AMRITA VISHWA VIDYAPEETHAM

Course Code: 19ECE383

Course Name: VLSI Design Lab

Component: System Design

Semester/Section: VI/C

Date of Evaluation:29-04-2025

Academic Year: 2024-2025 (Even Semester)

Batch Number for System Design: 8

Name of the Students: 1) K Pallavi ECE22224

With Registration no. 2) K Mahesh ECE22227

3) M L Shrividya ECE22228

Branch: Electronics and Communication Engineering

Semester, Section, Batch: VI,C,C1

Batch: 2022-2026

Faculty In-charge: Ms. Sonali Agrawal

Aim1:
To design and implement a 2-bit ALU capable of performing:

- 2-bit Addition
- 2-bit Subtraction using full adders and required logic gates
- 2-input XOR using Mirror circuit
- 2-input AND using minimum number of Transmission gates

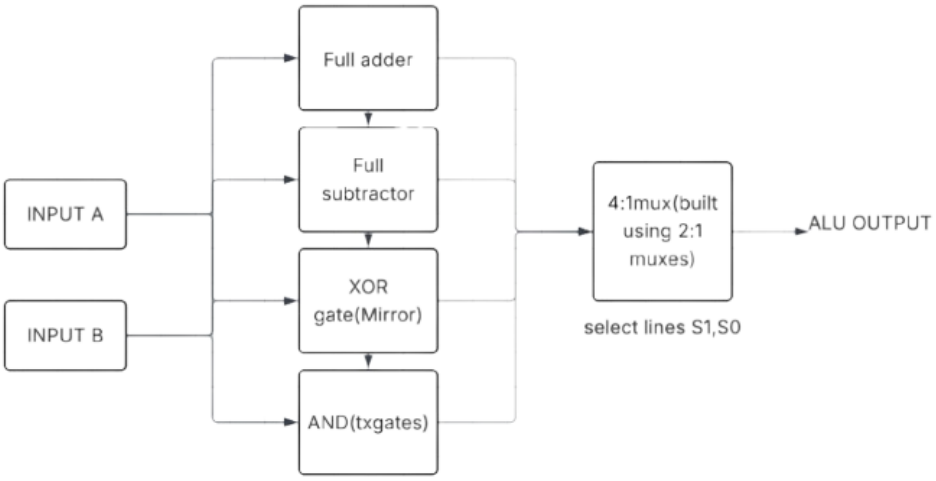
using Cadence Virtuoso with 45nm Technology.

Tool Used: Cadence Virtuoso

Technology Used: 45nm CMOS Technology

Theory:

1) **Introduction**
   An Arithmetic Logic Unit (ALU) is a critical component of a processor, performing arithmetic and logic operations. In this experiment, a 2-bit ALU is designed to perform addition, subtraction, XOR, and AND operations. Control logic selects the desired operation based on input signals.
2) **Block Diagram of the designed ALU with working principle**

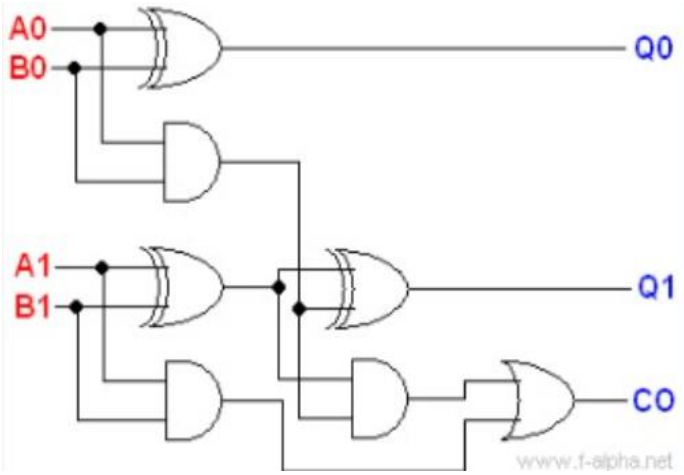


Inputs A and B are simultaneously fed to four blocks: a full adder, full subtractor, XOR gate, and AND gate. The full adder produces the sum of A and B, the full subtractor gives the difference (A - B), the XOR gate performs the bitwise XOR of A and B, and the AND gate performs the bitwise AND of A and B. The outputs from these four blocks are then given as inputs to a 4:1 multiplexer. Based on the select lines (S1, S0), the multiplexer selects one of these operations' results to pass to the ALU output.

S1	S0	Operation Selected	ALU Output
0	0	Full Adder	A + B
0	1	Full Subtractor	A - B
1	0	XOR Gate (Mirror)	$A \oplus B$
1	1	AND Gate (Txgates)	$A \cdot B$ (A AND B)

3) **Logic Circuit for each block of ALU with working principle (including Truth Table and Logical expressions)**

- **2-bit Addition:**
  Implemented using full adders cascading carry-out to next stage.

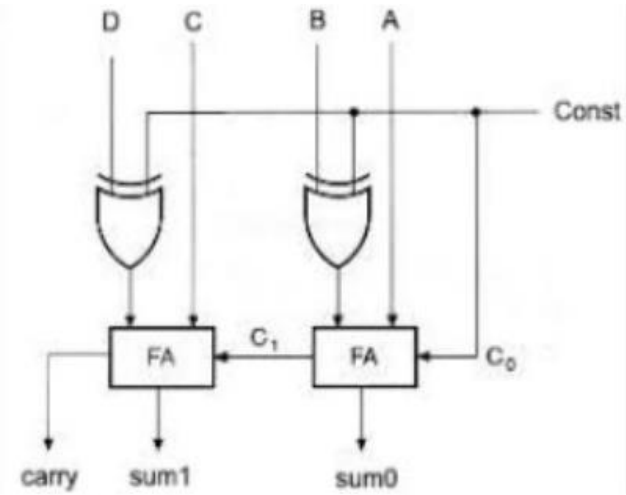


a	b	c	d	x	y	z
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	1	0	1
1	1	0	0	0	1	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	0

Logical expressions

- $S0 = A0 \oplus B0$
- $C0 = A0 \cdot B0$
- $S1 = A1 \oplus B1 \oplus C0$
- $C1 = (A1 \cdot B1) + (B1 \cdot C0) + (A1 \cdot C0)$

- **2-bit Subtraction**:
  Built using full adders and logic inversion (2's complement method).

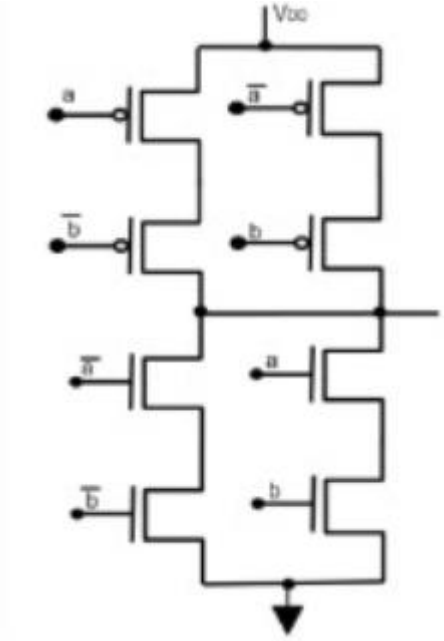


Input					Output		
V(const)	V(d)	V(c)	V(b)	V(a)	Carry	Sum1	Sum0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
0	1	0	0	1	0	1	1
0	1	0	1	0	0	1	1
0	1	0	1	1	1	0	0
0	1	1	0	0	1	0	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1
1	0	0	1	0	1	0	1
1	0	0	1	1	1	0	0
1	0	1	0	0	0	1	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	1	1
1	0	1	1	1	0	1	0
1	1	0	0	0	1	1	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	1	1
1	1	0	1	1	1	1	0
1	1	1	0	0	1	0	0
1	1	1	0	1	0	1	1
1	1	1	1	0	1	0	1
1	1	1	1	1	1	0	0

