CSE 4207 CT 4 Assignment
Roll No: 1903073

Assignment Problem:
Category: C
Word Size = 6
ALU Operations = SHR, XNOR

Solution:
Video:

Have you uploaded the video?	YES
Check List 1: Have you explained the design using testbenches to prove that your circuit is working correctly and giving correct results?	YES
Check List 2: Have you shown full synthesis results showing all the required info including RTL Synthesis, RTL Floorplan, RTL Power Analysis, GDS and Heatmap (for details, see assignment template doc)?	YES
NB: Failing to upload video will cause heavy point penalty (5-6 Marks)	
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

HDL Code:

Check List: Have you added all the modules written in verilog including ALU, ALU_OP1, ALU_OP2, CONTROLLER, TOP, TOP_TESTBENCH, ALU_TESTBENCH, CONTROLLER_TESTBENCH?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

Google drive link of all the code files:

19CT4Assignment_Code_Files_1903073

alu.v

```
module alu
(
    input wire [5:0] alu_in1, alu_in2,
    input wire [1:0] alu_op,
    output wire [5:0] alu_out,
    output wire alu_flag
);

reg [5:0] result;
```

```
wire [5:0] SHR, XNOR;

ALU_SHR_Nbit SHR1
(
    .A(alu_in1),
    .B(alu_in2),
    .R(SHR)
);

XNOR_Nbit XNOR1
(
    .A(alu_in1),
    .B(alu_in2),
    .R(XNOR)
);

always @(*)
begin
    case (alu_op)
        2'b00: result = SHR; // Right Shift
        2'b01: result = XNOR; // XNOR
        default: result = 6'b0;
    endcase
end

assign alu_out = result;
assign alu_flag = (result > 6'b0);

endmodule
```

alu_op1.v

```
module ALU_SHR_Nbit
(
    input wire [5:0] A,
    input wire [5:0] B,
    output wire [5:0] R
);

    assign R = A >> B;

endmodule
```

alu_op2.v

```
module XNOR_Nbit
(
    input wire [5:0] A,B,
    output wire [5:0] R
);
```

```
assign R = ~(A ^ B);
endmodule
```

controller.v

```
module alu_controller
(
    input wire clk, reset, start,
    input wire [5:0] a, b,
    input wire [1:0] op,

    input wire [5:0] alu_out,
    input wire alu_flag,
    output wire [5:0] alu_in1, alu_in2,
    output wire [1:0] alu_op,

    output reg [5:0] result,
    output reg flag
);

reg [2:0] pstate, nstate;

parameter [2:0] START = 3'b000,
                ONE = 3'b001,
                TWO = 3'b010,
                THREE = 3'b011,
                FINISH = 3'b100;

// Memory
always @(posedge clk, posedge reset)
begin : PSR
    if (reset)
    begin
        pstate <= START;
    end
    else
    begin
        pstate <= nstate;
    end
end

// Next State Logic
always @(*)
begin: NSOL
    // Monitor output
    if(pstate == START)
        $monitor("pstate = START -> clk = %b, reset = %b, start =
%b, a = %d, b = %d, op = %b, result = %d, flag =
%b\n**********************************\n", clk, reset, start, a,
b, op, result, flag);
```

```
    else if(pstate == ONE)
        $monitor("pstate = ONE -> clk = %b, reset = %b, start =
%b, a = %d, b = %d, op = %b, result = %d, flag =
%b\n**********************************\n", clk, reset, start, a,
b, op, result, flag);
    else if(pstate == TWO)
        $monitor("pstate = TWO -> clk = %b, reset = %b, start =
%b, a = %d, b = %d, op = %b, result = %d, flag =
%b\n**********************************\n", clk, reset, start, a,
b, op, result, flag);
    else if(pstate == THREE)
        $monitor("pstate = THREE -> clk = %b, reset = %b, start =
%b, a = %d, b = %d, op = %b, result = %d, flag =
%b\n**********************************\n", clk, reset, start, a,
b, op, result, flag);
    else if(pstate == FINISH)
        $monitor("pstate = FINISH -> clk = %b, reset = %b, start =
%b, a = %d, b = %d, op = %b, result = %d, flag =
%b\n**********************************\n", clk, reset, start, a,
b, op, result, flag);

    nstate = pstate;

    // Next State Logic and Output Logic
    begin: NSL
    case (pstate)
        START:
        begin
            if (start)
                nstate = ONE;
        end

        ONE:
        begin
            nstate = TWO;
        end

        TWO:
        begin
            nstate = THREE;
        end

        THREE:
        begin
            nstate = FINISH;
        end

        FINISH:
        begin
            nstate = FINISH;
        end
```

```
        default:
            nstate = START;
    endcase
end
begin: OL
    result = 6'b0;
    flag = 0;
    case (pstate)
        START:
        begin
            result = 6'b0;
            flag = 0;
        end

        ONE:
        begin
            result = alu_out;
            flag = alu_flag;
        end

        TWO:
        begin
            result = alu_out;
            flag = alu_flag;
        end

        THREE:
        begin
            result = alu_out;
            flag = alu_flag;
        end

        FINISH:
        begin
            result = alu_out;
            flag = alu_flag;
        end
        default:
            nstate = START;
    endcase
end
end

assign alu_in1 = a;
assign alu_in2 = b;
assign alu_op = op;

endmodule
```

