CSE 4207 CT 4 Assignment

Roll No: 1903048

Assignment Problem:

Category: A
Word Size = 7bits
ALU Operations = NOT ,ROR

HDL Code:

Check List: Have you added all the modules written in verilog including ALU, ALU_OP1, ALU_OP2, ALU_TESTBENCH?	YES

ALU_OP1.v

```
module ALU_OP1
(
    input wire [6:0] A,
    output wire [6:0] R
);
    assign R = ~A;
endmodule
```

ALU_OP2.v

```
module ALU_OP2
(
    input wire [6:0] A,
    input wire [2:0] B,      // Number of bits to rotate (0-
7)
    output reg [6:0] R
);
    always @(*) begin
        case (B)
            3'd0: R = A;
// No rotation
            3'd1: R = {A[0], A[6:1]};
            3'd2: R = {A[1:0], A[6:2]};
            3'd3: R = {A[2:0], A[6:3]};
            3'd4: R = {A[3:0], A[6:4]};
            3'd5: R = {A[4:0], A[6:5]};
            3'd6: R = {A[5:0], A[6]};
            3'd7: R = A;  // Full rotate results in original
            default: R = A;
        endcase
    end
endmodule
```

ALU.v

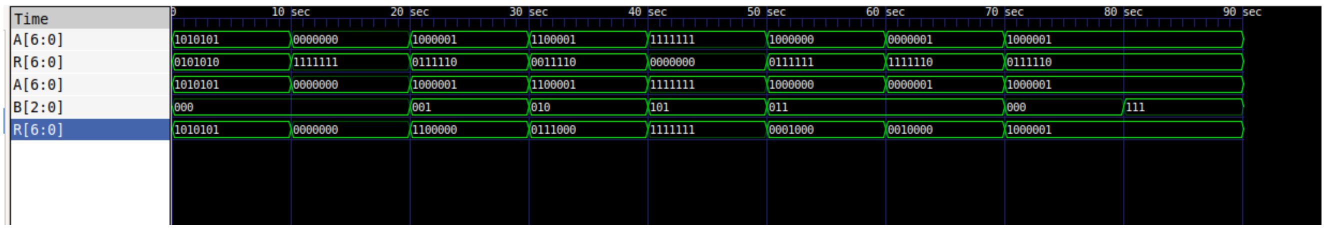
```
module ALU
(
    input wire [6:0] A, B,
    input wire [1:0] OP,         // 00 = NOT, 01 = ROR
    output reg [6:0] R,
    output wire CF,
    output wire SF,
    output wire ZF
);

    wire [6:0] R_NOT, R_ROR;

    // Submodules
    ALU_OP1 NOT1 (.A(A), .R(R_NOT));
    ALU_OP2 ROR1 (.A(A), .B(B[2:0]), .R(R_ROR));

    always @(*) begin
        case (OP)
            2'b00: R = R_NOT;
            2'b01: R = R_ROR;
            default: R = 7'b0000000;
        endcase
    end

    assign CF = (OP == 2'b01) ? A[B % 7] : 1'b0; // CF is
last bit rotated out
    assign SF = R[6];                              // Sign flag
(MSB)
    assign ZF = ~(|R);                             // Zero flag
endmodule
```

ALU_TESTBENCH.v

```
module ALU_TESTBENCH;
    reg [6:0] A, B;
    reg [1:0] OP;
    wire [6:0] R;
    wire CF, SF, ZF;

    ALU uut (
        .A(A),
        .B(B),
        .OP(OP),
        .R(R),
        .CF(CF),
        .SF(SF),
        .ZF(ZF)
    );

    initial begin
        $dumpfile("test7.vcd");
        $dumpvars(0, ALU_TESTBENCH);

        // Test NOT operation
        A = 7'b1010101; B = 7'b0000000; OP = 2'b00; #10;
        A = 7'b0000000; B = 7'b0000000; OP = 2'b00; #10;

        // Test Rotate Right by various amounts
        A = 7'b1000001; B = 7'b0000001; OP = 2'b01; #10; //
Rotate by 1
        A = 7'b1100001; B = 7'b0000010; OP = 2'b01; #10; //
Rotate by 2
        A = 7'b1111111; B = 7'b0000101; OP = 2'b01; #10; //
Rotate by 5
        A = 7'b1000000; B = 7'b0000011; OP = 2'b01; #10; //
Rotate by 3
        A = 7'b0000001; B = 7'b0000011; OP = 2'b01; #10; //
Rotate by 3
        A = 7'b1000001; B = 7'b0000000; OP = 2'b01; #10; //
Rotate by 0
        A = 7'b1000001; B = 7'b0000111; OP = 2'b01; #10; //
Rotate by 7 (same as original)

        $finish;
    end

    initial begin
        $monitor("Time=%0t A=%b B=%b OP=%b -> R=%b CF=%b
ZF=%b SF=%b", $time, A, B, OP, R, CF, ZF, SF);
    end
endmodule
```