Logical expressions

- D0 = A0 ⊕ B0
- B0 = (~A0) · B0
- D1 = A1 ⊕ B1 ⊕ B0
- B1 = (~A1) · (B1 + B0) + (B1 · B0)

- **2-input XOR using Mirror Circuit:**
  Designed with a symmetrical (mirror) layout for reduced delay and power**.**

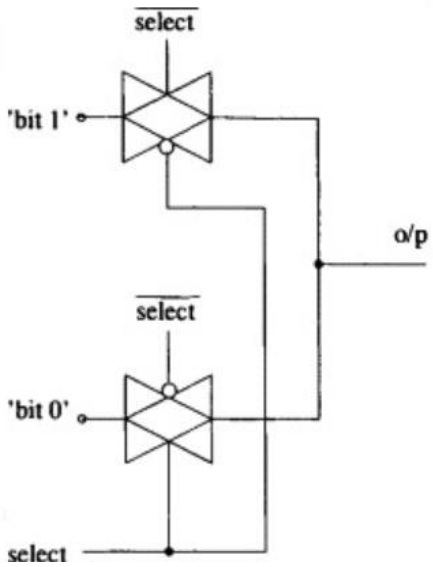


A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

Logical expression

$$Y = (A \cdot \sim B) + (\sim A \cdot B)$$

- **2-input AND using minimum number of Transmission Gates:**
  Efficient design for minimal area and power.



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Logical expression

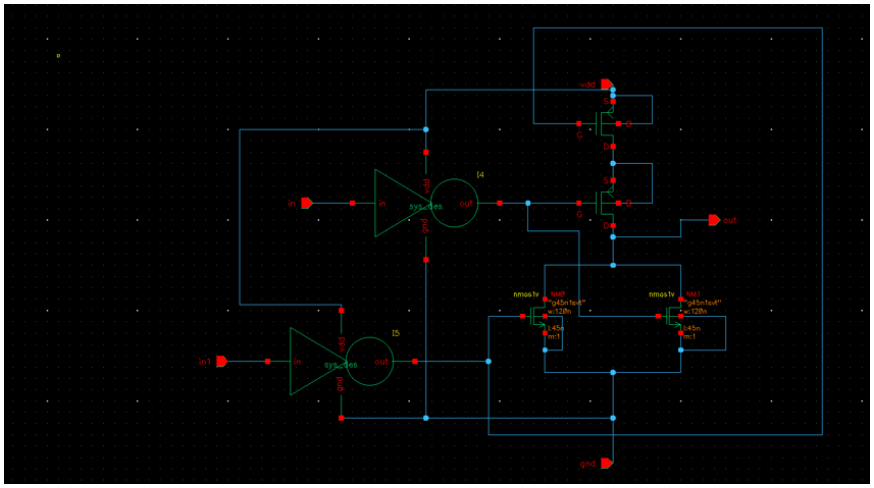
$$A \cdot B$$

CMOS Schematics

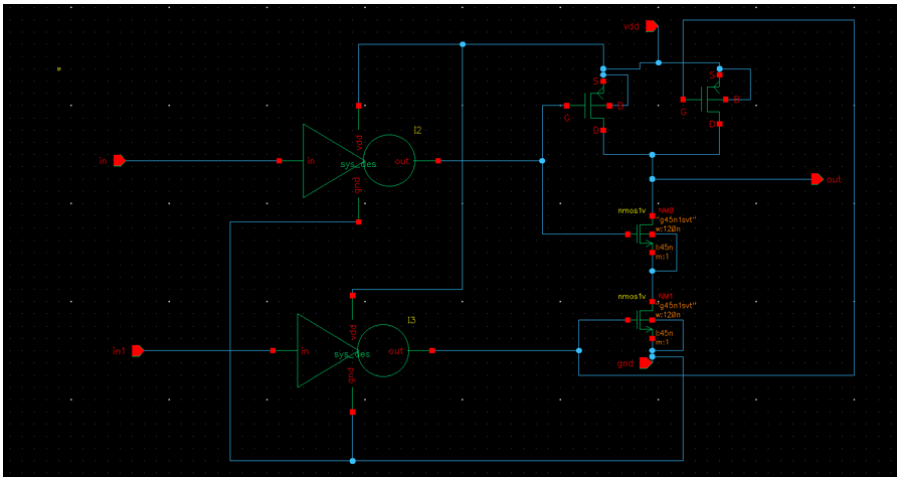
Design Specifications:

Supply Voltage	1V
Transistor Length	45 nm
Transistor Width	120 nm
Input Signal Frequency	Upto 1GHz

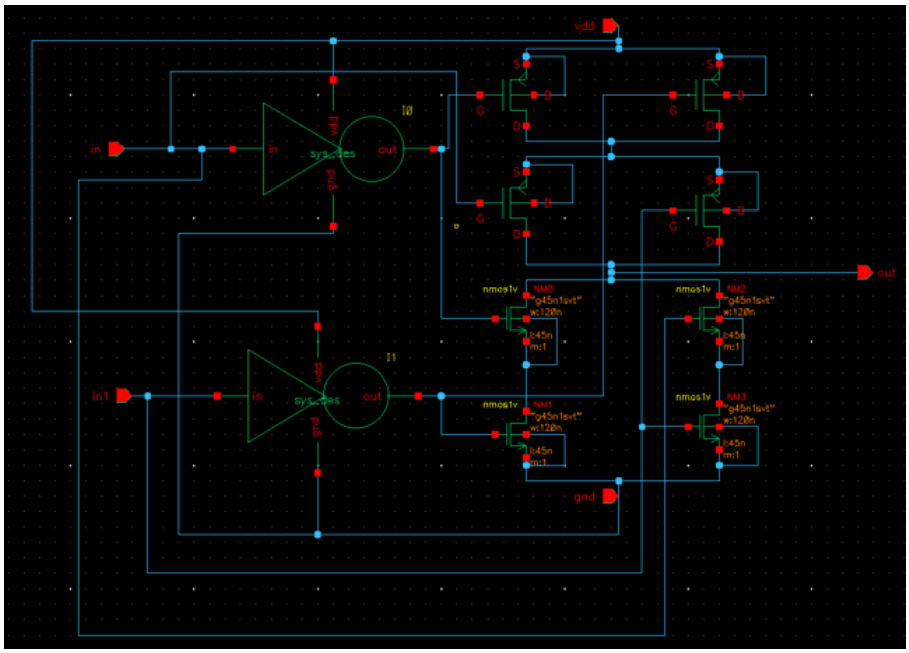
Schematic Screenshots from the tool (Each block of ALU + Complete ALU Design):



