University of Texas, Dallas
Department of Electrical and Computer Engineering

CE/EECT 6325: VLSI Design

ARITHMETIC LOGIC UNIT OPERATIONS

Submitted by
Dhruvi Shah (DDS200004)
Navya Bandari (NXB210004)
Naga Mutya Kumar Kumtsam (NXK210028)

DESIGN SPECIFICATIONS

The project is to design and simulate arithmetic logic unit operations. The *input1* and *input2* are two 32-bits inputs. The 32-bits output *out* stores the result of the performed operation. The 1-bit output Cout stores the carry and borrow outputs for addition and subtraction operations. The 4-bit sel line decides the operation to be performed.

The following operations are designed in this ALU

1.Addition

2.Subtraction

3.Logical AND

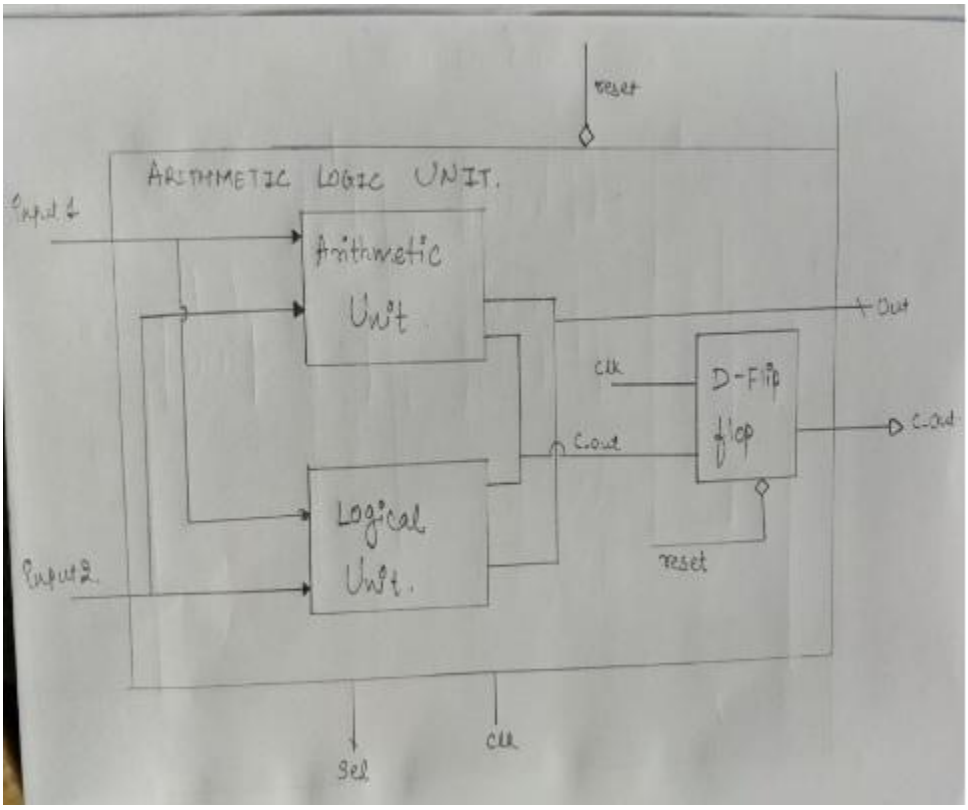
4. Logical OR

5. Division

6. Comparison and Equality

7.Logical XOR

BLOCK DIAGRAM



DESIGN AND TESTING PROCESS

- Initially in project1 we designed our block diagram and implemented it by writing a verilog code and verified the functionality of each operation with the help of derived value along with the waveforms.
- In the next step, we introduced to Design vision where we can generate the netlist of our verilog code by reading it with the help of the library provided.
- According to the specifications, our design must contain a minimum of 3000 cells. We modified our verilog code inorder to obtain the minimum number of cells. We can extract all the information such as number of cells reported, timing,area using Design Vision.
- Once we met the above requirements, we started designing the layout and schematic of Inverter using 65nm technology using the Cadence Virtuoso tool.
- After Designing the Inverter we verified Design Rule Check(DRC) , Layout vs Schematic (LVS) and verified the functionality of our inverter by using Hspice and Waveform Viewer.
- Once we are done with designing and verification of Inverter we repeated the above process for designing and verification of other gates which includes NAND2,NOR2,XOR,MUX,OAI21,AOI211,AOI22 and DFF.
- We also created abstract views of our gates with the help of Virtuoso. Once these are available we will create a .lef file.
- After completion of the above process we started creating individual libraries for each gate using SiliconSmart. Once all the libraries are created we combine all the lib files to make a single .lib file.
- With the help of a .lib file converted into a .db file using Lc shell which we used in the process of Design Vision will again generate our netlist file and verify the functionality of our verilog code.
- Once we have .lib,.lef,.db files we will move onto the next process which includes Cadence Innovus.Followed the entire process which is available in the tutorial for creating floor plan,power plan and routing. Once all these tasks are done we will generate a .def file and synthesized verilog file.
- In the next process we will be using Cadence Virtuoso again and import .def file,.lib file and generate a final design which comprises all the gates.
- With the help of Final Design we again perform DRC,LVS.
- In order to get the critical path and timing report of the final design we used Prime Tool.
- In the prime tool it required .db file,.v synthesized innovus file and changed the variables in the variables file according to our files and ran the prime script to obtain the timing report.