RTL Timing Diagram:

Check List: Have you added all the timing diagrams of ALU_TESTBENCH?	YES



```
Time     0        10 sec   20 sec   30 sec   40 sec   50 sec   60 sec   70 sec   80 sec   90 sec
A[6:0]   1010101  0000000  1000001  1100001  1111111  1000000  0000001  1000001
R[6:0]   0101010  1111111  0111110  0011110  0000000  0111111  1111110  0111110
A[6:0]   1010101  0000000  1000001  1100001  1111111  1000000  0000001  1000001
B[2:0]   000               001      010      101      011               000      111
R[6:0]   1010101  0000000  1100000  0111000  1111111  0001000  0010000  1000001
```

Category: B
Word Size = 7bits
ALU Operations = NOT ,ROR

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

RTL synthesis summary

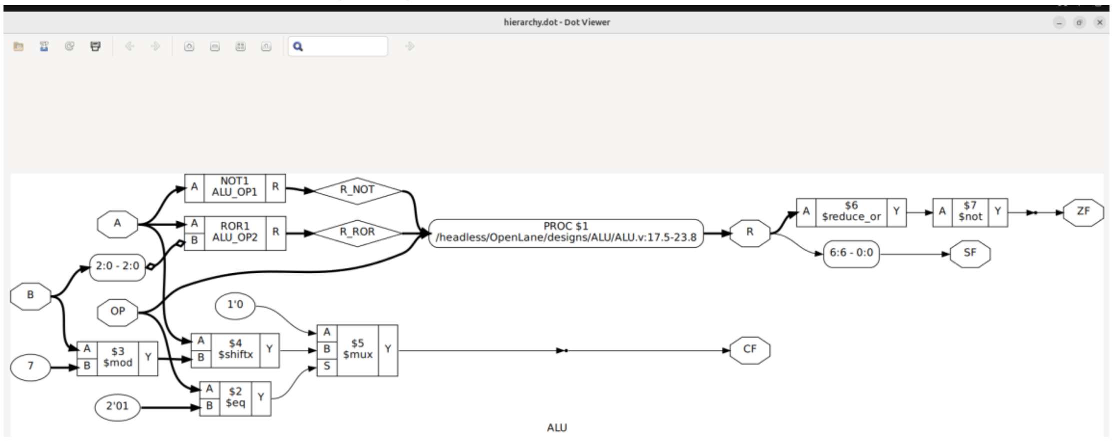
```
=== ALU ===

METRIC                                COUNT
   Number of wires:               134
   Number of wire bits:           153
   Number of public wires:          7
   Number of public wire bits:     26
   Number of ports:                 7
   Number of port bits:            26
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:               137
```

