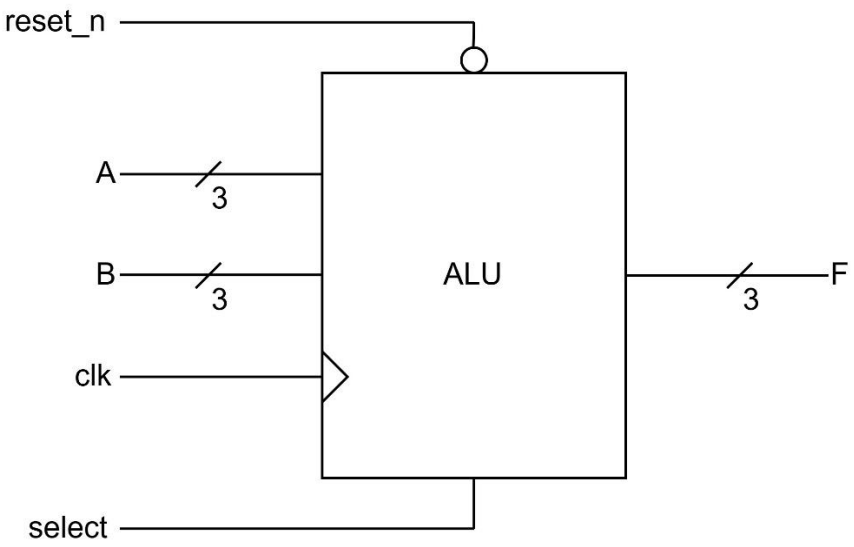
PROJECT OVERVIEW:

- Designed a custom 3-bit Arithmetic Logic Unit (ALU) using a 7nm technology node.
- The ALU accepts two 3-bit operands (A and B), an active-low reset signal (reset_n), and a single-bit select line (select) that determines the operation mode.
- When select is 0, the ALU performs 3-bit unsigned addition; when select is 1, it performs 3-bit subtraction using two's complement arithmetic.
- The 3-bit result is provided on output F.
- A total of 34 standard cells were integrated into the final layout.
- The design was successfully verified both functionally and post-layout using HSPICE simulations.

BLOCK DIAGRAM:



HDL/RTL CODE:

The HDL code was written in Verilog as below and verified using functional simulation.

```
`timescale 1ns / 1ps

module ALU(
    input wire clk,
    input wire [2:0] A,
    input wire [2:0] B,
    input wire reset_n,
    input wire select,
```

```
    output wire [2:0] F
    );
reg [2:0] F_out;
reg [2:0] ALU_out;

always @ (posedge clk)begin
    if (~reset_n)begin
        F_out <= 4'd0;
    end else begin
       F_out <= ALU_out;
    end
end

always @ (*) begin
    case(select)
        1'd0:ALU_out = A+B;                 // Addition
        1'd1:ALU_out = A+(~B)+3'b001;       // Subtraction(2's complement addition)
    endcase
end

assign F = F_out;
endmodule
```