VERILOG CODE

```
module PROJECT_1_A (input1, input2, reset, clk, sel, out,
c_out);

input [31:0] input1, input2;

input reset;

input clk;

input[3:0] sel;

output[31:0] out;

output c_out;

reg[31:0] out;

reg c_out;

always@(posedge clk)

begin

//Implementing D-FlipFlop. If the reset signal is 1,c_out will
be reset

if(reset)

{out,c_out} = 33'b0;

else

begin

    case(sel)

      // Arithmetic Addition

                        4'b0000: begin {c_out,out} = input1 +
input2; end

      // Arithmetic Subtraction

                        4'b0001: begin {c_out,out} = input1 -
input2; end

      //Logical AND Operation. 'c_out' is not used so it is
initialised as 0

                        4'b0010: begin out = input1 & input2;
c_out = 1'b0; end

      //Logical OR Operation. 'c_out' is not used so it is
initialised as 0
```

```
                        4'b0011: begin out = input1 | input2;
c_out = 1'b0; end

      // Arithmetic Division

                        4'b0100: begin {c_out,out} = input1 /
input2; end

      // Comparison Operations where 'out' is not used to
initialised to 16'd0

                        4'b0101: begin c_out = (input1>input2) ?
1'b1:1'b0; out = 16'd0;  end

                        4'b0110: begin c_out = (input1<input2) ?
1'b1:1'b0; out = 16'd0;  end

                        4'b0111: begin c_out = (input1==input2) ?
1'b1:1'b0; out = 16'd0; end

      //Logical XOR Operation. 'c_out' is not used so it is
initialised as 0

                        default : begin out = input1 ^ input2;
c_out = 1'b0; end

    endcase

end

end

endmodule
```

VERILOG TEST BENCH

```
module PROJECT_1_TB;
reg[31:0] input1,input2;
reg clk,reset;
reg[3:0] sel;
wire[31:0] out;
wire c_out;

integer i;

  PROJECT_1_A DUT(input1,input2,reset,clk,sel,out,c_out);

  initial begin
```

```
      #0  input1 = 32'b11111111111111111111111111111111;
                input2 = 32'b11111000011111111111111111111111;
      #0 sel =  4'b0000;
      reset = 0;
      clk =0;
  end
  always begin
      #5 clk = !clk;
  end

  always@(posedge clk)
      begin
          for(i = 0; i <= 16 ; i =  i+1)
            begin
                #20; sel = sel + 4'b0001;
            end
      end

endmodule
```

GENERATED VERILOG NETLIST

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : L-2016.03-SP3
// Date      : Wed Dec  8 12:19:10 2021
////////////////////////////////////////////////////////////


module PROJECT_1_A ( input1, input2, reset, clk, sel, out, c_out
);
  input [31:0] input1;
  input [31:0] input2;
  input [3:0] sel;
  output [31:0] out;
  input reset, clk;
  output c_out;
  wire   N16, N19, N20, N21, N22, N23, N24, N25, N27, N28, N29,
N30, N31, N32,
```