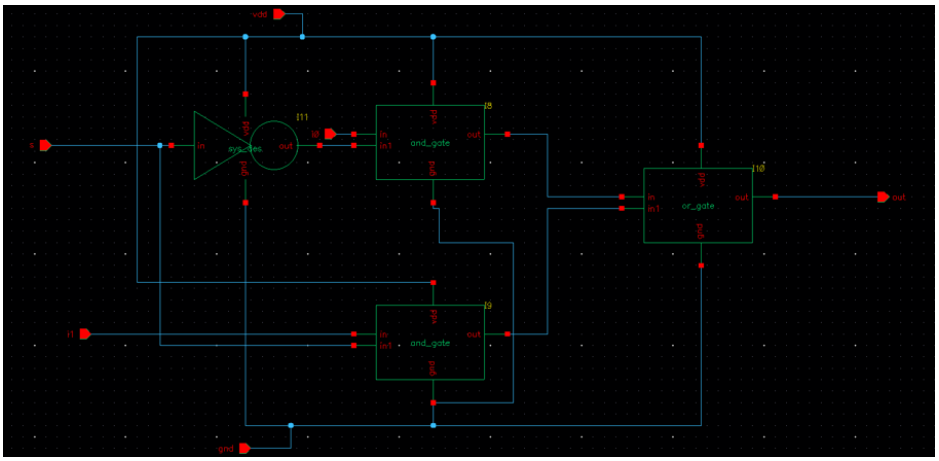
AND Gate



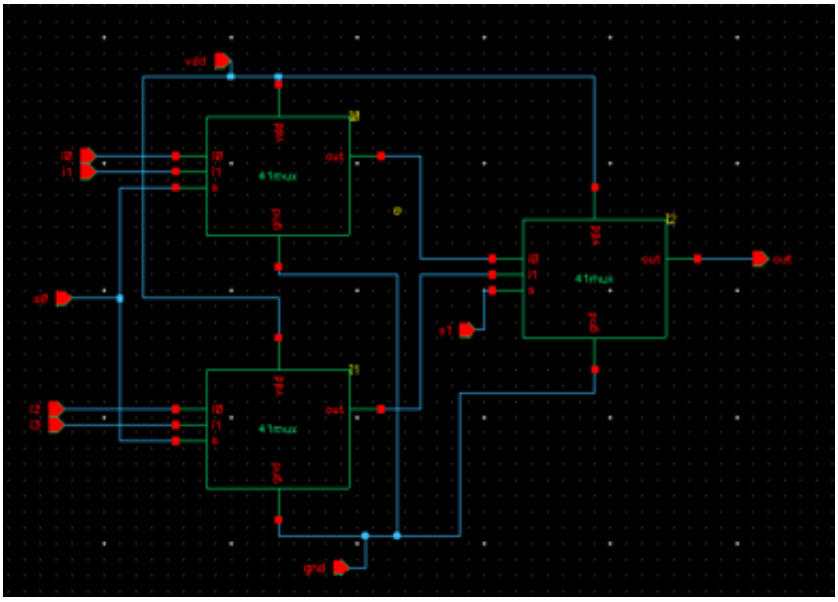
OR Gate



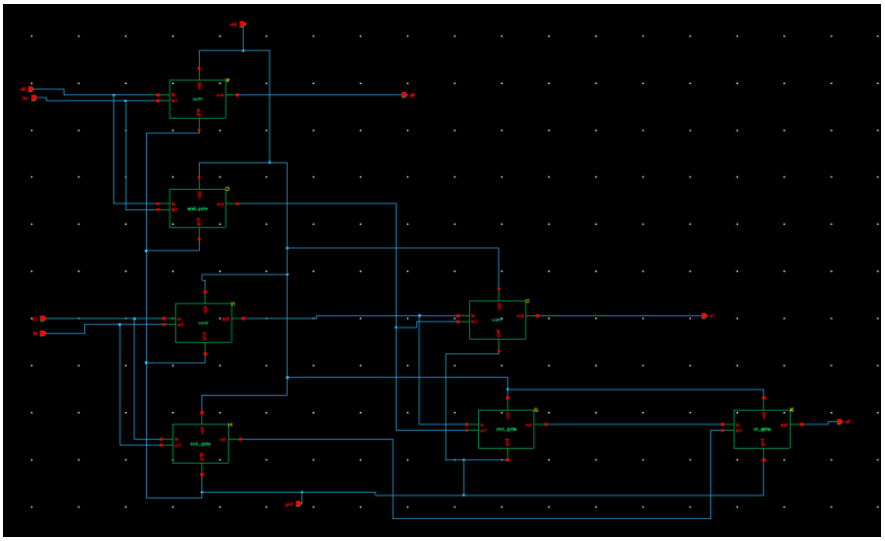
XOR Gate



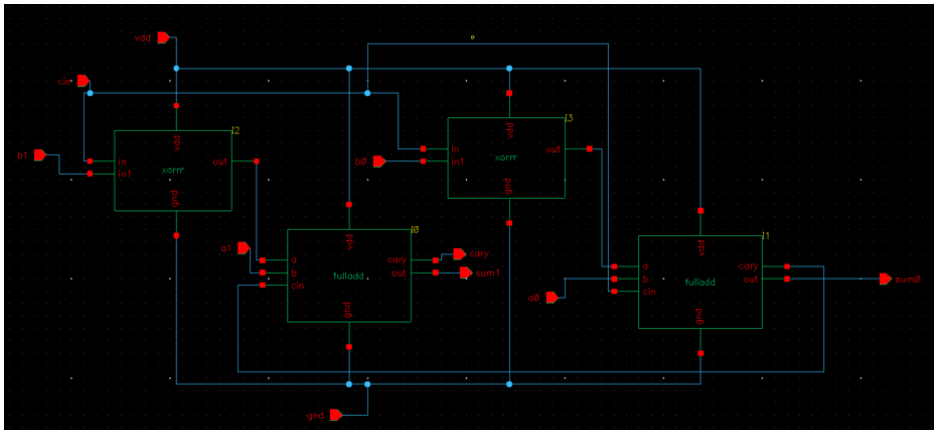
2:1 MUX



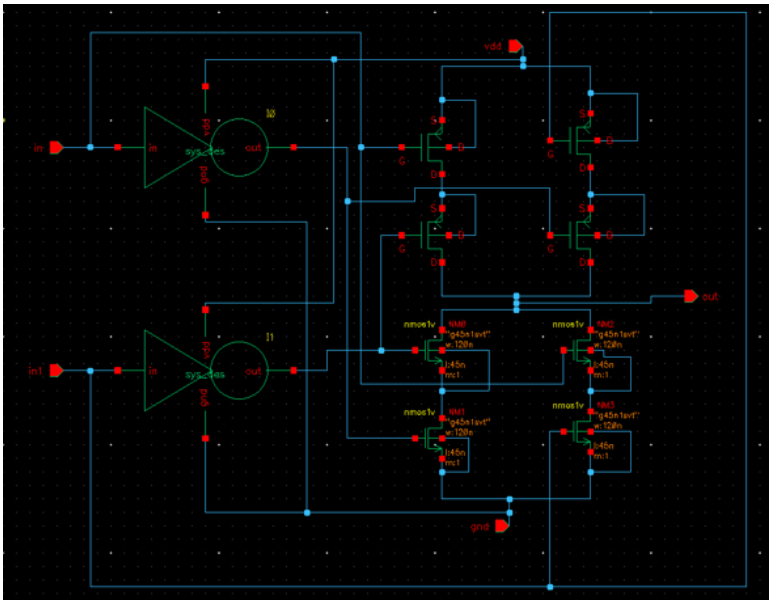
4:1MUX



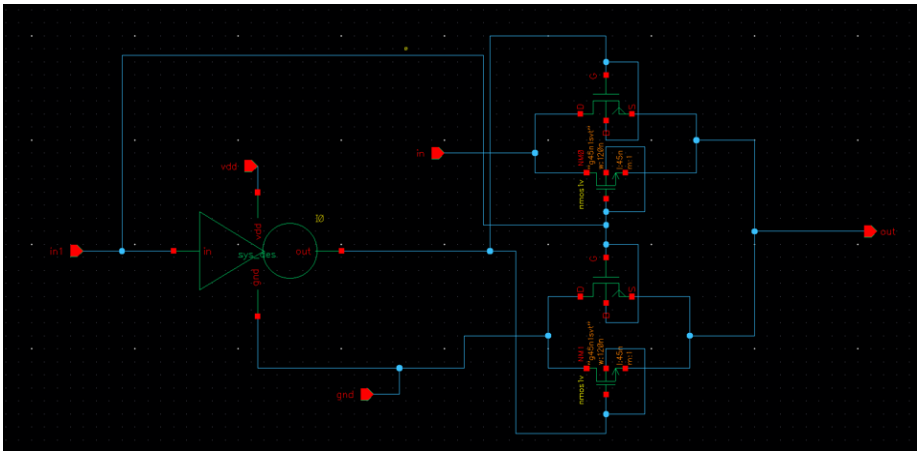
2 bit addition



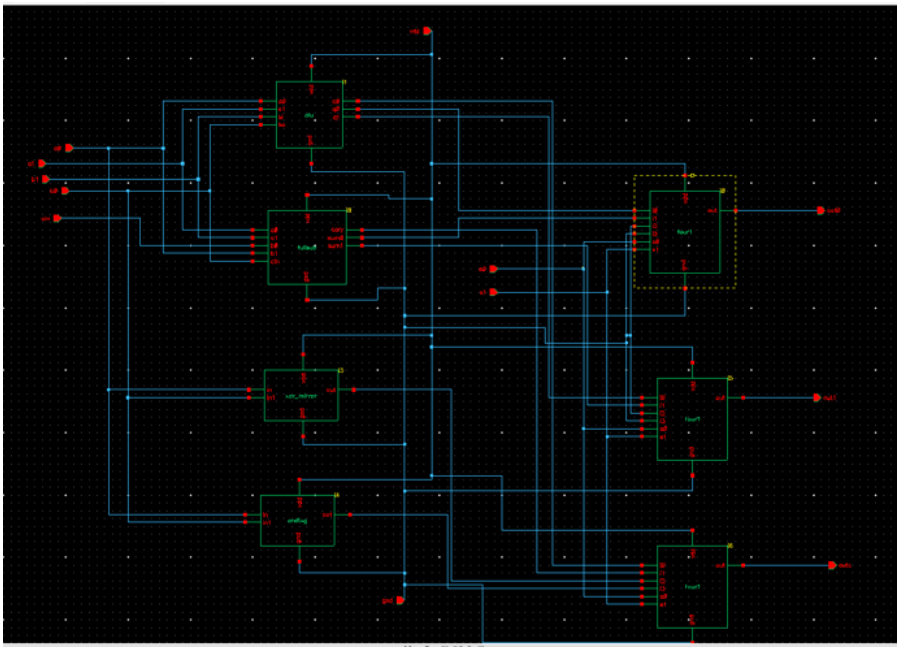
2 bit subtractor



2-input XOR using Mirror circuit

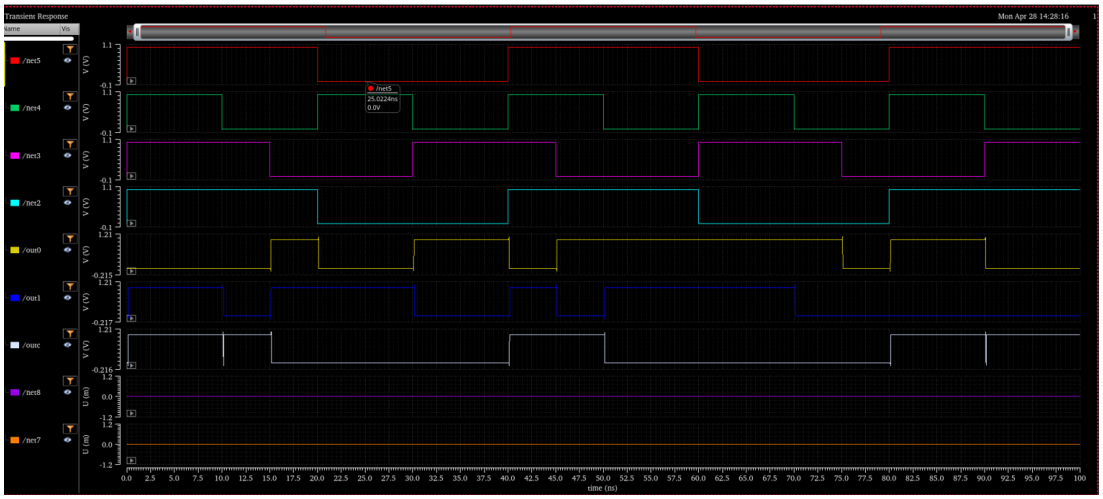


2-input AND using minimum number of Transmission gates

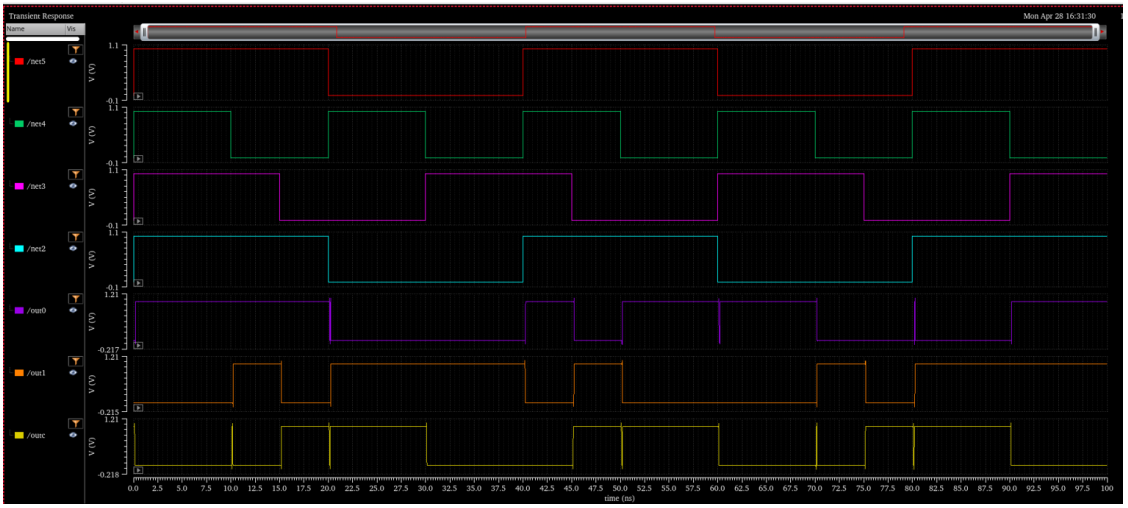


Complete ALU Design

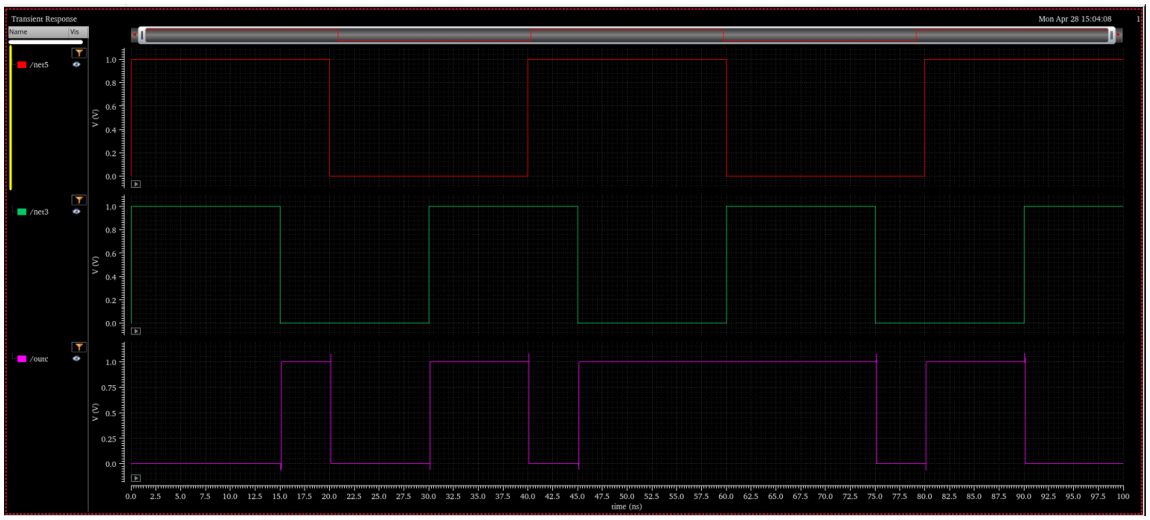
Simulation Results (Screenshot from the tool-Each block of ALU + Complete ALU Design):



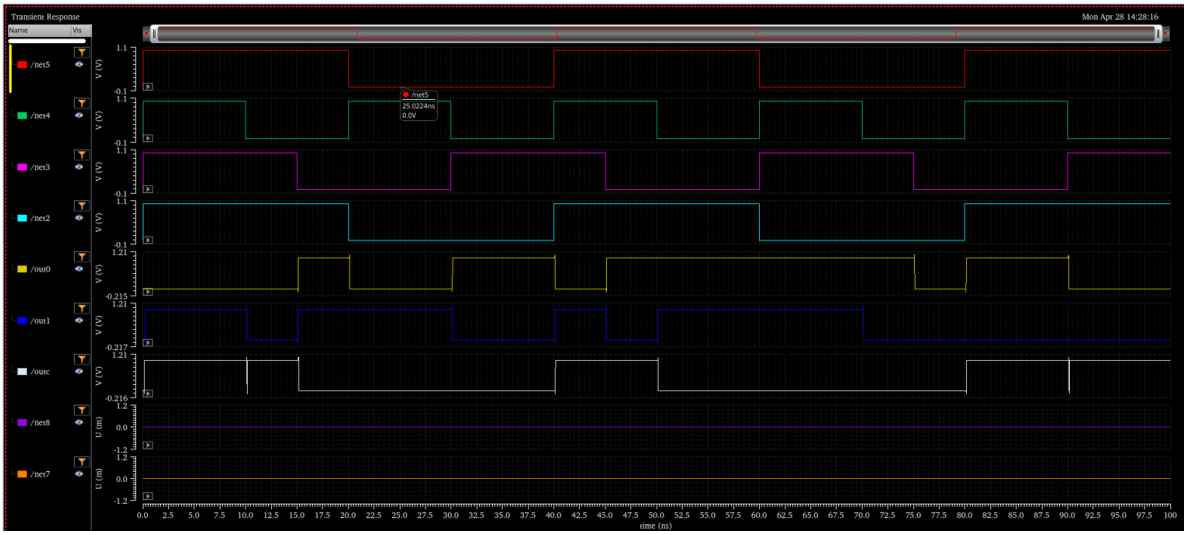
2-bit Addition



2-bit Subtraction using full adders and required logic gates



