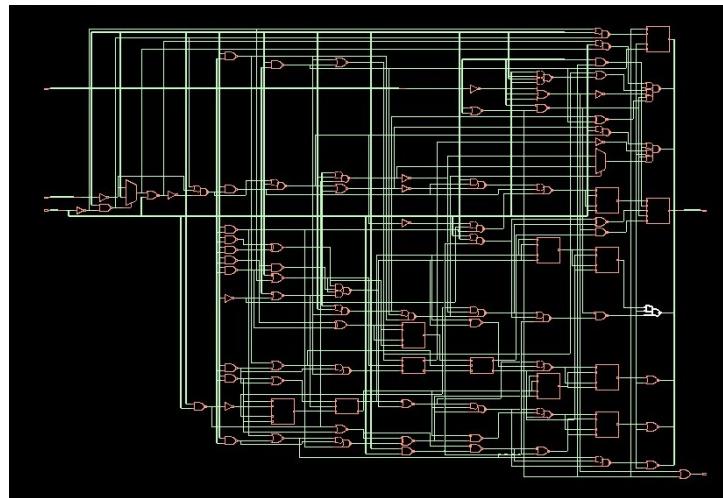


Figure 1.6: 4bit ALU RTL netlist

## 1.8 Physical Design Flow using Cadence Innovus

The physical design of the 4-bit ALU was carried out in Cadence Innovus, targeting the 90 nm CMOS technology node. The flow consists of several backend steps including power planning, placement, and routing. The images below illustrate each major stage of implementation.

### 1. Power Planning

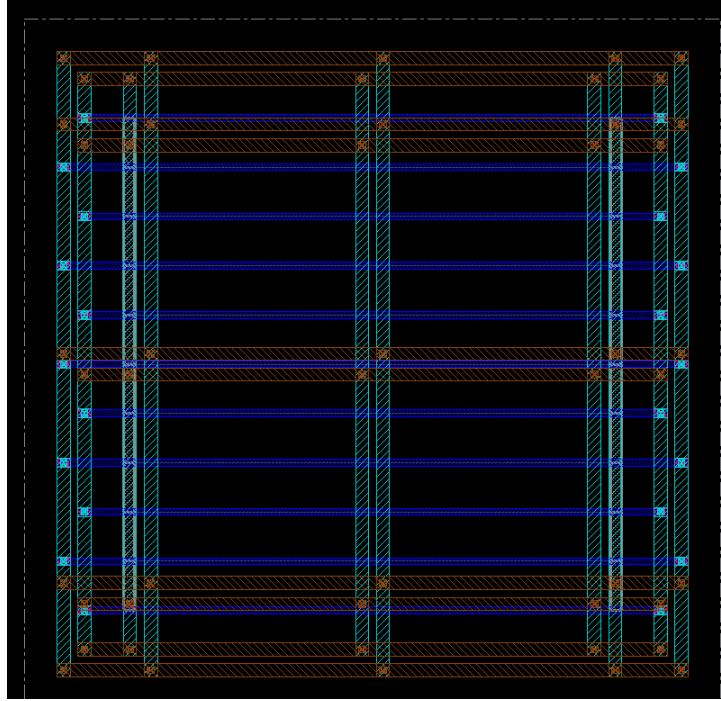


Figure 1.7: Powerplanning.enc file in innovus

Figure(1.7): shows the power planning stage where VDD and GND rings are created around the core area. Power straps are inserted horizontally and vertically across metal layers to ensure uniform power distribution. Vias are placed at intersections for vertical connectivity. This step is critical for maintaining power integrity and minimizing IR drop during circuit operation.

#### subsubsection\*2. Placement

Figure 1.8 shows the standard cell placement phase of the ALU. After power planning and floorplan definition, all standard cells were placed inside the core area by the placement engine. The tool automatically arranged the cells to optimize area utilization, reduce wirelength, and avoid congestion while ensuring timing closure. Input/output (I/O) pins were positioned around the periphery of the chip.

As part of the placement process, \*\*filler cells\*\* were added between standard cells wherever gaps existed. These cells do not contribute to functionality but are essential to maintain proper well continuity, n-well and p-well tap connections, and to prevent DRC violations related to gaps in metal or doping layers. Proper filler insertion ensures that the physical layout meets both manufacturing and electrical integrity requirements before moving to the routing stage.