```
         N33, N34, N35, N36, N40, N41, N42, N43, N44, N45, N46,
N47, N48, N49,
         N50, N51, N52, N53, N54, N55, N56, N57, N58, N59, N60,
N61, N62, N63
  INV U4332 ( .IN(n4199), .OUT(\div_27/u_div/BAnd [4]) );
  INV U4333 ( .IN(n4200), .OUT(\div_27/u_div/BAnd [3]) );
  INV U4334 ( .IN(n4201), .OUT(\div_27/u_div/BAnd [2]) );
  INV U4335 ( .IN(n4202), .OUT(\div_27/u_div/BAnd [1]) );
  INV U4336 ( .IN(n4203), .OUT(N200) );
  INV U4337 ( .IN(n4204), .OUT(N199) );
  INV U4338 ( .IN(n4205), .OUT(N198) );
  INV U4339 ( .IN(n4206), .OUT(N197) );
  INV U4340 ( .IN(n4207), .OUT(N196) );
  INV U4341 ( .IN(n4208), .OUT(N195) );
  INV U4342 ( .IN(n4209), .OUT(N194) );
  INV U4343 ( .IN(n4210), .OUT(N193) );
  INV U4344 ( .IN(n4211), .OUT(N192) );
  INV U4345 ( .IN(n4212), .OUT(N191) );
  INV U4346 ( .IN(n4213), .OUT(N190) );