Standard cell usage in synthesized design

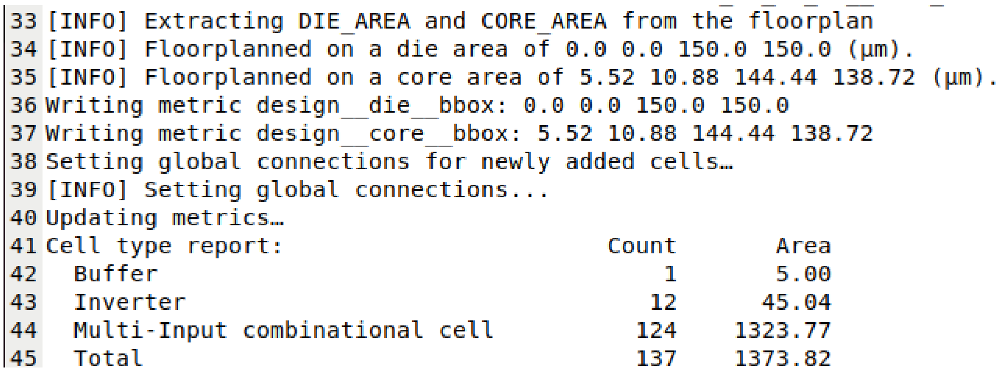
```
     sky130_fd_sc_hd__a2111oi_2      1
     sky130_fd_sc_hd__a211o_2        4
     sky130_fd_sc_hd__a21bo_2        2
     sky130_fd_sc_hd__a21o_2         5
     sky130_fd_sc_hd__a21oi_2        4
     sky130_fd_sc_hd__a221o_2        3
     sky130_fd_sc_hd__a221oi_2       3
     sky130_fd_sc_hd__a22o_2         1
     sky130_fd_sc_hd__a311o_2        1
     sky130_fd_sc_hd__a31o_2         4
     sky130_fd_sc_hd__a31oi_2        1
     sky130_fd_sc_hd__a32o_2         2
     sky130_fd_sc_hd__and2_2         4
     sky130_fd_sc_hd__and2_4         1
     sky130_fd_sc_hd__and3_2         4
     sky130_fd_sc_hd__and4b_2        1
     sky130_fd_sc_hd__buf_2          1
     sky130_fd_sc_hd__inv_2         12
     sky130_fd_sc_hd__mux2_1         3
```

```
sky130_fd_sc_hd__mux2_2        4
sky130_fd_sc_hd__mux4_2        8
sky130_fd_sc_hd__nand2_2       9
sky130_fd_sc_hd__nand2b_2      2
sky130_fd_sc_hd__nand3_2       2
sky130_fd_sc_hd__nand3b_2      1
sky130_fd_sc_hd__nor2_2        7
sky130_fd_sc_hd__nor2b_2       1
sky130_fd_sc_hd__nor3_2        1
sky130_fd_sc_hd__nor3b_2       1
sky130_fd_sc_hd__nor4_4        1
sky130_fd_sc_hd__nor4b_2       1
sky130_fd_sc_hd__o211a_2       8
sky130_fd_sc_hd__o211a_4       2
sky130_fd_sc_hd__o211ai_2      1
sky130_fd_sc_hd__o21a_2        5
sky130_fd_sc_hd__o21bai_2      1
sky130_fd_sc_hd__o22a_2        1
sky130_fd_sc_hd__o2bb2a_2      1
sky130_fd_sc_hd__o311a_2       1
sky130_fd_sc_hd__o31a_2        3
sky130_fd_sc_hd__o31ai_2       1
sky130_fd_sc_hd__o32ai_2       1
sky130_fd_sc_hd__or2_2         6
sky130_fd_sc_hd__or2_4         2
sky130_fd_sc_hd__or3_2         1
sky130_fd_sc_hd__or3_4         2
sky130_fd_sc_hd__or3b_2        1
sky130_fd_sc_hd__or4_4         1
sky130_fd_sc_hd__xnor2_2       2
sky130_fd_sc_hd__xor2_2        2
```

RTL synthesized design figure

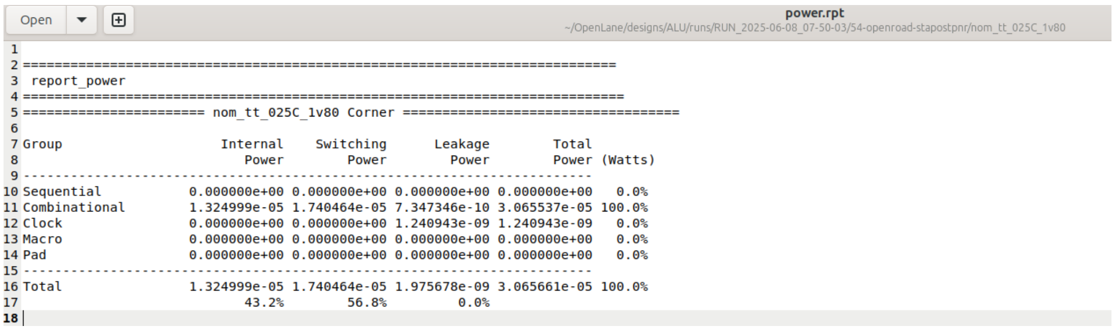


RTL Floorplan (130nm Skywater PDK with OpenLane toolchain)



```
[INFO] Extracting DIE_AREA and CORE_AREA from the floorplan
[INFO] Floorplanned on a die area of 0.0 0.0 150.0 150.0 (µm).
[INFO] Floorplanned on a core area of 5.52 10.88 144.44 138.72 (µm).
Writing metric design__die__bbox: 0.0 0.0 150.0 150.0
Writing metric design__core__bbox: 5.52 10.88 144.44 138.72
Setting global connections for newly added cells…
[INFO] Setting global connections...
Updating metrics…
Cell type report:                       Count      Area
  Buffer                                    1      5.00
  Inverter                                 12     45.04
  Multi-Input combinational cell          124   1323.77
  Total                                   137   1373.82
```