2-input XOR using Mirror circuit



2-input AND using minimum number of Transmission gates

Conclusion:

The ALU was successfully designed and implemented using Cadence Virtuoso at 45nm technology node. All operations (Addition, Subtraction, XOR, AND) were verified through schematic design and simulation. The use of mirror circuits and transmission gates resulted in optimized performance in terms of area and power.

Aim 2:
To design and implement a 4-bit ALU capable of performing :

- 4-bit Addition
- 4-bit Subtraction using full adders and required logic gates
- 4-bit Left shift SISO shift register
- 4-bit Ring Counter

Tool Used: Xilinx Vivado 2024.2

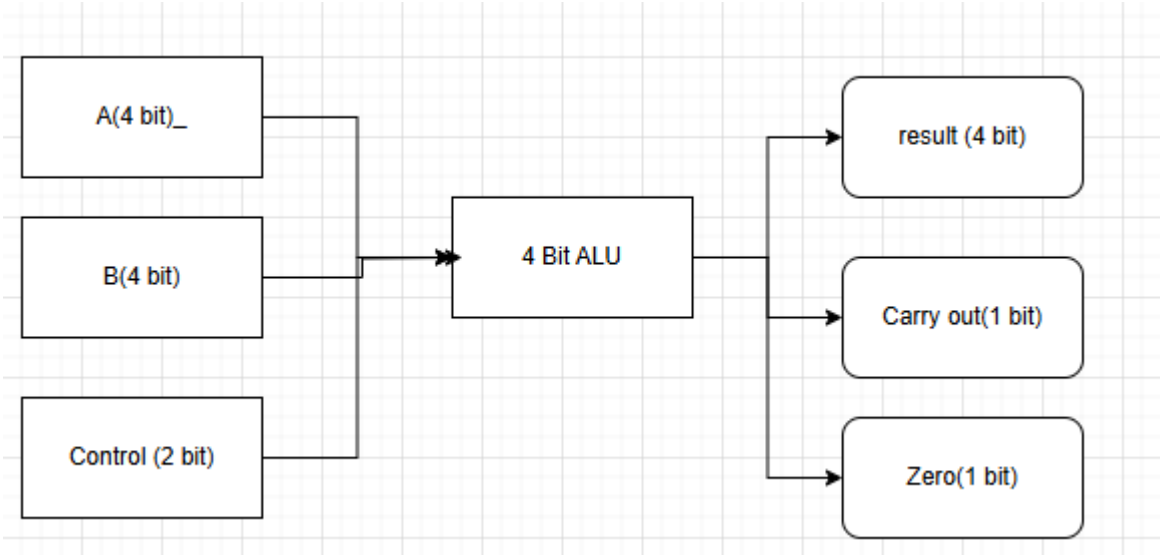
FPGA Family (for Synthesis): Virtex-7

Theory:

1) Introduction

   This Verilog module implements a simple 4-bit Arithmetic Logic Unit (ALU) that performs fundamental arithmetic and logic operations based on a 2-bit control signal. Operating synchronously with a clock input (clk), it accepts two 4-bit inputs (A and B) and outputs a 4-bit result along with a carry-out flag and a zero flag. The ALU supports four operations: addition (control = 00), subtraction (control = 01), a serial-in serial-out (SISO) left shift (control = 10), and a ring counter operation (control = 11). In addition and subtraction modes, it calculates the result and sets the carry and zero flags accordingly. In shift mode, it loads and shifts A left over successive clock cycles. The ring counter rotates the bits of A left by one position, effectively forming a circular shift. This ALU is suitable for basic computational tasks in digital systems and can be integrated into larger sequential logic designs.