  INV U4347 ( .IN(n4214), .OUT(N189) );
  INV U4348 ( .IN(n4215), .OUT(N188) );
  INV U4349 ( .IN(n4216), .OUT(N187) );
  INV U4350 ( .IN(n4217), .OUT(N186) );
  INV U4351 ( .IN(n4218), .OUT(N185) );
  INV U4352 ( .IN(n4219), .OUT(N184) );
  INV U4353 ( .IN(n4220), .OUT(N183) );
  INV U4354 ( .IN(n4221), .OUT(N182) );
  INV U4355 ( .IN(n4222), .OUT(N181) );
  INV U4356 ( .IN(n4223), .OUT(N180) );
  INV U4357 ( .IN(n4224), .OUT(N179) );
  INV U4358 ( .IN(n4225), .OUT(N178) );
  INV U4359 ( .IN(n4226), .OUT(N177) );
  INV U4360 ( .IN(n4227), .OUT(N176) );
  INV U4361 ( .IN(n4228), .OUT(N175) );
  INV U4362 ( .IN(n4229), .OUT(N174) );
  INV U4363 ( .IN(n4230), .OUT(N173) );
  INV U4364 ( .IN(n4231), .OUT(N172) );
  INV U4365 ( .IN(n4232), .OUT(N171) );
  INV U4366 ( .IN(n4233), .OUT(N170) );
  INV U4367 ( .IN(n4234), .OUT(N169) );