top.v

```
module alu_top
(
    input wire clk, reset, start,
    input wire [5:0] a, b,
    input wire [1:0] op,
    output wire [5:0] result,
    output wire flag
);

wire [5:0] alu_in1, alu_in2;
wire [1:0] alu_op;
wire [5:0] alu_out;
wire alu_flag;

alu_controller controller1
(
    .clk(clk),
    .reset(reset),
    .start(start),
    .a(a),
    .b(b),
    .op(op),
    .alu_in1(alu_in1),
    .alu_in2(alu_in2),
    .alu_op(alu_op),
    .alu_out(alu_out),
    .alu_flag(alu_flag),
    .result(result),
    .flag(flag)
);

alu datapath1
(
    .alu_in1(alu_in1),
    .alu_in2(alu_in2),
    .alu_op(alu_op),
    .alu_out(alu_out),
    .alu_flag(alu_flag)
);

endmodule
```

top_testbench.v

```
`timescale 1ns/1ns

module alu_top_tb;

reg clk, reset, start;
reg [5:0] a, b;
reg [1:0] op;
```

```
wire [5:0] result;
wire flag;

alu_top top1
(
    .clk(clk),
    .reset(reset),
    .start(start),
    .a(a),
    .b(b),
    .op(op),
    .result(result),
    .flag(flag)
);

initial begin

    clk = 0;
    forever #5 clk = ~clk;  // Toggle clk every 5 time units
end

initial begin
    $dumpfile("alu_top.vcd");
    $dumpvars(0, alu_top_tb);

    clk <= 0;
    reset <= 0;
    start <= 0;
    a <= 0;
    b <= 0;
    op <= 0;

    @(negedge clk);
    reset <= 1;

    @(negedge clk);
    reset <= 0;
    start <= 1;
    a <= 6'b111100; // 60
    b <= 6'b000010; // 2
    op <= 2'b00; // Right Shift: 111100 >> 2 = 001111 (15, flag =
1)
    #20;

    a <= 6'b000000;
    b <= 6'b000001;
    op <= 2'b00; // Right Shift: 000000 >> 1 = 000000 (0, flag =
0)
     #20;

    a <= 6'b101010; // 42
```

```
    b <= 6'b110011; // 51
    op <= 2'b01; // XNOR: ~(101010 ^ 110011) = ~011001 = 100110
(38, flag = 1)
    #20;

    $finish();
end

endmodule
```