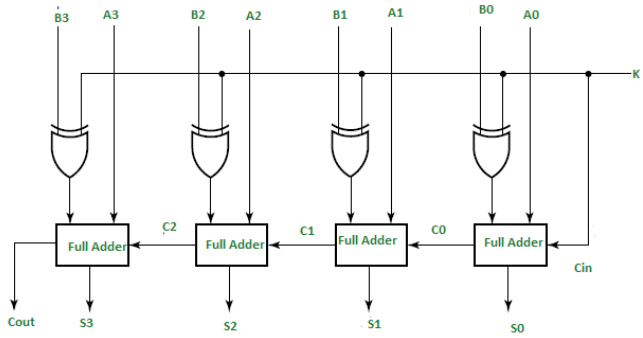
2) Block Diagram of designed ALU with working principle



The 4-bit ALU operates synchronously with the clock signal and performs one of four operations based on a 2-bit control input. When the control is `00`, it adds inputs `A` and `B`, outputs the result, and sets the carry and zero flags accordingly. For `01`, it subtracts `B` from `A`, with the result, carry (as borrow), and zero flag updated. When the control is `10`, the ALU performs a Serial-In Serial-Out (SISO) left shift on input `A`, storing it in a shift register that shifts one bit left on each clock cycle; the shifted-out bit becomes the carry-out, and the zero flag reflects the current output. If the control is `11`, a ring counter operation is executed by rotating the bits of `A` left, with no carry involved. In all cases, the ALU continuously updates the `result`, `carry_out`, and `zero` outputs on each positive clock edge based on the selected operation.

3) Logic Circuit for each block of the ALU with working principle (including Truth Table and Logic expressions)

- **4-bit Addition using full adders:**



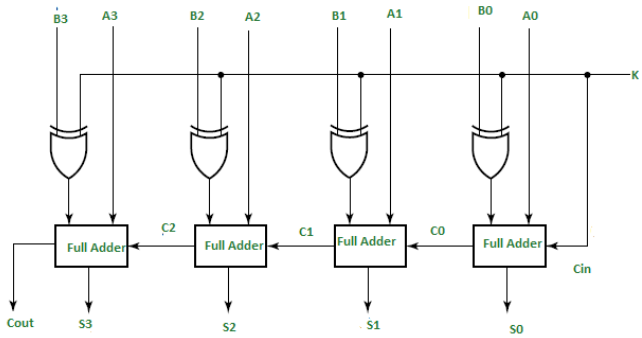
	A				B				Sum				Carry
Cin	A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0	Cout
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

If Cin=0,it performs addition

Logical expressions

- **S = (A ⊕ B) ⊕ C_in**: (XOR operation)
- **C_out = (A . B) + (A . C_in) + (B . C_in)**: (AND and OR operations)

- **4-bit subtraction using full adders**

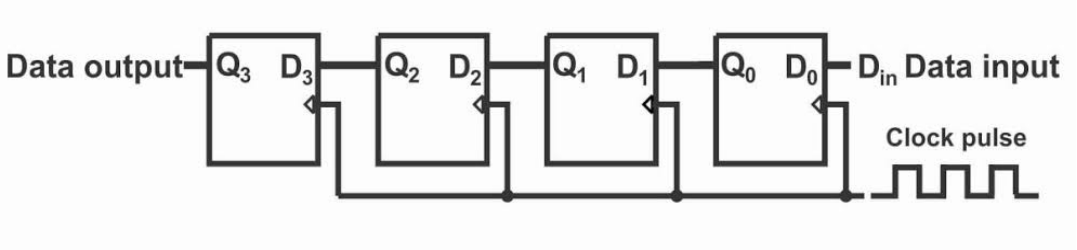


If Cin=1,it performs subtraction

Logical expressions

- $S_i = A_i \oplus B_i'' \oplus C_{i-1}$
- $C_i = (A_i \& B_i'') | (C_{i-1} \& (A_i \oplus B_i''))$

- **4-bit left shift using SISO shift register**



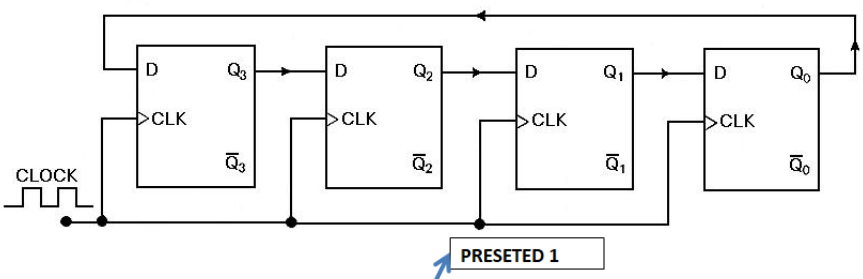
CLK	A0	A1	A2	A3
1st	1	0	1	0
2nd	0	1	0	0
3rd	1	0	0	0
4th	0	0	0	0

Logical expressions

- $D_i = Q_{(i-1) \bmod N}$, for $i = 0, 1, 2, \ldots, N-1$

This gives the generalized form of the left shift for any N-bit SISO shift register.

- **4-bit Ring counter**



PRESETED 1

ORI	CLK	Q0	Q1	Q2	Q3
low	X	1	0	0	0
high	low	0	1	0	0
high	low	0	0	1	0
high	low	0	0	0	1
high	low	1	0	0	0

Logical expressions

- $D_i = Q_{(i-1) \bmod N}$, for $i = 0, 1, 2, \ldots, N-1$

Where $(i-1) \bmod N$ represents the previous flip-flop in a cyclic manner.

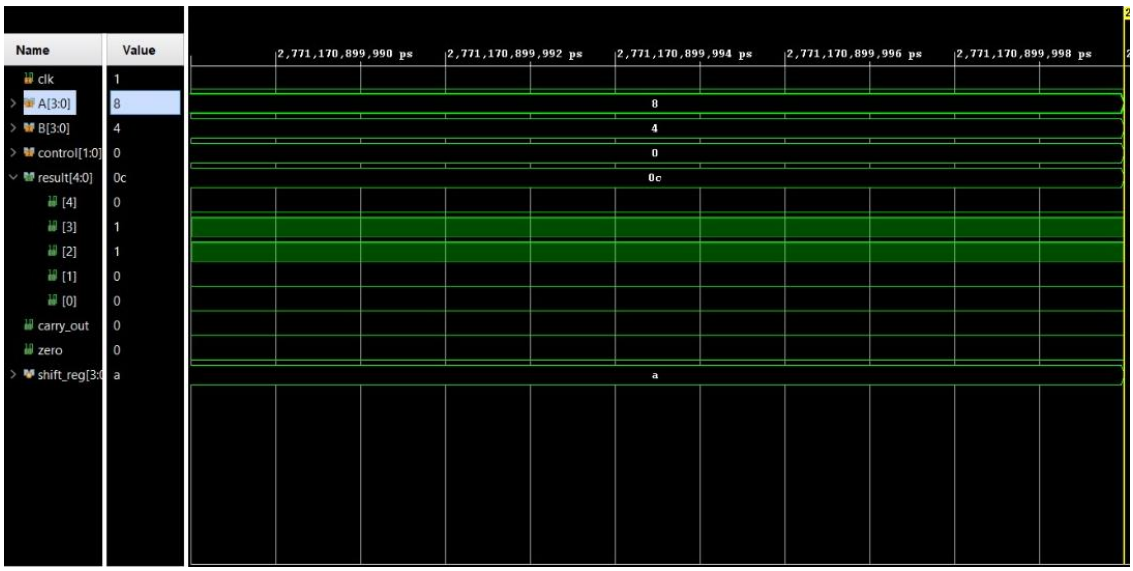
Verilog Code for Complete ALU Design and Sub-blocks of ALU:

```
module ALU (
    input clk,              // Clock signal
    input [3:0] A,          // 4-bit input A
    input [3:0] B,          // 4-bit input B
    input [1:0] control,    // 2-bit control signal to select operation
    output reg [3:0] result,  // 4-bit result of operation
    output reg carry_out,     // Carry out for addition
    output reg zero           // Zero flag
);
    reg [3:0] shift_reg;      // Register for SISO left shift

    always @(posedge clk) begin
        case (control)
            2'b00: begin // ADDITION
                {carry_out, result} = A + B;
                zero = (result == 4'b0000); // Zero flag
            end

            2'b01: begin // SUBTRACTION
                {carry_out, result} = A - B;
                zero = (result == 4'b0000); // Zero flag
            end

            2'b10: begin // SISO LEFT SHIFT
                if (shift_reg == 4'b0000)
                    shift_reg = A; // Initialize the shift register with input A
                else
                    shift_reg = {shift_reg[2:0], 1'b0}; // Shift left, shift in 0

                result = shift_reg;        // Update result with shifted value
```