endmodule
```

SIMULATION WAVEFORMS

Addition

input 1: 11111111111111111111111111111111

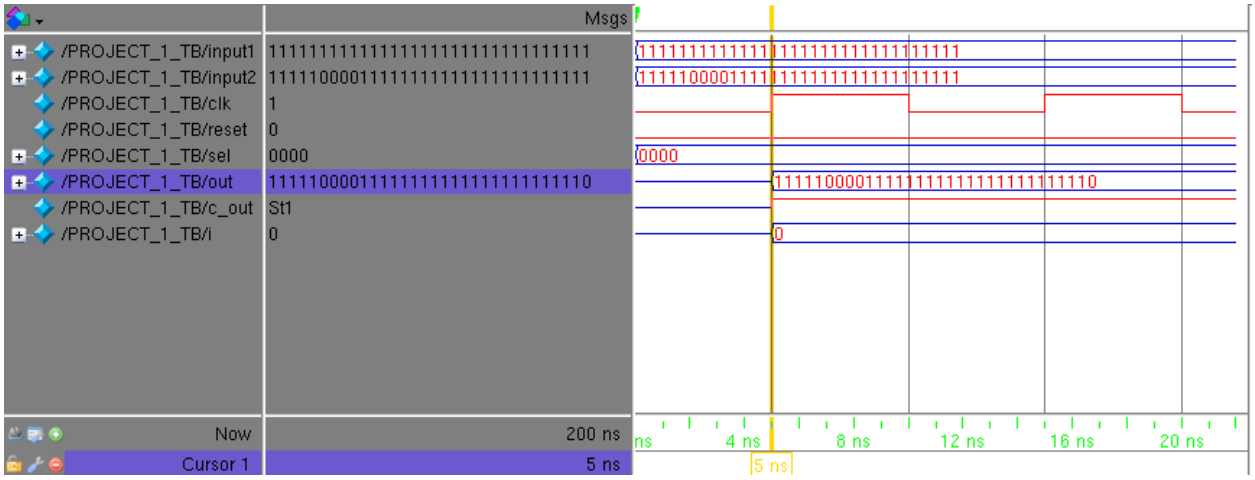
input 2:11111000011111111111111111111111

sel: (0000)

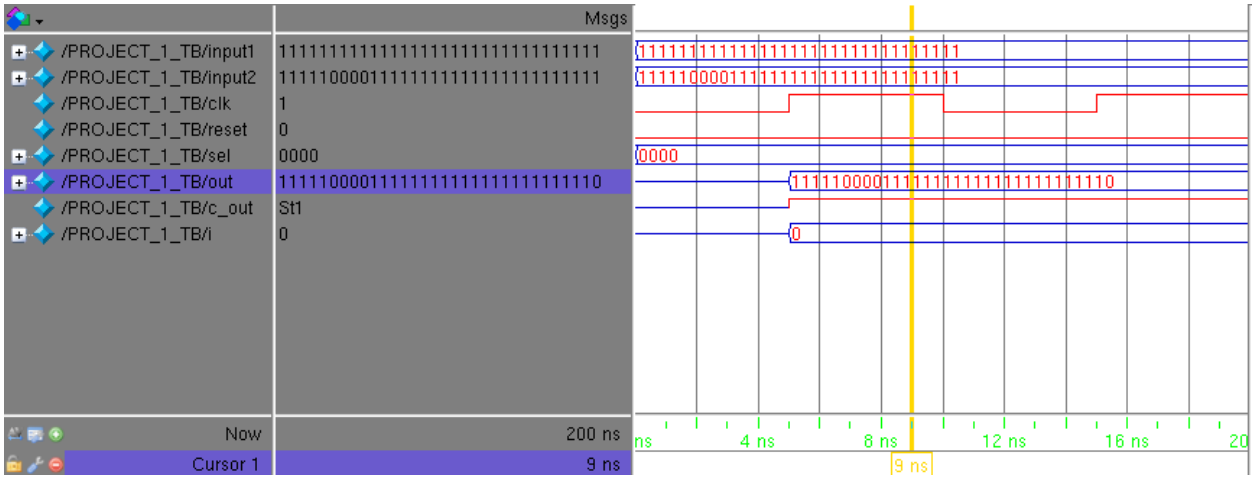
reset : 0

clk : 1

out :input1+input2= 11111000011111111111111111111110



Behavioural Code Simulation



Netlist Simulation

Logical AND

input 1: 11111111111111111111111111111111

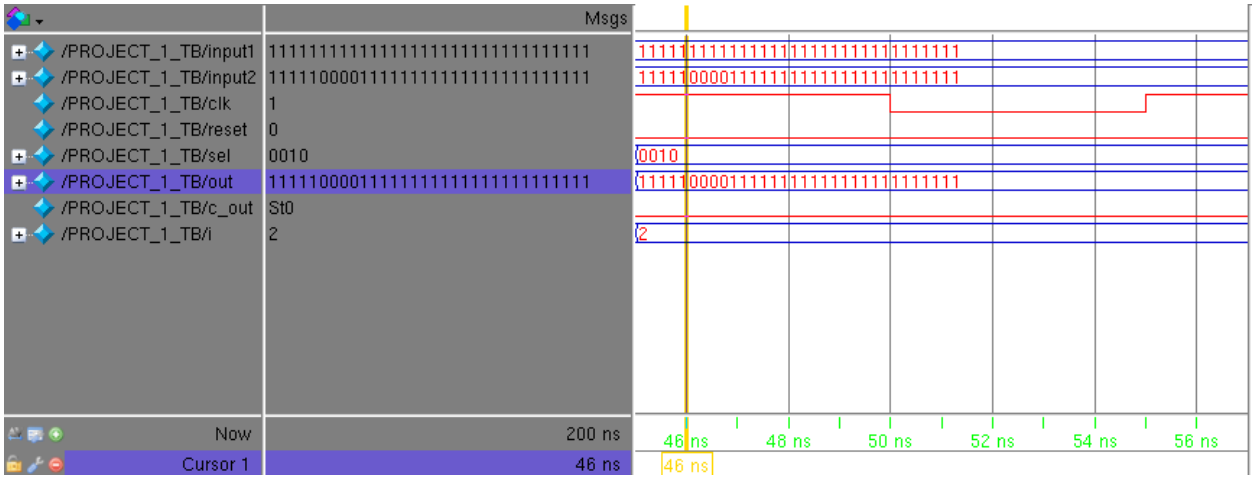
input 2:11111000011111111111111111111111

sel: (0010)

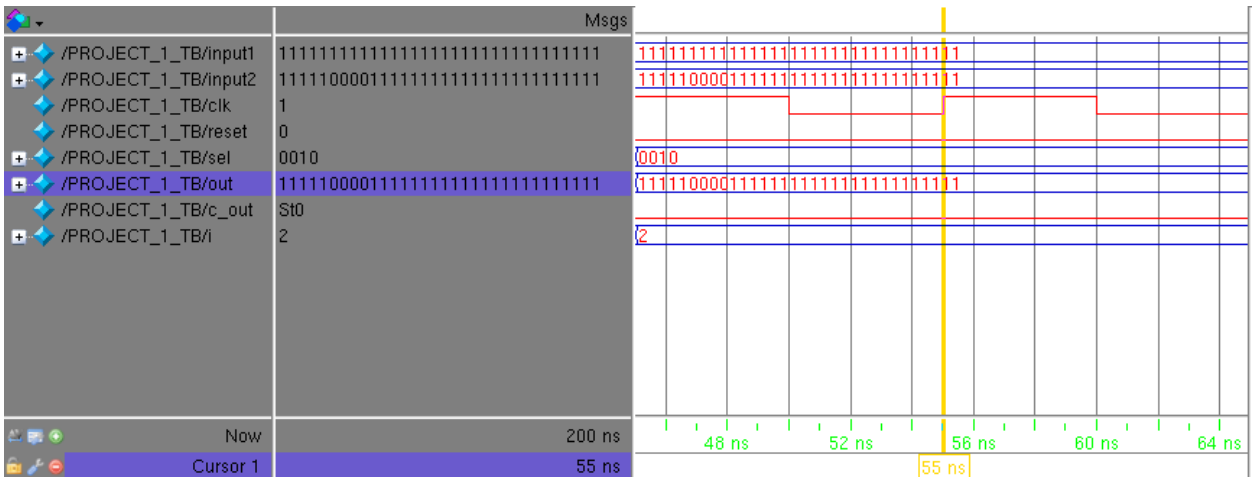
reset : 0

clk : 1

out :input1&input2= 11111000011111111111111111111111



Behavioural Code Simulation



Netlist Simulation

Division

input 1: 11111111111111111111111111111111

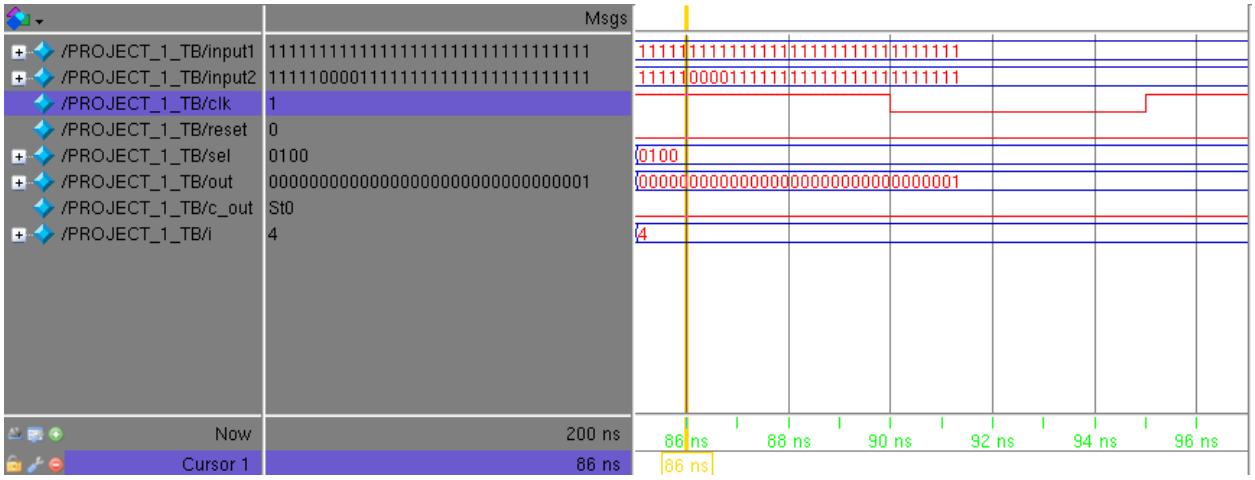
input 2:11111000011111111111111111111111