alu_testbench.v

```
`timescale 1ns / 1ps


module alu_tb;


   // Inputs
   reg [5:0] alu_in1;
   reg [5:0] alu_in2;
   reg [1:0] alu_op;

   // Outputs
   wire [5:0] alu_out;
   wire alu_flag;

   // Instantiate the Unit Under Test (UUT)
   alu uut (
       .alu_in1(alu_in1),
       .alu_in2(alu_in2),
       .alu_op(alu_op),
       .alu_out(alu_out),
       .alu_flag(alu_flag)
   );

   initial begin
       $dumpfile("alu_tb.vcd");
       $dumpvars(0, alu_tb);
```

```
        // Initialize Inputs
        alu_in1 = 6'b0;
        alu_in2 = 6'b0;
        alu_op = 2'b0;

        // Wait for global reset
        #100;

        // Test Case 1: Right Shift (alu_op = 00)
        alu_op = 2'b00;
        alu_in1 = 6'b110000; // 48 in decimal
        alu_in2 = 6'b000010; // Shift by 2
        #20;
        $display("Test 1 - SHR: in1=%b, in2=%b, out=%b, flag=%b",
alu_in1, alu_in2, alu_out, alu_flag);

        // Test Case 2: Right Shift (alu_op = 00)
        alu_in1 = 6'b001100; // 12 in decimal
        alu_in2 = 6'b000001; // Shift by 1
        #20;
        $display("Test 2 - SHR: in1=%b, in2=%b, out=%b, flag=%b",
alu_in1, alu_in2, alu_out, alu_flag);

        // Test Case 3: XNOR (alu_op = 01)
        alu_op = 2'b01;
        alu_in1 = 6'b101010;
        alu_in2 = 6'b110011;
        #20;
        $display("Test 3 - XNOR: in1=%b, in2=%b, out=%b, flag=%b",
alu_in1, alu_in2, alu_out, alu_flag);

        // Test Case 4: XNOR (alu_op = 01)
        alu_in1 = 6'b111111;
        alu_in2 = 6'b111111;
        #20;
        $display("Test 4 - XNOR: in1=%b, in2=%b, out=%b, flag=%b",
```

```
alu_in1, alu_in2, alu_out, alu_flag);

       // Test Case 5: Default case (alu_op = 10 or 11)
       alu_op = 2'b10;
       alu_in1 = 6'b101010;
       alu_in2 = 6'b110011;
       #20;
       $display("Test 5 - Default: in1=%b, in2=%b, out=%b,
flag=%b", alu_in1, alu_in2, alu_out, alu_flag);

       // Test Case 6: Zero output test
       alu_op = 2'b01; // XNOR
       alu_in1 = 6'b000000;
       alu_in2 = 6'b000000;
       #20;
       $display("Test 6 - XNOR Zero: in1=%b, in2=%b, out=%b,
flag=%b", alu_in1, alu_in2, alu_out, alu_flag);

       // Finish simulation
       #20;
       $finish;
   end

endmodule
```

controller_testbench.v

```
`timescale 1ns / 1ps
module alu_controller_tb;

   reg clk, reset, start;
   reg [5:0] a, b;
   reg [1:0] op;
   reg [5:0] alu_out;
   reg alu_flag;
   wire [5:0] alu_in1, alu_in2;
   wire [1:0] alu_op;
   wire [5:0] result;
   wire flag;

   alu_controller dut (
       .clk(clk),
       .reset(reset),
       .start(start),
       .a(a),
       .b(b),
       .op(op),
       .alu_out(alu_out),
       .alu_flag(alu_flag),
       .alu_in1(alu_in1),
       .alu_in2(alu_in2),
       .alu_op(alu_op),
       .result(result),
       .flag(flag)
   );

   initial begin
       $dumpfile("alu_controller.vcd");
       $dumpvars(0, alu_controller_tb);

       clk = 0;
```

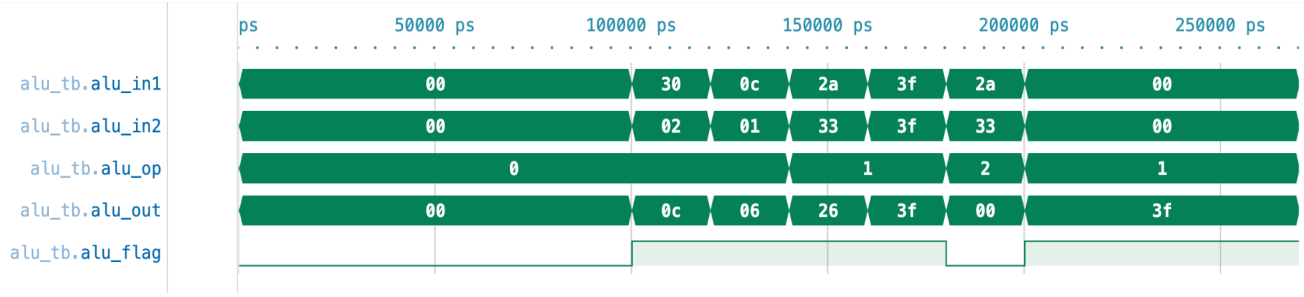
```
    forever #5 clk = ~clk;
end


initial begin

    reset = 0;
    start = 0;
    a = 6'b000000;
    b = 6'b000000;
    op = 2'b00;
    alu_out = 6'b000000;
    alu_flag = 0;

    #10;
    reset = 1;

    #10;
    reset = 0;
    start = 1;
    a = 6'b111100;
    b = 6'b000010;
    op = 2'b00;
    alu_out = 6'b001111;
    alu_flag = 1;


    #15;
    a = 6'b000000;
    b = 6'b000001;
    op = 2'b00;
    alu_out = 6'b000000;
    alu_flag = 0;
```

```
        #10;
        a = 6'b101010;
        b = 6'b110011;
        op = 2'b01;
        alu_out = 6'b100110;
        alu_flag = 1;

        #20 $finish;
    end

    // Monitor signals
    initial begin
        $monitor("Time=%0t: clk=%b, reset=%b, start=%b, a=%d, b=%d,
op=%b, alu_in1=%d, alu_in2=%d, alu_op=%b, result=%d, flag=%b",
                 $time, clk, reset, start, a, b, op, alu_in1,
alu_in2, alu_op, result, flag);
    end

endmodule
```