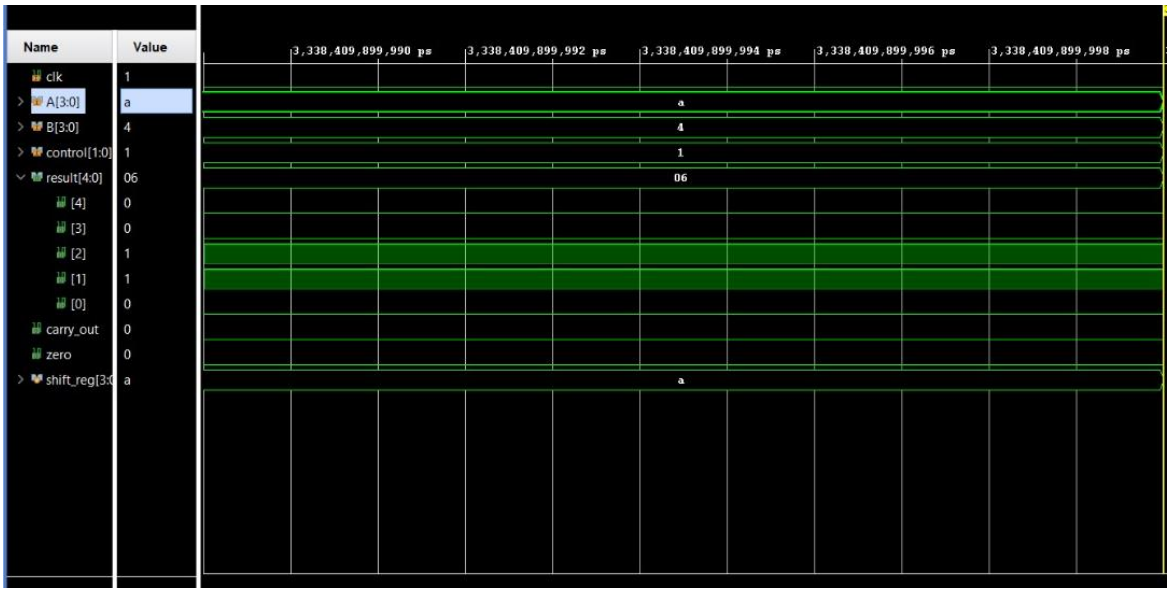
```
            carry_out = shift_reg[3]; // MSB shifted out
            zero = (result == 4'b0000); // Zero flag
        end

        2'b11: begin // RING COUNTER
            result = {A[2:0], A[3]}; // Rotate bits to form a ring counter
            carry_out = 0;          // No carry in ring counter
            zero = (result == 4'b0000); // Zero flag
        end
        default: begin
            result = 4'b0000;
            carry_out = 0;
            zero = 1;
        end
      endcase
   end
endmodule
```

Simulation Results (Screenshots from the tool):

4-bit Addition(8+4)

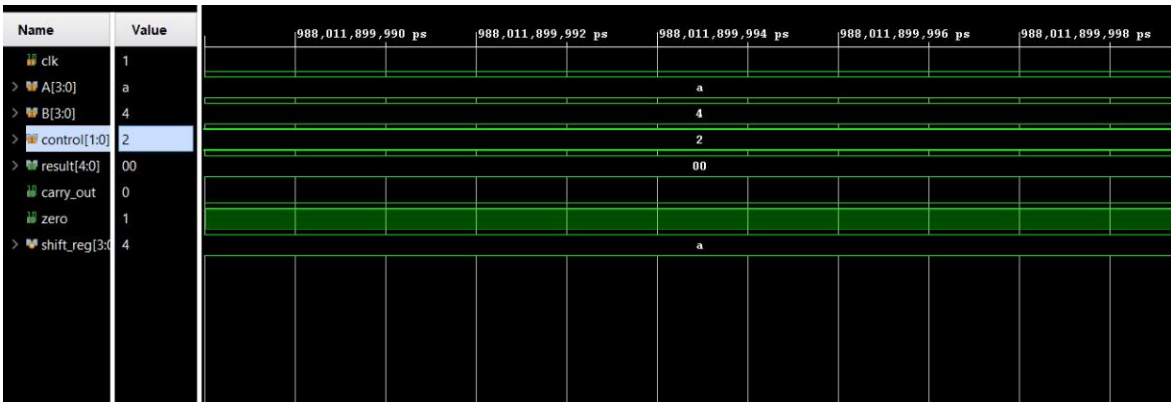


4-bitSubraction(10-4)

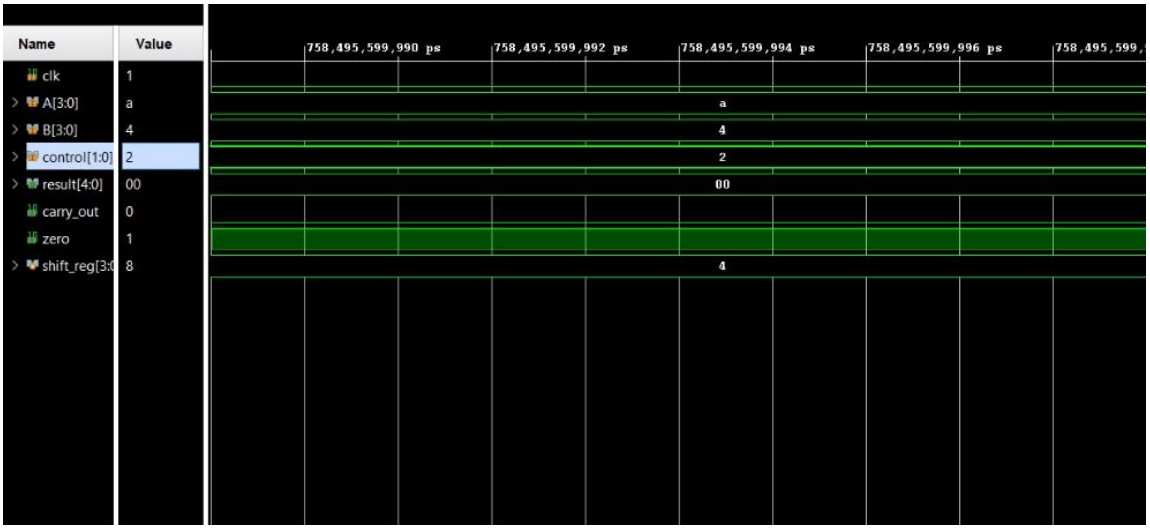


4-bit Left Shift SISO shift register(A=10)