sel: (0100)

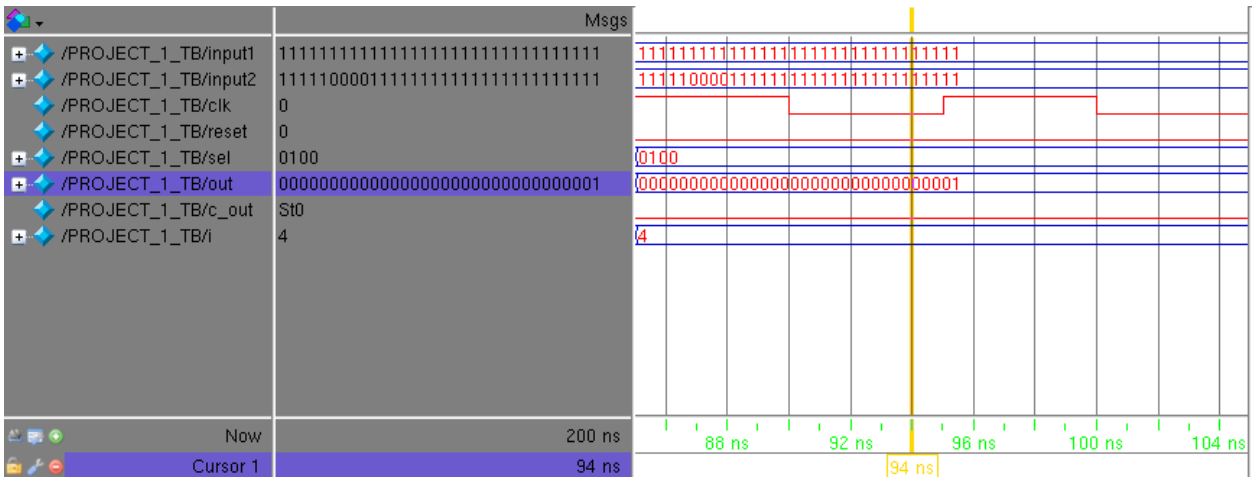
reset : 0

clk : 1

out :input1/input2= 00000000000000000000000000000001



Behavioural Code Simulation



Netlist Simulation

Logical XOR

input 1: 11111111111111111111111111111111

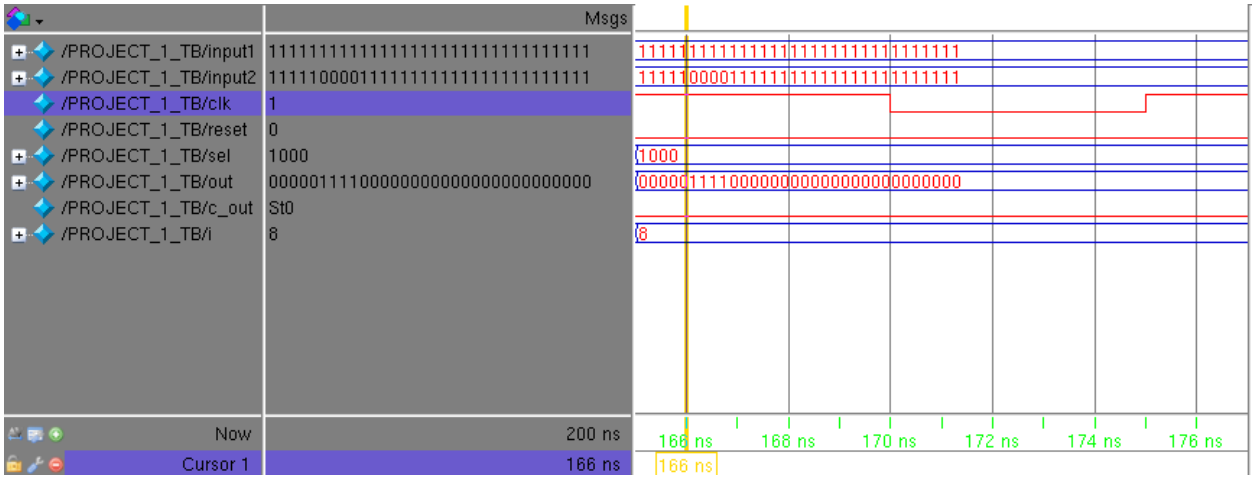
input 2:11111000011111111111111111111111

sel: (1000)

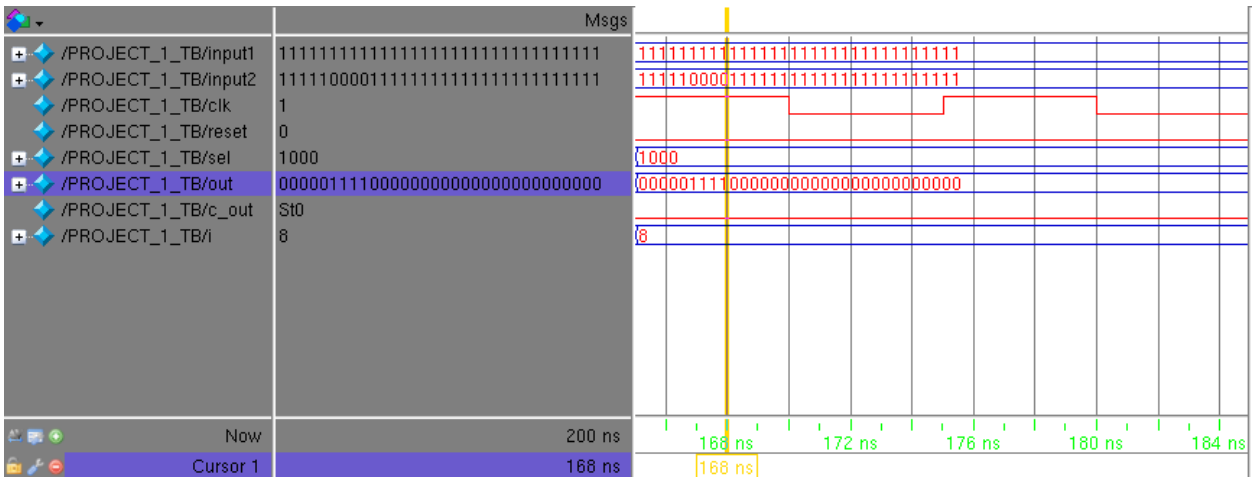
reset : 0

clk : 1

out :input1^input2= 00000111100000000000000000000000

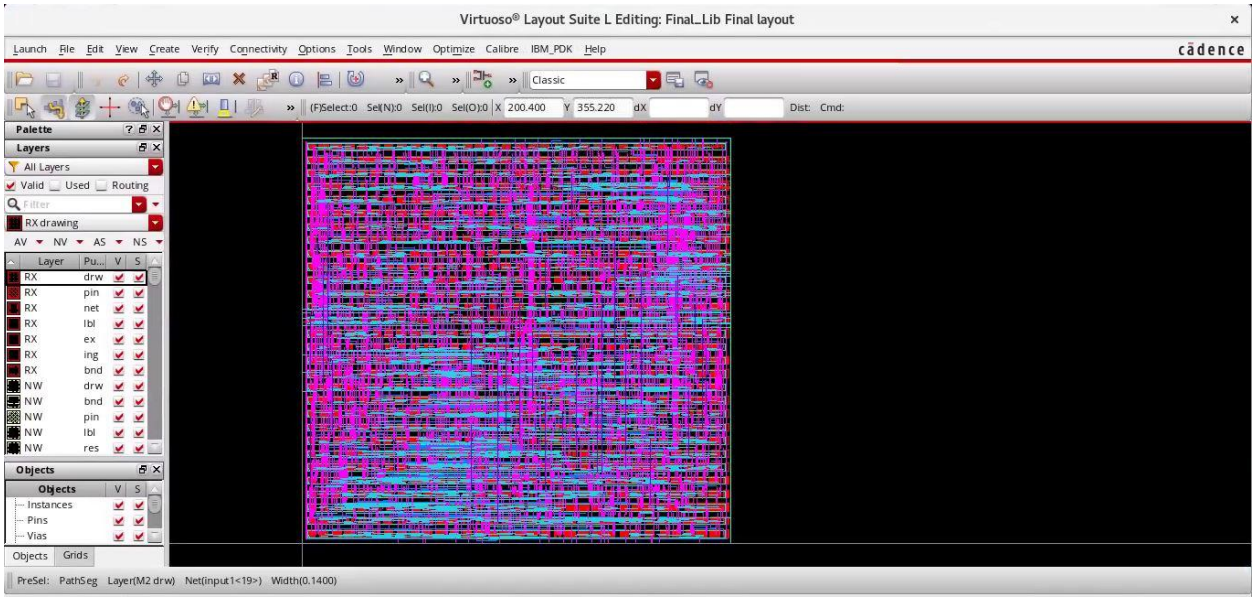


Behavioural Code Simulation



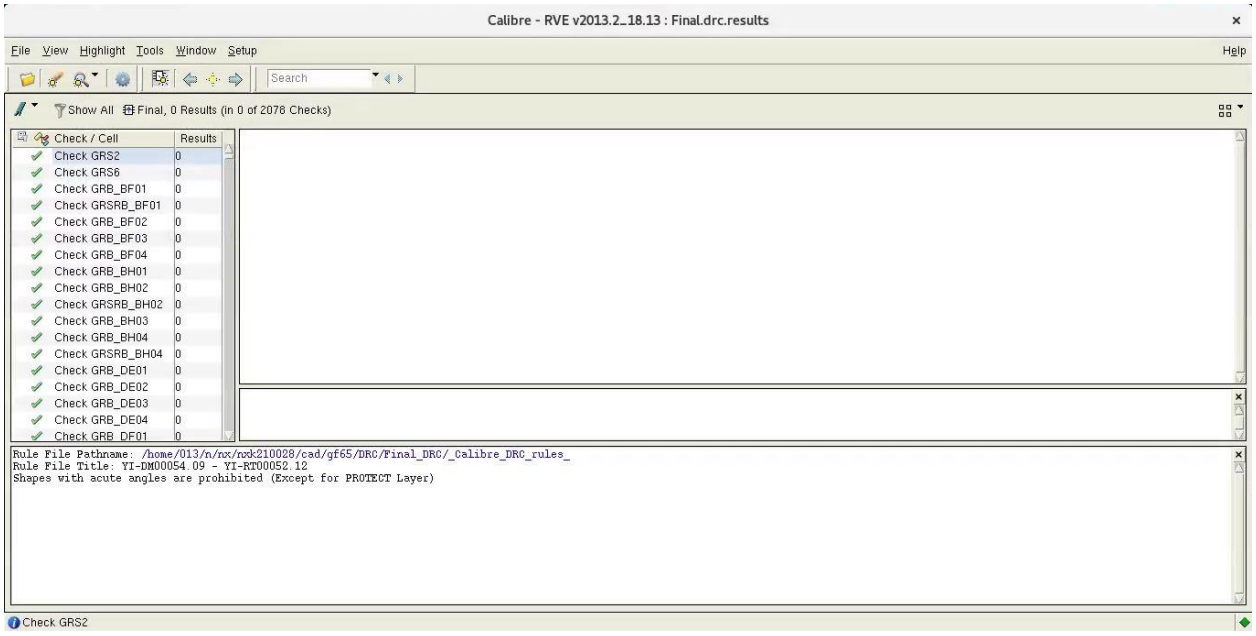
Netlist Simulation

Layout

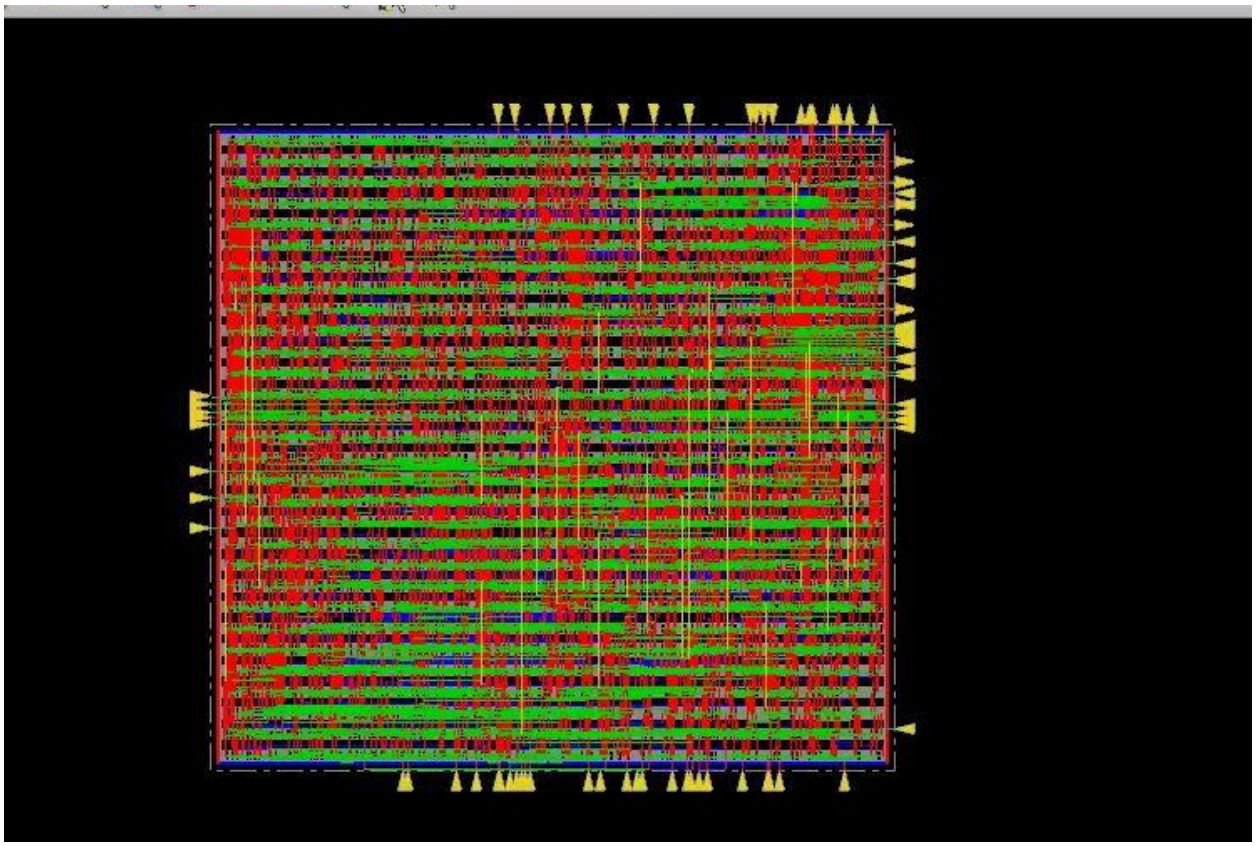


Final Design Layout

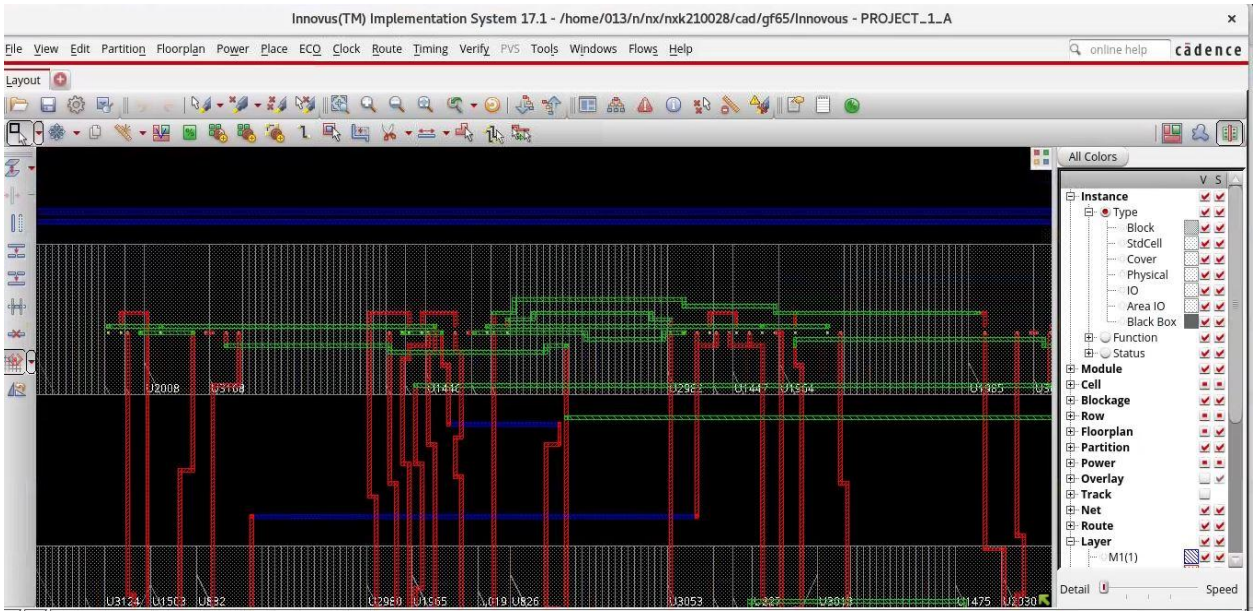
DRC OF FINAL LAYOUT



INNOVUS SPACING AND ROUTING DESIGN



Innovus Design with all spacing and Routing



Spacing

CELL REPORT GENERATED