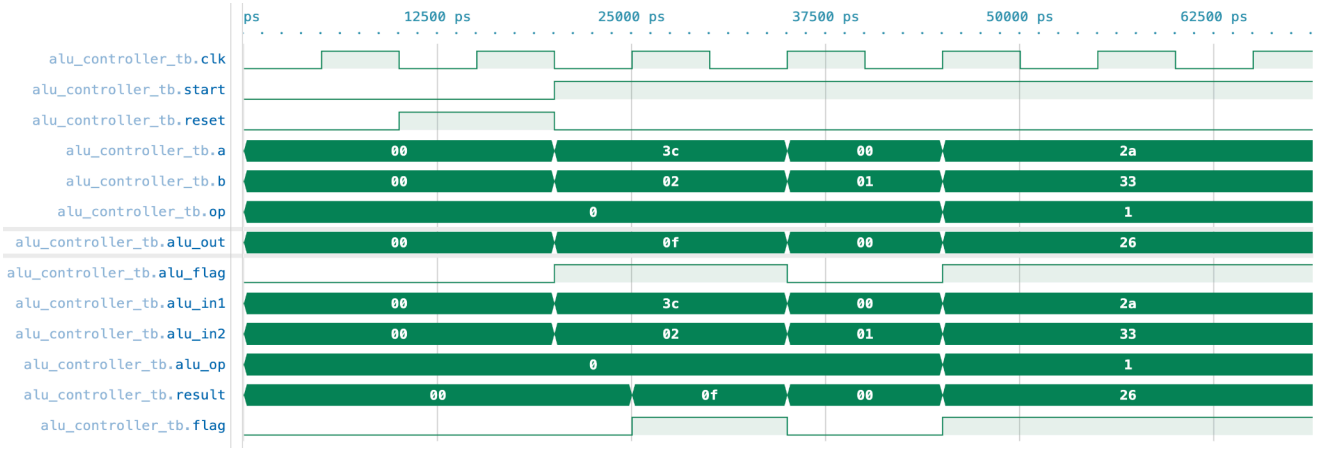
RTL Timing Diagram:

Check List: Have you added all the timing diagrams of ALU_TESTBENCH, CONTROLLER_TESTBENCH, TOP_TESTBENCH?	YES/NO
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

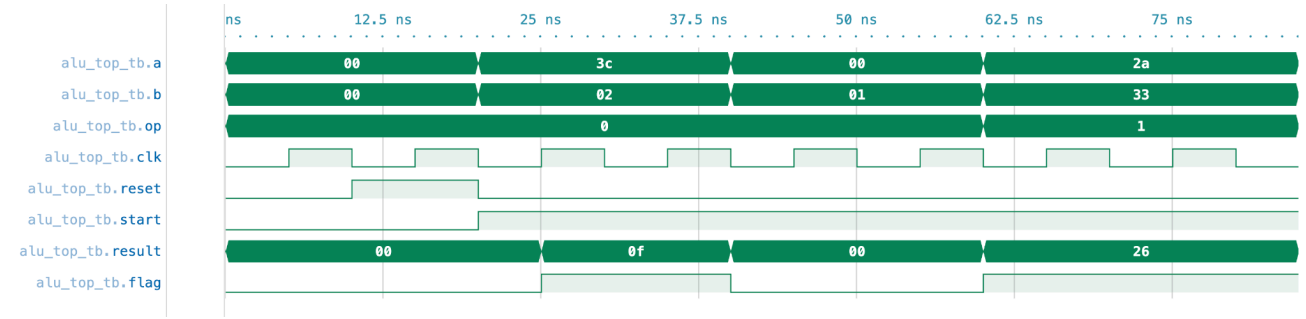
alu_tb.v



controller_tb.v



`top_tb.v`



RTL Synthesis (130nm Skywater PDK with OpenLane toolchain):

Check List: Have you added *RTL synthesis summary*, *RTL synthesized design figure* and *Standard cell usage in synthesized design*?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

RTL synthesis summary

post_dff.rpt:

Metric	Count
Number of Wires	**111**
Number of wire bits	**131**
Number of public wires	**10**
Number of public wire bits	**30**
Number of ports	**8**
Number of port bits	**24**
Number of memories	**0**
Number of memory bits	**0**