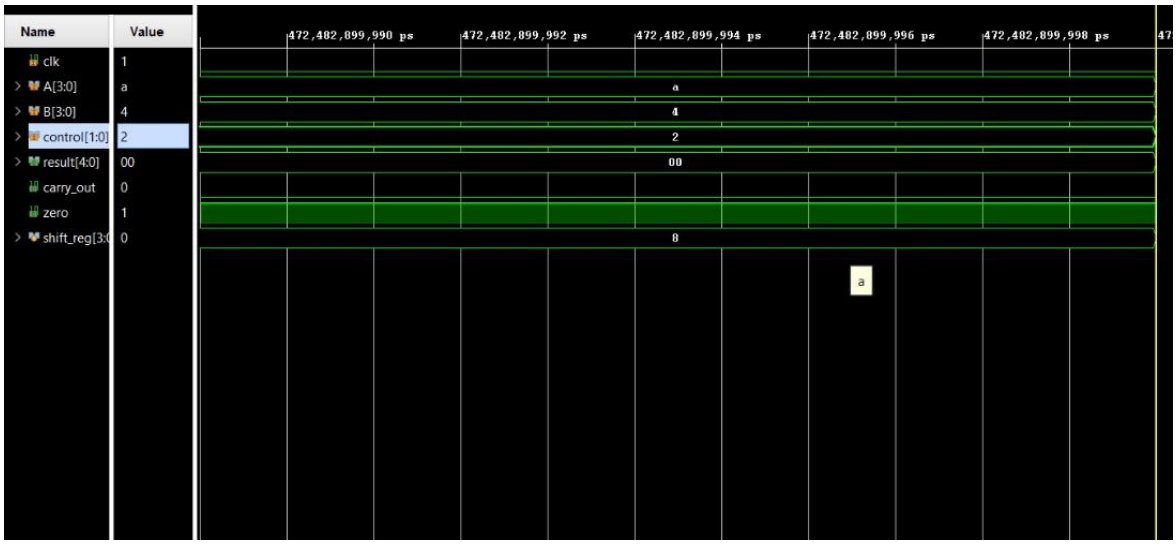
1st clock:



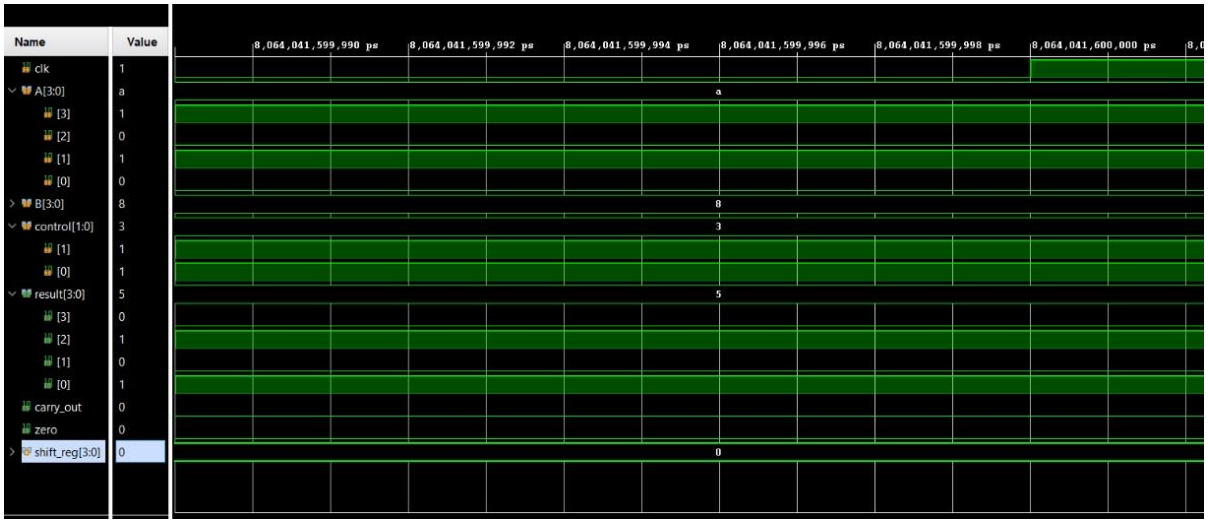
2nd clock:



3rd clock:



4-bit Ring Counter(A=10):

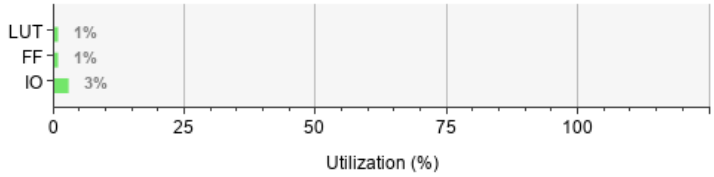


Simulation Results

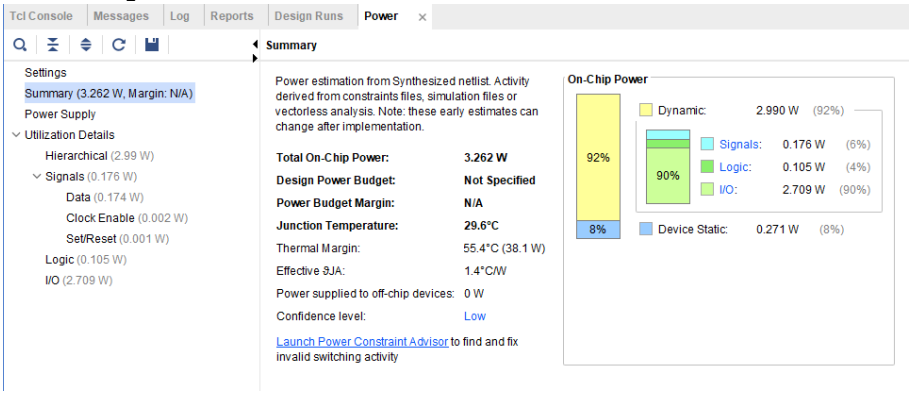
- **Area (Utilization) report:**

Summary

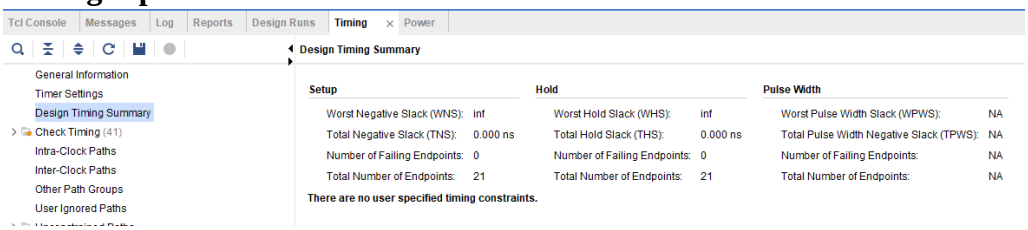
Resource	Utilization	Available	Utilization %
LUT	24	303600	0.01
FF	10	607200	0.00
IO	17	600	2.83



- **Power report:**



- **Timing report:**



Conclusion:

The 4-bit ALU was successfully designed and implemented using Verilog to perform addition, subtraction, left shift (SISO), and ring counter operations based on a 2-bit control signal. The design correctly handled carry-out and zero detection, with all operations synchronized to the clock edge. Simulation results verified the correct functionality of each operation. The ALU design is efficient in terms of speed, area, and power, making it well-suited for FPGA implementation. Overall, the ALU meets the required functional and performance specifications effectively.