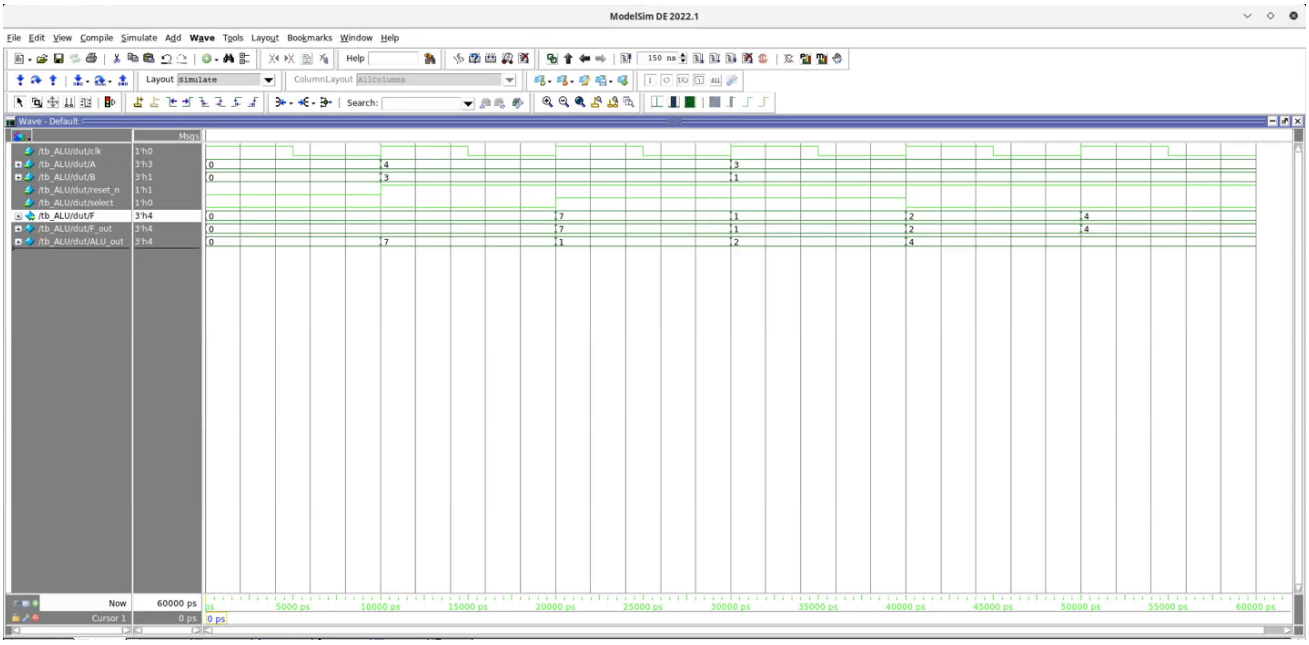
TESTBENCH:

```
`timescale 1ns / 1ps

module tb_ALU;
reg [2:0] A;
reg [2:0] B;
reg select;
reg clk;
reg reset_n;
wire [2:0] F;

always #5 clk = ~clk;

ALU dut (
    .A(A),
    .B(B),
    .select(select),
    .clk(clk),
    .reset_n(reset_n),
    .F(F)
);

initial begin
    clk = 1;
    A = 3'd0;
    B = 3'd0;
    reset_n = 0;
    select = 1'd0;
```

```
    #10;
    reset_n = 1;
    A = 3'd4;
    B = 3'd3;
    #10;
    select =1'd1;
    #10;
    A = 3'd3;
    B = 3'd1;
    #10;
    select = 1'd0;
    #20;
    $finish;
end

endmodule
```

HDL/RTL SIMULATION WAVEFORM:

The simulated waveform below confirmed the correct functionality of the ALU with accurate results for both operations based on the select line.



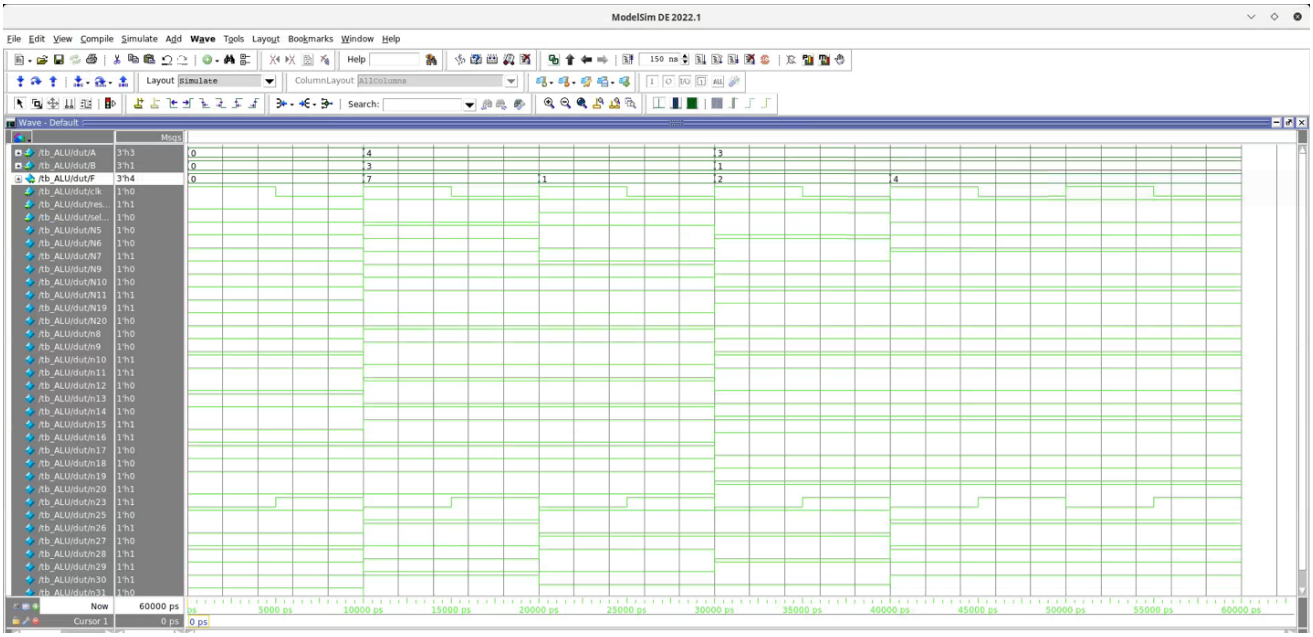
SYNTHESIZED NETLIST:

The Synthesized netlist below was also verified using simulation.

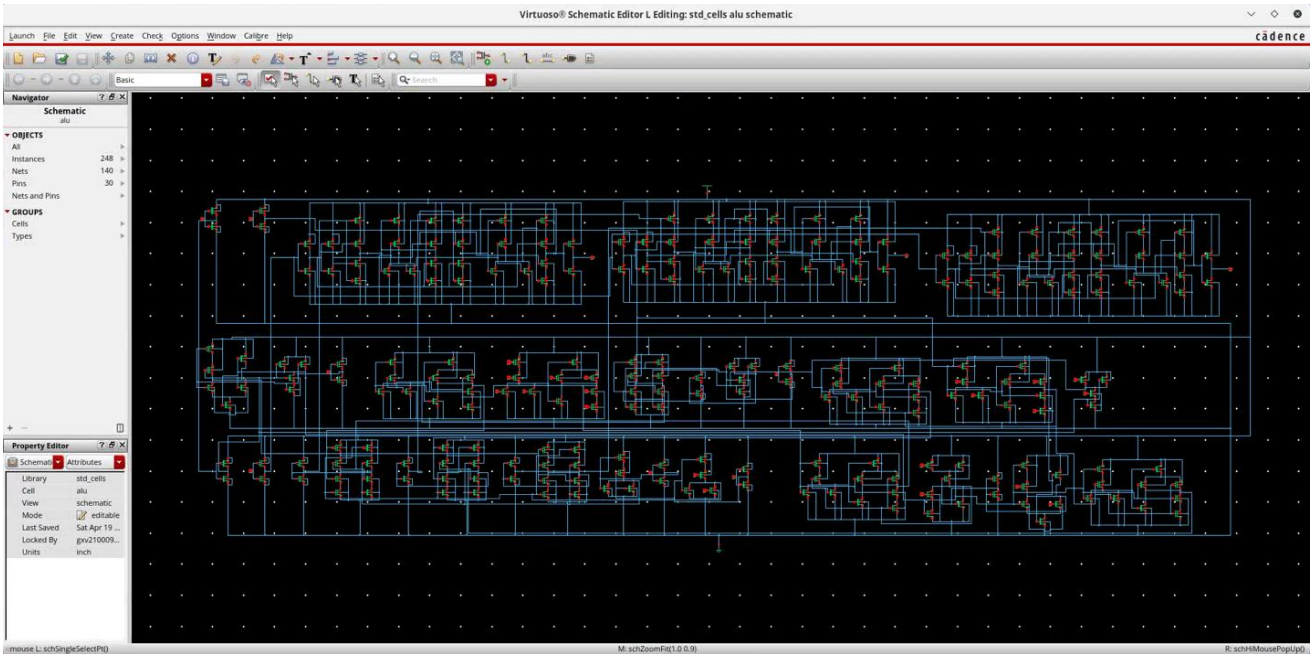
```
module INV(IN, OUT);
input IN;
output OUT;
assign OUT = ~IN;
endmodule

module NAND2(A, B, OUT);
input A, B;
output OUT;
assign OUT = ~(A & B);
endmodule

module NOR2(A, B, OUT);
input A, B;
output OUT;
assign OUT = ~(A | B);
endmodule

module XOR2(A, B, OUT);
input A, B;
output OUT;
assign OUT = (A ^ B);
endmodule
module AOI21(A, B, C, OUT);
input A, B, C;
output OUT;
assign OUT = ~((A & B) | C);
endmodule

module AOI22(A, B, C, D, OUT);
input A, B, C, D;
output OUT;
assign OUT = ~((A & B) | (C & D));
endmodule

module OAI21(A, B, C, OUT);
input A, B, C;
output OUT;
assign OUT = ~((A | B) & C);
endmodule

module OAI22(A, B, C, D, OUT);
input A, B, C, D;
output OUT;
assign OUT = ~((A | B) & (C | D));
endmodule

module DFF( D, CLK, R, Q);
input D, CLK, R;
output Q;
reg Q;
always @(negedge CLK or posedge R)
```

```
  if (R == 1'b1)
    Q = 1'b0;
  else
    Q = D;
endmodule

module ALU ( clk, A, B, reset_n, select, F );
  input [2:0] A;
  input [2:0] B;
  output [2:0] F;
  input clk, reset_n, select;
  wire   N5, N6, N7, N9, N10, N11, N19, N20, n8, n9, n10, n11, n12, n13, n14,
        n15, n16, n17, n18, n19, n20, n23, n25, n26, n27, n28, n29, n30, n31;
  wire   [2:0] \sub_1_root_add_45_2/B_not ;
  INV \sub_1_root_add_45_2/U1_0  ( .IN(B[0]), .OUT(\sub_1_root_add_45_2/B_not [0]) );
  INV \sub_1_root_add_45_2/U1_1  ( .IN(B[1]), .OUT(\sub_1_root_add_45_2/B_not [1]) );
  DFF \F_out_reg[2]  ( .D(N7), .CLK(n23), .R(1'b0), .Q(F[2]) );