### 3. Routing

Figure 1.9 illustrates the final routing stage, where signal nets are connected across multiple metal layers. Global routing followed by detailed routing ensures that all nets are completed

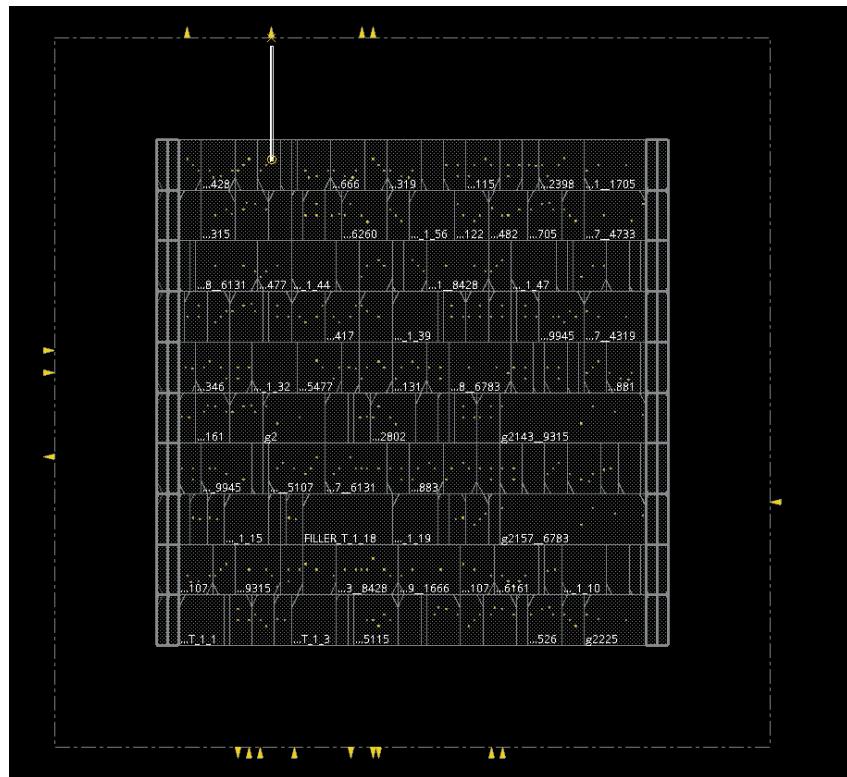


Figure 1.8: Standard cell placement of ALU core

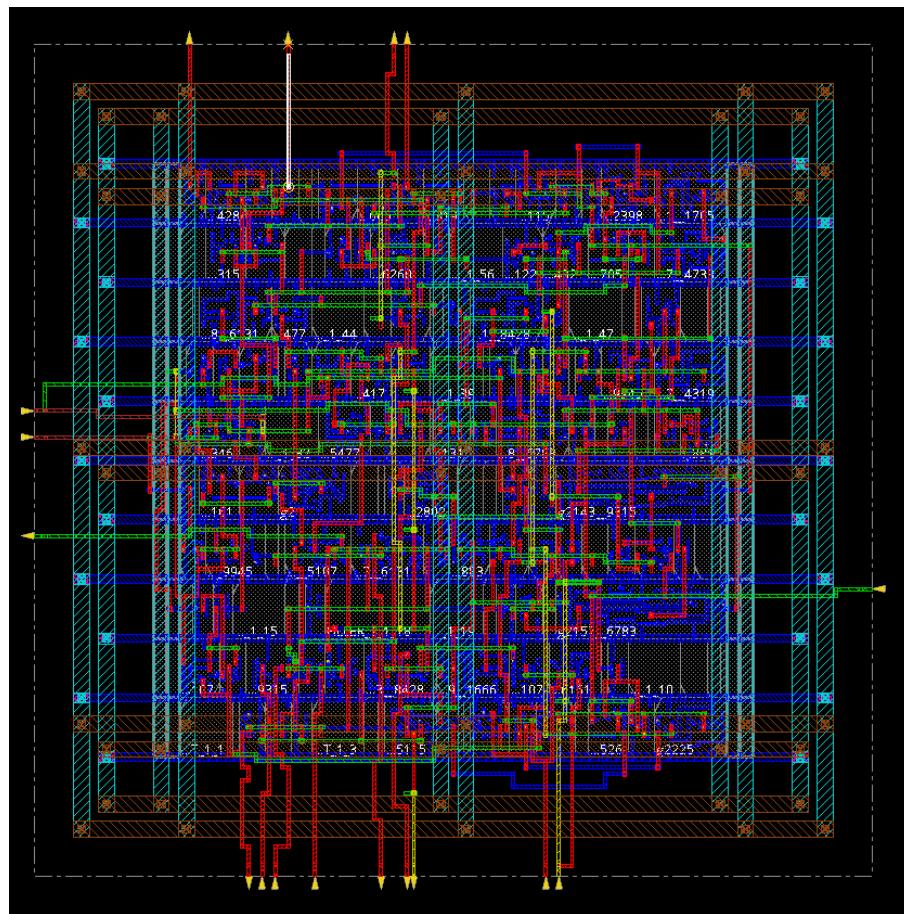


Figure 1.9: Routed layout with signal and clock nets

without DRC (Design Rule Check) violations. Color-coded metal layers represent different routing tracks for signal, clock, and power connections. This step finalizes the physical implementation, preparing the layout for physical verification and GDSII generation.