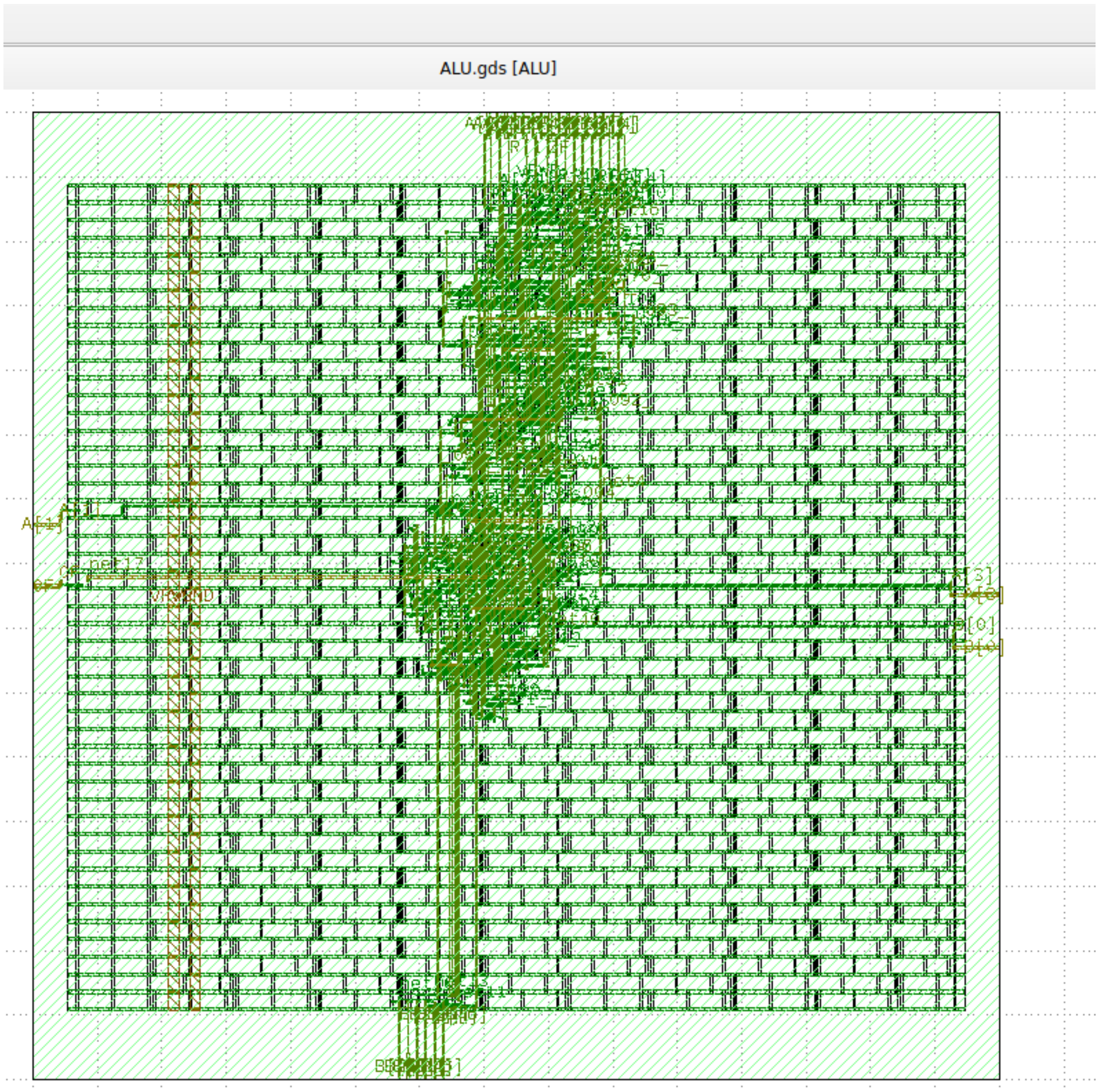
RTL Power Analysis (130nm Skywater PDK with OpenLane toolchain)