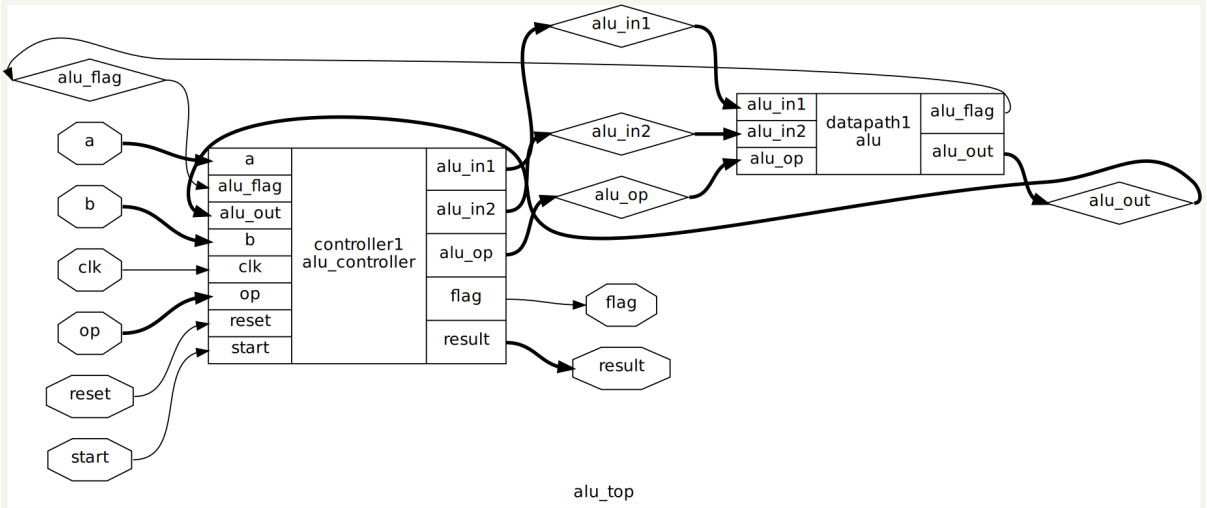
Number of processes	**0**
Number of cells	**114**

stat.rpt:

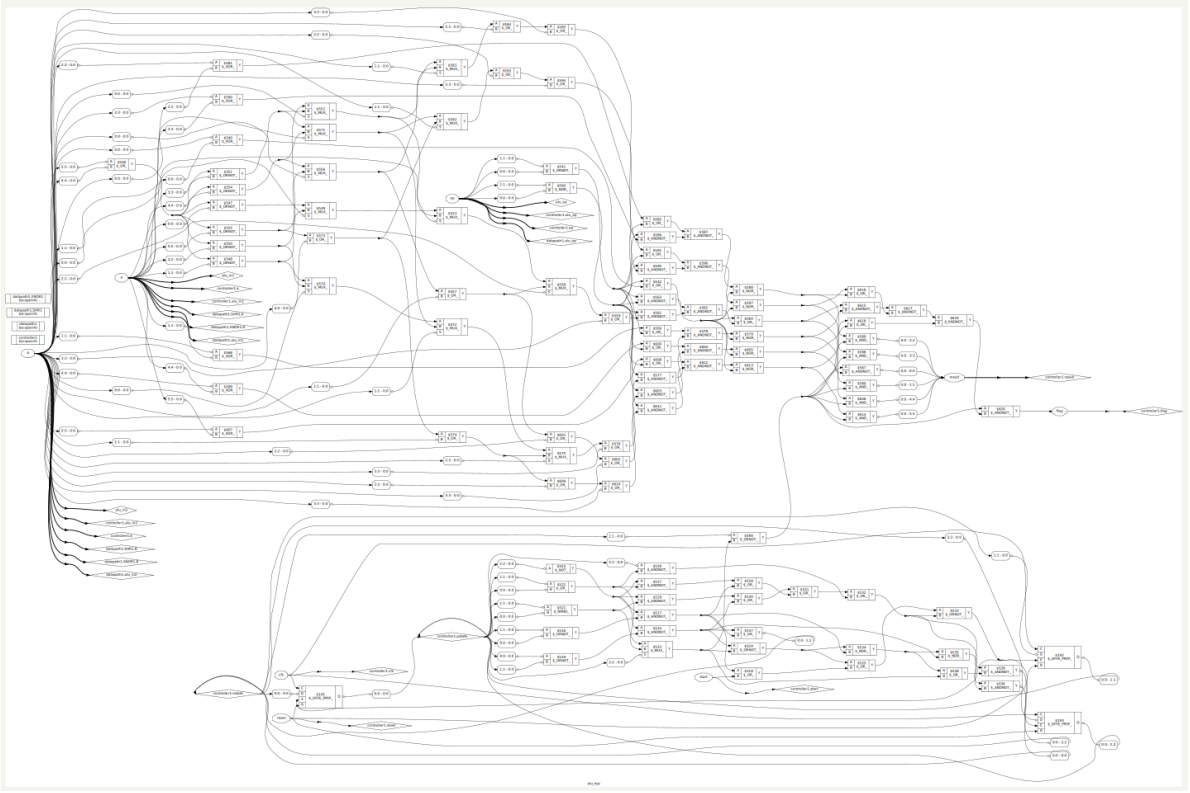
Metric	Count
Number of Wires	**63**
Number of wire bits	**79**
Number of public wires	**11**
Number of public wire bits	**27**
Number of ports	**8**
Number of port bits	**24**
Number of memories	**0**
Number of memory bits	**0**
Number of processes	**0**
Number of cells	**62**