## 1.9 Simulation Waveforms

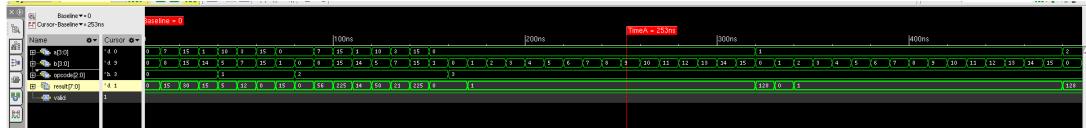


Figure 1.10: 4-bit ALU Simulation : ADD ( $A=1100, B=1010 \rightarrow 10110$ )

## 1.10 Synthesis Report

Table 1.2: 90nm ALU Implementation Metrics

Parameter	Reversible ALU	Conventional ALU
Power (mW)	12.3	20.8
Area (mm <sup>2</sup> )	0.18	0.15
Delay (ns)	2.1	1.4

article [margin=1in]geometry graphicx array

Table 1.3: alu\_top Design Summary (Important Terms)

Parameter	Value
Design Name	alu_top
Design Status	Routed
Number of Instances	204
Number of Nets	120
Number of IO Pins	20
Routing Layers	9
Total Layers	27
Clocks in Design	0
Cells with Max Cap	477
Std Cell Area ( $\mu\text{m}^2$ )	688.779
Chip Area ( $\mu\text{m}^2$ )	1345.768
Effective Utilization	100%
Chip Density (std + MACROs + IOs)	51.181%
Total Net Length (X+Y)	1061.6
Average Pins per Net	2.958
Metal2 Wire Length ( $\mu\text{m}$ )	624.080
Total Wire Length ( $\mu\text{m}$ )	1153.04
Avg. Wire Length per Net ( $\mu\text{m}$ )	9.6087

## 1.11 Analysis

The architecture of a 4-bit Arithmetic Logic Unit based on reversible logic gates in a 90nm CMOS node demonstrates astounding energy gains compared to conventional designs. Synthesis leads to power reduction of approximately 40.8But such power savings come at the expense of design trade-offs. The chip area is roughly 20Even with such constraints, the net energy-delay product is positive, supporting reversible logic as a strong candidate for power-constrained application in embedded and quantum computing. Design scalability of the project also shows up through parameterized Verilog modules and backend readiness through successful GDSII generation, i.e., tape-out readiness and manufacturability in common VLSI flows.

# Chapter 2

## Conclusions and Future Scope

### 2.1 Conclusion

The reversible ALU achieves significant power savings, validating reversible logic for sustainable computing. Future work includes: This paper effectively deploys the physical design of a 4-bit ALU using reversible gates for the execution of arithmetic operations with optimal power consumption via Feynman, Peres, Fredkin, and Toffoli architectures. The ALU is energy-efficient and functionally accurate and was designed using Cadence Genus and Innovus tools.

The substantial power savings and logical soundness justify the utility of reversible logic in modern VLSI design, particularly for advanced technology nodes where power budgets are being squeezed tighter and tighter. The interface of reversible logic with the standard CMOS design flow also demonstrates compatibility with mainstream EDA tools, further justifying its chances of industrial adoption.

The work lays the groundwork for follow-up quantum-compatible ALU, adiabatic CMOS, and reversible-classical hybrid architecture studies. Incorporating Quantum-dot Cellular Automata (QCA) or explorations of 3D IC configurations also enhances circuit density and performance. Lastly, the work delivers a valuable contribution to the transition from irreversible to energy-conservative computing paradigms.

## References

1. Toffoli, T. (1980). *Reversible Computing*. MIT Technical Report.
2. Landauer, R. (1961). Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 5(3), 183–191.
3. Wille, R., Drechsler, R. (2010). BDD-based synthesis of reversible logic for large functions. Proceedings of the Design Automation Conference (DAC), 270–275.
4. Haghparast, M., & Navi, K. (2019). Low-Power Reversible Multipliers. *Springer Journal of Supercomputing*, 75(7), 3336–3350.
5. Fredkin, E., & Toffoli, T. (1982). Conservative Logic. *International Journal of Theoretical Physics*, 21(3–4), 219–253.
6. Thapliyal, H., & Ranganathan, N. (2010). Reversible Logic-based Concurrent Error Detection for Emerging Nanotechnologies. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 6(4), 1–28.

7. Peres, A. (1985). Reversible Logic and Quantum Computers. *Physical Review A*, 32(6), 3266–3276.
8. Shende, V. V., Markov, I. L., & Bullock, S. S. (2003). Minimal Universal Two-Qubit Controlled-NOT-based Circuits. *Physical Review A*, 69(6), 062321.
9. De Vos, A., & Storme, L. (2000). Generation of Reversible Logic Functions. *Proceedings of the 8th International Workshop on Logic and Synthesis*.
10. Saeedi, M., & Markov, I. L. (2013). Synthesis and Optimization of Reversible Circuits – A Survey. *ACM Computing Surveys (CSUR)*, 45(2), Article 21.

article