  DFF \F_out_reg[1]  ( .D(N6), .CLK(n23), .R(1'b0), .Q(F[1]) );
  DFF \F_out_reg[0]  ( .D(N5), .CLK(n23), .R(1'b0), .Q(F[0]) );
  XOR2 U3 ( .A(n8), .B(n9), .OUT(N20) );
  AOI21 U4 ( .A(A[1]), .B(n10), .C(n11), .OUT(n9) );
  INV U5 ( .IN(n12), .OUT(n11) );
  OAI21 U6 ( .A(n10), .B(A[1]), .C(\sub_1_root_add_45_2/B_not [1]), .OUT(n12));
  XOR2 U7 ( .A(n10), .B(n13), .OUT(N19) );
  OAI21 U8 ( .A(n14), .B(n15), .C(n10), .OUT(N9) );
  NAND2 U9 ( .A(n15), .B(n14), .OUT(n10) );
  INV U10 ( .IN(A[0]), .OUT(n14) );
  XOR2 U11 ( .A(n16), .B(n8), .OUT(N11) );
  XOR2 U12 ( .A(B[2]), .B(A[2]), .OUT(n8) );
  OAI22 U13 ( .A(n17), .B(\sub_1_root_add_45_2/B_not [1]), .C(n18), .D(n19), .OUT(n16) );
  INV U14 ( .IN(n20), .OUT(n17) );
  NAND2 U15 ( .A(n19), .B(n18), .OUT(n20) );
  INV U16 ( .IN(A[1]), .OUT(n19) );
  XOR2 U17 ( .A(n18), .B(n13), .OUT(N10) );
  XOR2 U18 ( .A(\sub_1_root_add_45_2/B_not [1]), .B(A[1]), .OUT(n13) );
  NAND2 U19 ( .A(A[0]), .B(n15), .OUT(n18) );
  INV U20 ( .IN(\sub_1_root_add_45_2/B_not [0]), .OUT(n15) );
  INV U23 ( .IN(clk), .OUT(n23) );
  INV U25 ( .IN(n25), .OUT(N7) );
  AOI22 U26 ( .A(N11), .B(n26), .C(N20), .D(n27), .OUT(n25) );
  INV U27 ( .IN(n28), .OUT(N6) );
  AOI22 U28 ( .A(N10), .B(n26), .C(N19), .D(n27), .OUT(n28) );
  INV U29 ( .IN(n29), .OUT(N5) );
  AOI22 U30 ( .A(N9), .B(n26), .C(N9), .D(n27), .OUT(n29) );
  NOR2 U31 ( .A(n30), .B(n31), .OUT(n27) );
  INV U32 ( .IN(select), .OUT(n30) );
  NOR2 U33 ( .A(n31), .B(select), .OUT(n26) );
  INV U34 ( .IN(reset_n), .OUT(n31) );
endmodule
```