RTL synthesized design figure

Hierarchy.dot:



Primitive_techmap.dot:



Standard cell usage in synthesized design

sky130_fd_sc_hd__a2111o_2	1
sky130_fd_sc_hd__a21o_2	1
sky130_fd_sc_hd__a21oi_2	8
sky130_fd_sc_hd__a2bb2o_2	2
sky130_fd_sc_hd__a31o_2	1
sky130_fd_sc_hd__and2_2	4
sky130_fd_sc_hd__and3_2	2
sky130_fd_sc_hd__and4b_2	1
sky130_fd_sc_hd__dfrtp_2	3
sky130_fd_sc_hd__inv_2	6
sky130_fd_sc_hd__mux2_1	4
sky130_fd_sc_hd__nand2_2	1
sky130_fd_sc_hd__nand2b_2	1
sky130_fd_sc_hd__nor2_2	5
sky130_fd_sc_hd__o2111a_2	1
sky130_fd_sc_hd__o211ai_2	1
sky130_fd_sc_hd__o21a_2	3
sky130_fd_sc_hd__o21bai_2	1
sky130_fd_sc_hd__o22ai_2	1
sky130_fd_sc_hd__o32a_2	2
sky130_fd_sc_hd__or2_2	5
sky130_fd_sc_hd__or3_2	3
sky130_fd_sc_hd__or4b_2	1
sky130_fd_sc_hd__xor2_2	4

RTL Floorplan (130nm Skywater PDK with OpenLane toolchain)

Check List: Have you added RTL Floorplan info?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

General RTL Floorplan Information		
Parameter	**Value**	**Unit**
Database Units(DBU)		**--**
Site Size	**(2.72, 0.46)**	**µm**
Die Area (BBox)	**(0.0 0.0) to (49.705 60.425)**	**µm**
Core Area(BBox)	**(5.52 10.88) to (44.16 48.96)**	**µm**
Core Area	**1471.42**	**µm2**
Die Area	**3003.42**	**µm2**
Placement Utilization		**%**

Openroad_floorplan.log:

```
Reading design constraints file at '/nix/store/b5k0h8cjlpypdp7lb98p9n8qqanyralc-python3-3.11.9-env/lib/python3.11/site-packages/openlane/scripts/base.sdc$
[WARNING] No CLOCK_PORT found. A dummy clock will be used.
[WARNING STA-0366] port '__VIRTUAL_CLK__' not found.
[INFO] Using clock __VIRTUAL_CLK__…
[INFO] Setting output delay to: 2.0
[INFO] Setting input delay to: 2.0
[WARNING STA-0366] port '__VIRTUAL_CLK__' not found.
[INFO] Setting load to: 0.033442
[INFO] Setting clock uncertainty to: 0.25
[INFO] Setting clock transition to: 0.14999999999999999444888487687421729788184165954589843 75
[WARNING STA-0419] transition time can not be specified for virtual clocks.
[INFO] Setting timing derate to: 5%
[WARNING STA-0450] virtual clock __VIRTUAL_CLK__ can not be propagated.
Using site height: 2.72 and site width: 0.46…
[INFO] Using relative sizing for the floorplan.
[INFO IFP-0001] Added 14 rows of 84 site unithd.
[INFO IFP-0030] Inserted 0 tiecells using sky130_fd_sc_hd__conb_1/LO.
[INFO IFP-0030] Inserted 0 tiecells using sky130_fd_sc_hd__conb_1/HI.
[INFO] Extracting DIE_AREA and CORE_AREA from the floorplan
[INFO] Floorplanned on a die area of 0.0 0.0 49.705 60.425 (µm).
[INFO] Floorplanned on a core area of 5.52 10.88 44.16 48.96 (µm).
Writing metric design__die__bbox: 0.0 0.0 49.705 60.425
Writing metric design__core__bbox: 5.52 10.88 44.16 48.96
Setting global connections for newly added cells…
[INFO] Setting global connections...
Updating metrics…
Cell type report:                       Count       Area
  Inverter                                  6      22.52
  Sequential cell                           3      78.83
  Multi-Input combinational cell           53     496.73
  Total                                    62     598.07
```