```
Information: Updating graph... (UID-83)

****************************************
Report : cell
Design : PROJECT_1_A
Version: L-2016.03-SP3
Date   : Wed Dec  8 12:10:12 2021
****************************************

Attributes:
    b - black box (unknown)
    h - hierarchical
    n - noncombinational
    r - removable
    u - contains unmapped logic
```

Cell	Reference	Library	Area	Attributes
C16	NOR_2	gf65	7.774000	
C19	NOR_2	gf65	7.774000	
C23	NOR_2	gf65	7.774000	
C32	NOR_2	gf65	7.774000	
C42	NOR_2	gf65	7.774000	
C321	NOR_2	gf65	7.774000	
C329	NOR_2	gf65	7.774000	
U74	INV	gf65	4.664000	
U75	INV	gf65	4.664000	
U76	AOI_211	gf65	12.440000	
U77	INV	gf65	4.664000	
U78	AOI_22	gf65	12.440000	
U79	OAI_21	gf65	10.880000	
U80	AOI_22	gf65	12.440000	
U81	INV	gf65	4.664000	
U82	INV	gf65	4.664000	
U83	AOI_211	gf65	12.440000	
U84	INV	gf65	4.664000	
U85	AOI_22	gf65	12.440000	
U86	OAI_21	gf65	10.880000	
U87	AOI_22	gf65	12.440000	
U88	INV	gf65	4.664000	

U89	INV	gf65	4.664000
U90	AOI_211	gf65	12.440000
U91	INV	gf65	4.664000
U92	AOI_22	gf65	12.440000
U93	OAI_21	gf65	10.880000
U94	AOI_22	gf65	12.440000
U95	INV	gf65	4.664000
U96	INV	gf65	4.664000
U97	AOI_211	gf65	12.440000
U98	INV	gf65	4.664000
U99	AOI_22	gf65	12.440000
U100	OAI_21	gf65	10.880000
U101	AOI_22	gf65	12.440000
U102	INV	gf65	4.664000
U103	INV	gf65	4.664000
out_reg[0]	DFF	gf65	40.419998 n
out_reg[1]	DFF	gf65	40.419998 n
out_reg[2]	DFF	gf65	40.419998 n
out_reg[3]	DFF	gf65	40.419998 n
out_reg[4]	DFF	gf65	40.419998 n
out_reg[5]	DFF	gf65	40.419998 n
out_reg[6]	DFF	gf65	40.419998 n
out_reg[7]	DFF	gf65	40.419998 n
out_reg[8]	DFF	gf65	40.419998 n
out_reg[9]	DFF	gf65	40.419998 n
out_reg[10]	DFF	gf65	40.419998 n
out_reg[11]	DFF	gf65	40.419998 n
out_reg[12]	DFF	gf65	40.419998 n
out_reg[13]	DFF	gf65	40.419998 n
out_reg[14]	DFF	gf65	40.419998 n
out_reg[15]	DFF	gf65	40.419998 n
out_reg[16]	DFF	gf65	40.419998 n
out_reg[17]	DFF	gf65	40.419998 n
out_reg[18]	DFF	gf65	40.419998 n
out_reg[19]	DFF	gf65	40.419998 n
out_reg[20]	DFF	gf65	40.419998 n
out_reg[21]	DFF	gf65	40.419998 n
out_reg[22]	DFF	gf65	40.419998 n
out_reg[23]	DFF	gf65	40.419998 n
out_reg[24]	DFF	gf65	40.419998 n
out_reg[25]	DFF	gf65	40.419998 n
out_reg[26]	DFF	gf65	40.419998 n
out_reg[27]	DFF	gf65	40.419998 n
out_reg[28]	DFF	gf65	40.419998 n
out_reg[29]	DFF	gf65	40.419998 n