NETLIST SIMULATION WAVEFORM:

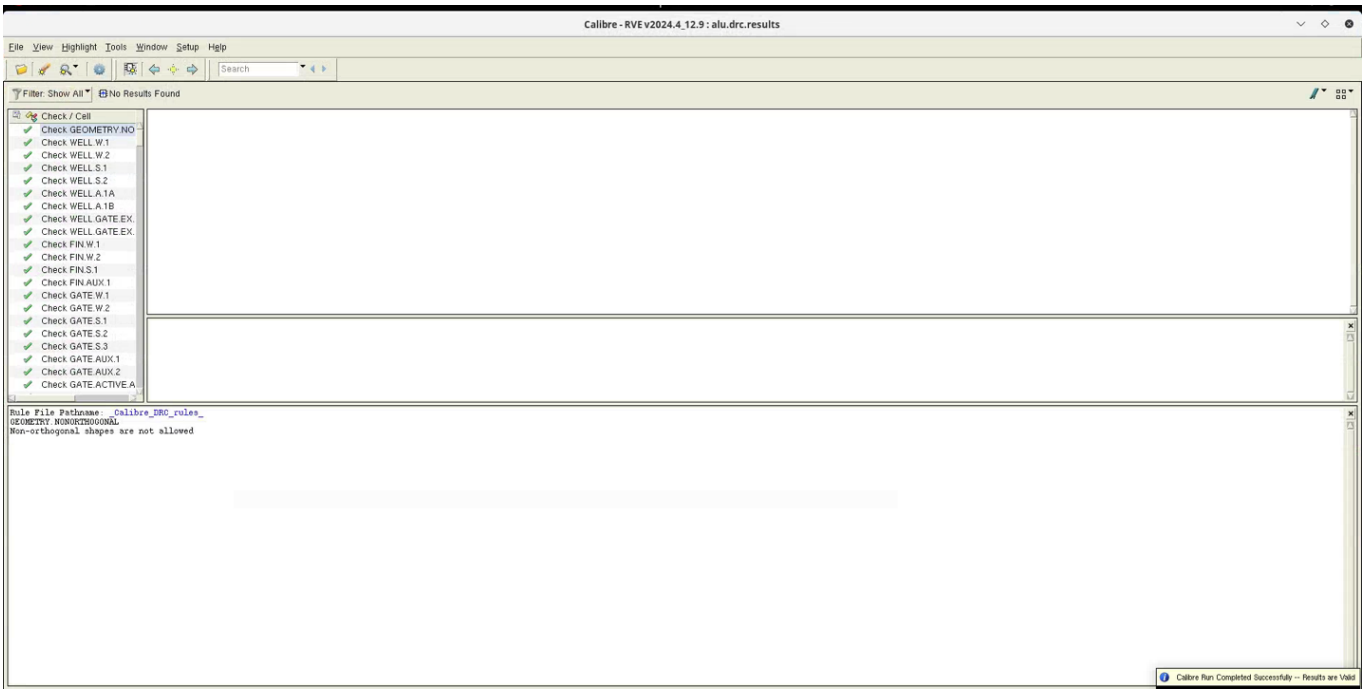


SCHEMATIC:

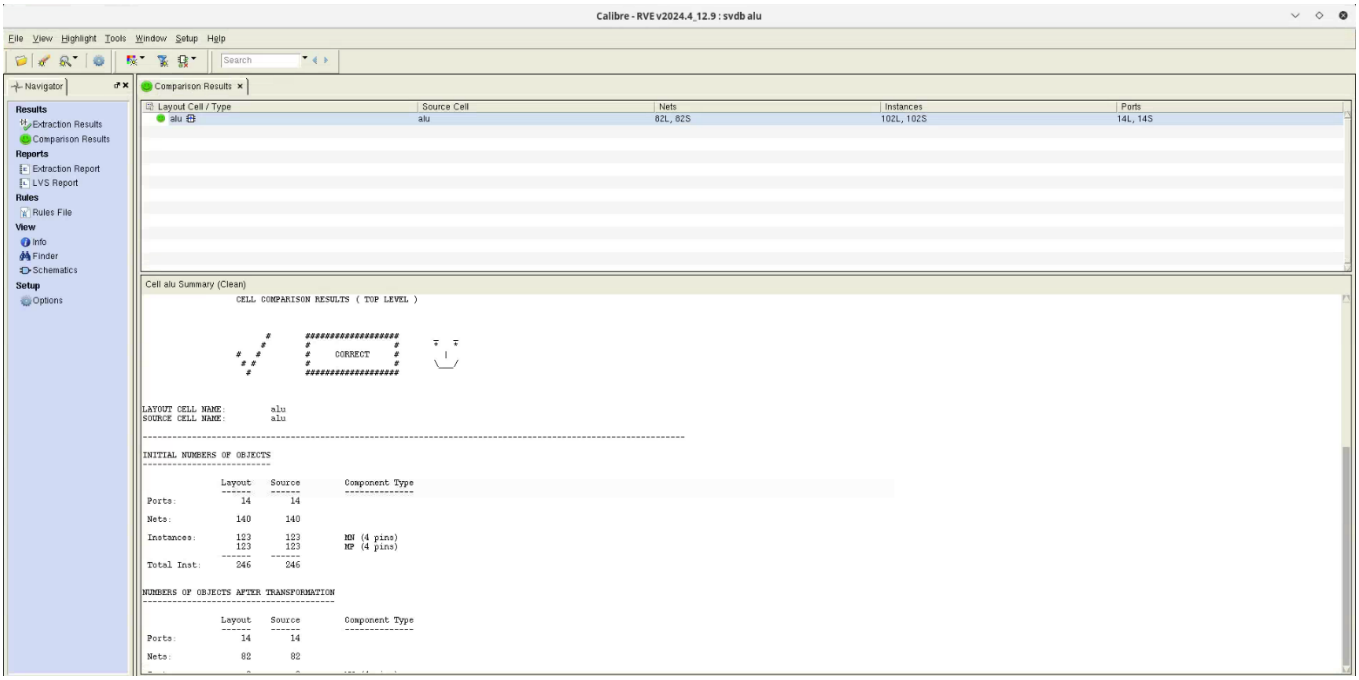


LAYOUT:

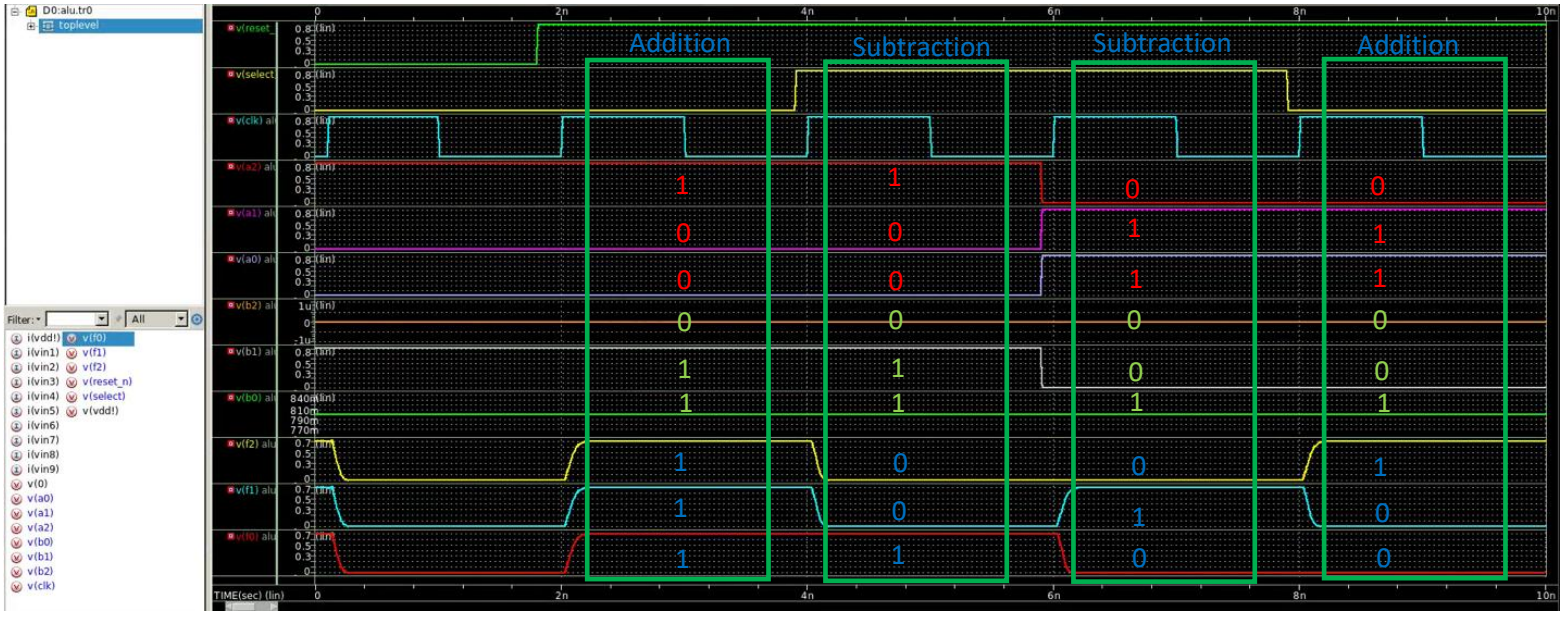
DRC Check:



LVS Check:



HSPICE SIMULATION:



HSPICE FILE:

```
.include "~/asap7/7nm_TT.pm"
.include alu.pex.netlist
.option post runlvl=5
xi A0 A1 A2 B0 B1 B2 F0 F1 F2 CLK RESET_N SELECT alu
VDD! VDD! GND! 0.7v
vin1 CLK GND! pwl(0ps 0v 100ps 0v 114ps 0.8v 1000ps 0.8v 1014ps 0v 2000ps 0v 2014ps 0.8v
3000ps 0.8v 3014ps 0v 4000ps 0v 4014ps 0.8v 5000ps 0.8v 5014ps 0v 6000ps 0v 6014ps 0.8v
7000ps 0.8v 7014ps 0v 8000ps 0v 8014ps 0.8v 9000ps 0.8v 9014ps 0v)
vin2 RESET_N GND! pwl(0ps 0v 1800ps 0v 1814ps 0.8v)
vin3 A2 GND! pwl(0ps 0.8v 5900ps 0.8v 5914ps 0v)
vin4 A1 GND! pwl(0ps 0v 5900ps 0v 5914ps 0.8v)
vin5 A0 GND! pwl(0ps 0v 5900ps 0v 5914ps 0.8v)
vin6 B2 GND! pwl(0ps 0v)
vin7 B1 GND! pwl(0ps 0.8v 5900ps 0.8v 5914ps 0v)
vin8 B0 GND! pwl(0ps 0.8v)
vin9 SELECT GND! pwl(0ps 0v 3900ps 0v 3914ps 0.8v 7900ps 0.8v 7914ps 0v)
cout1 F2 GND! 10fF
cout2 F1 GND! 10fF
cout3 F0 GND! 10fF
.tr 1ps 10000ps
.end
```

VERIFICATION:

- **DRC**: Passed (No violations)
- **LVS**: Passed (Schematic matches layout)
- **PEX Simulation**: Conducted using HSPICE.

CONCLUSION:

- The design and implementation of the 7nm ALU demonstrate a functional and verifiable digital circuit capable of performing addition and subtraction based on a select signal.
- The integration of 34 standard cells and compliance with all physical verification checks (DRC, LVS, and PEX) validate the robustness of the layout.
- Consistency between HDL and HSPICE simulation results confirms the correctness of the design.
- The project highlights key aspects of custom chip development, reinforces the physical design flow, and emphasizes the importance of verification at each step of the VLSI design process.