Or_metrix_out.json:


```
  "design__die__bbox": "0.0 0.0 49.705 60.425",
  "design__core__bbox": "5.52 10.88 44.16 48.96",
  "design__io": 24,
  "design__die__area": 3003.42,
  "design__core__area": 1471.41,
  "design__instance__count": 62,
  "design__instance__area": 598.074,
  "design__instance__count__stdcell": 62,
  "design__instance__area__stdcell": 598.074,
  "design__instance__count__macros": 0,
  "design__instance__area__macros": 0,
  "design__instance__utilization": 0.406463,
  "design__instance__utilization__stdcell": 0.406463,
  "design__instance__count__class:inverter": 6,
  "design__instance__count__class:sequential_cell": 3,
  "design__instance__count__class:multi_input_combinational_cell": 53,
  "flow__warnings__count": 6,
  "flow__errors__count": 0
```

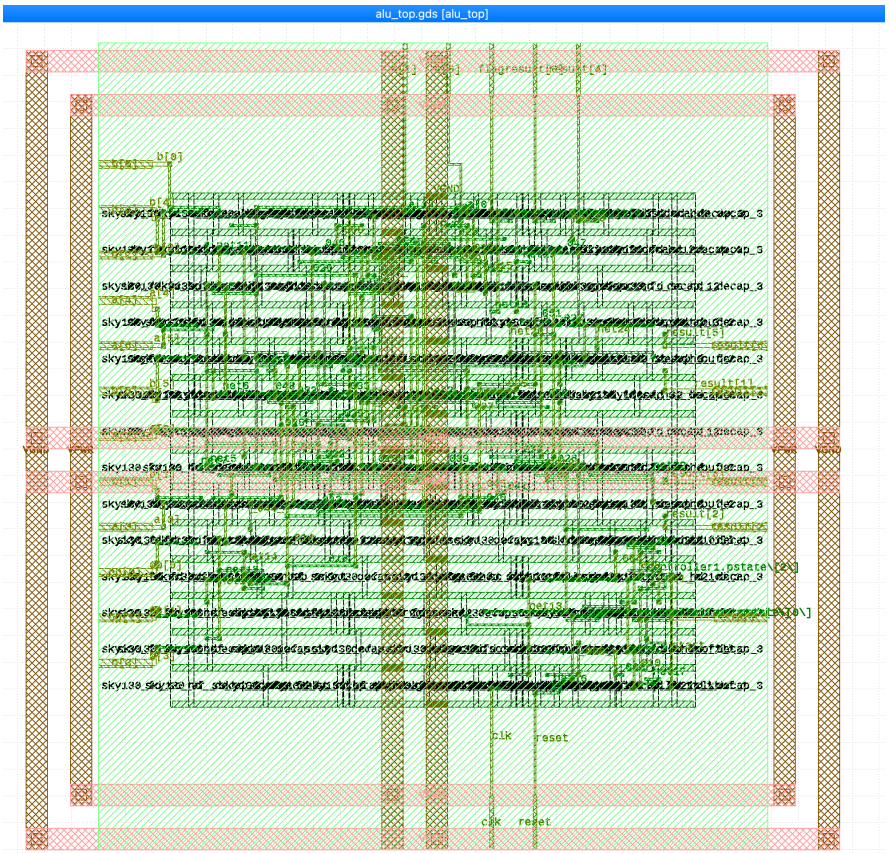

RTL Power Analysis (130nm Skywater PDK with OpenLane toolchain)