```
out_reg[30]                DFF             gf65            40.419998 n
out_reg[31]                DFF             gf65            40.419998 n
r82/ULTI2_25               NAND_2          gf65            7.774000
r82/ULTI2_26               NAND_2          gf65            7.774000
r82/ULTI2_27               NAND_2          gf65            7.774000
r82/ULTI2_28               NAND_2          gf65            7.774000
r82/ULTI2_29               NAND_2          gf65            7.774000
r82/ULTI2_30               NAND_2          gf65            7.774000
r82/UNGT0                  NAND_2          gf65            7.774000
r82/UNLT0                  NAND_2          gf65            7.774000
--------------------------------------------------------------------------
-----------
Total 4782 cells
46191.880435
```

PRIME TIME REPORT GENERATED

```
****************************************
Report : timing
     -path_type full
     -delay_type min_max
     -input_pins
     -max_paths 1
     -transition_time
     -capacitance
     -sort_by slack
Design : PROJECT_1_A
Version: M-2016.12-SP3-1
Date   : Wed Dec  8 17:32:17 2021
****************************************


  Startpoint: out_reg[20]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: out[20] (output port)
  Path Group: (none)
  Path Type: min

  Point                                    Cap      Trans
Incr       Path
```

```
-------------------------------------------------------------------
-------------
  out_reg[20]/CLK (DFF)                                  0.00
0.00       0.00 r
  out_reg[20]/Q (DFF)                    0.05            0.15
0.19       0.19 f
  out[20] (out)                                          0.15
0.00       0.19 f
  data arrival time
0.19

-------------------------------------------------------------------
-------------
  (Path is unconstrained)



  Startpoint: out_reg[20]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: out[20] (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Cap      Trans
Incr       Path

-------------------------------------------------------------------
-------------
  out_reg[20]/CLK (DFF)                                   0.05
0.00       0.00 r
  out_reg[20]/Q (DFF)                    0.05            0.19
0.25       0.25 r
  out[20] (out)                                          0.19
0.00       0.25 r
  data arrival time
0.25

-------------------------------------------------------------------
-------------
  (Path is unconstrained)
```

CONCLUSION

- We have Designed our 32 Bit ALU and verified the functionality along with the standard library and the library we created.
- Based on the results the functionality and correctness of our cells was verified.
- In Order to Complete this project we have used cadence tools,model sim and Design Vision.