Check List: Have you added RTL Power Analysis info?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	

```
========================================================================
 report_power
========================================================================
====================== nom_tt_025C_1v80 Corner ===================================

Group                  Internal    Switching      Leakage        Total
                          Power        Power        Power        Power (Watts)
----------------------------------------------------------------------------
Sequential           5.102053e-06 4.209583e-06 3.768916e-11 9.311674e-06  16.4%
Combinational        1.747334e-05 3.005282e-05 2.372030e-10 4.752640e-05  83.6%
Clock                0.000000e+00 0.000000e+00 2.203231e-10 2.203231e-10   0.0%
Macro                0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00   0.0%
Pad                  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00   0.0%
----------------------------------------------------------------------------
Total                2.257540e-05 3.426240e-05 4.952152e-10 5.683830e-05 100.0%
                            39.7%        60.3%         0.0%
```

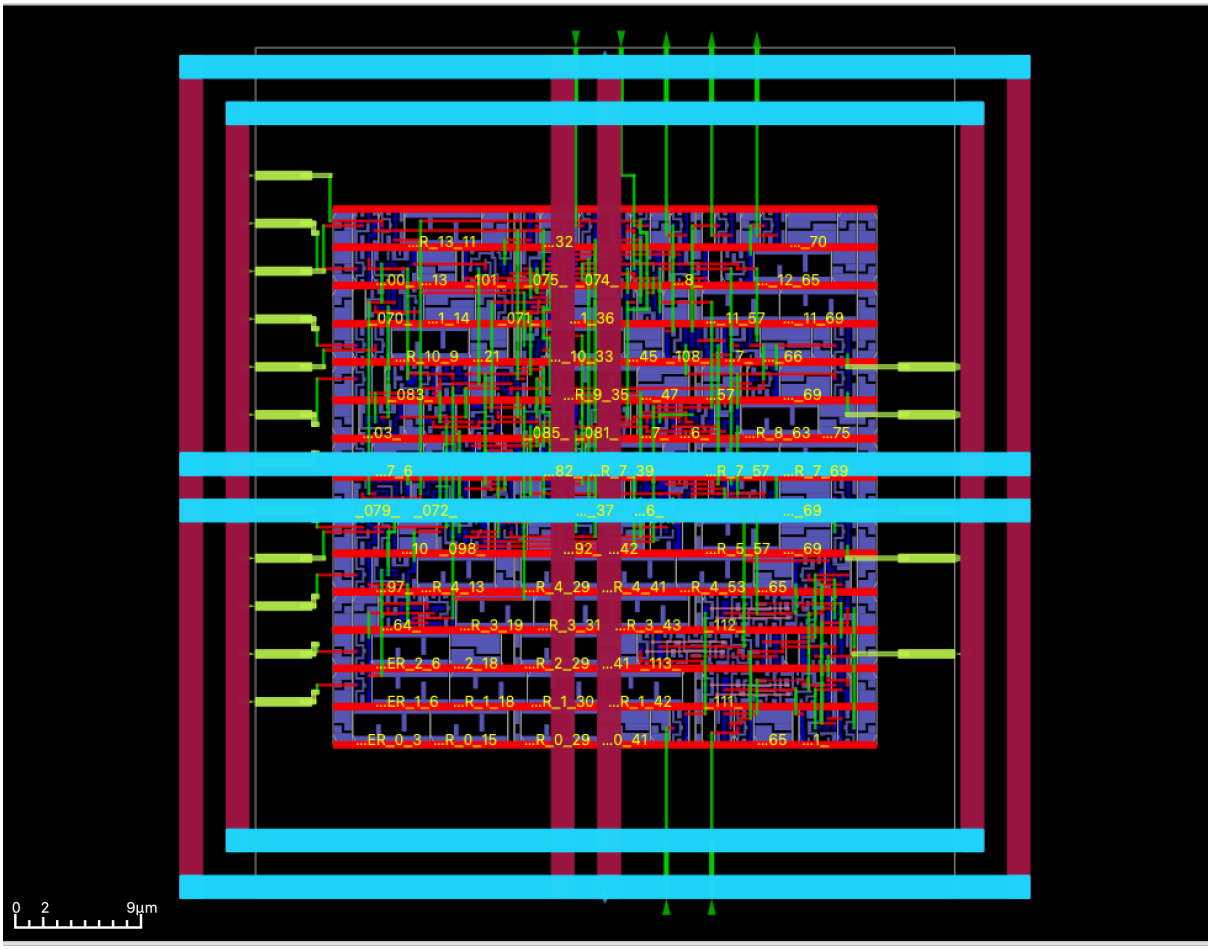
GDS Layout (130nm Skywater PDK with OpenLane toolchain)

Check List: Have you added the GDS Layout figure?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	



Heatmap (130nm Skywater PDK with OpenLane toolchain)

Check List: Have you added the heatmap?	YES
NB: Failing to add any required info will cause point penalty (1-2 Marks)	


0 2 9µm