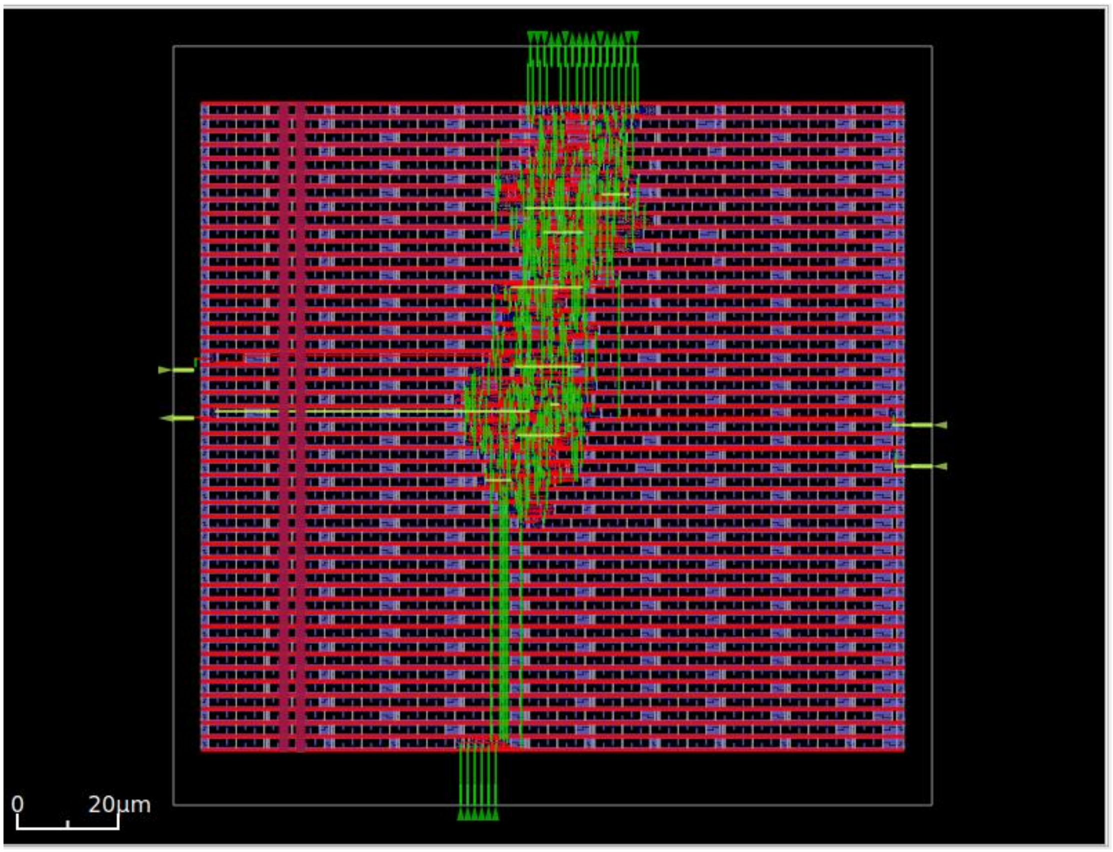
Open
power.rpt
~/OpenLane/designs/ALU/runs/RUN_2025-06-08_07-50-03/54-openroad-stapostpnr/nom_tt_025C_1v80

```

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

Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
------------------------------------------------------------------------
Sequential             0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00   0.0%
Combinational          1.324999e-05 1.740464e-05 7.347346e-10 3.065537e-05 100.0%
Clock                  0.000000e+00 0.000000e+00 1.240943e-09 1.240943e-09   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                  1.324999e-05 1.740464e-05 1.975678e-09 3.065661e-05 100.0%
                              43.2%        56.8%         0.0%
```

GDS Layout (130nm Skywater PDK with OpenLane toolchain)



Heatmap (130nm Skywater PDK with OpenLane toolchain)



Category: C
Word Size = 7bits
ALU Operations = NOT ,ROR

HDL Code:

Check List: Have you added all the modules written in verilog including CONTROLLER,TOP, TOP_TESTBENCH,CONTROLLER_TESTBENCH?	YES

TOP.v

```
module TOP(
    input wire clk, reset,
    output wire [6:0] result,
    output wire flag
);

wire [6:0] a, b;
wire [2:0] op;
wire [6:0] res;
wire f;
```

```

CONTROLLER controller(
    .clk(clk), .reset(reset),
    .a(a), .b(b), .op(op),
    .result(res),
    .flag(f)
);

CONTROLLER_TESTBENCH datapath(
    .a(a), .b(b), .op(op),
    .result(res), .flag(f)
);

assign result = res;
assign flag = f;

endmodule
```

TOP_CONTROLLER.v

```
`timescale 1ns/1ns

module TOP_TESTBENCH;
reg clk, reset;
wire [6:0] result;
wire flag;

TOP top (
    .clk(clk), .reset(reset),
    .result(result), .flag(flag)
);

initial begin
    clk = 0;
    forever #5 clk = ~clk;
end

initial begin
    $dumpfile("alu_fsm.vcd");
    $dumpvars(0,TOP_TESTBENCH);

    reset = 1;
    #10;
    reset = 0;

    #100;  // Run for a few clock cycles
    $finish;
end
endmodule
```

CONTROLLER.v

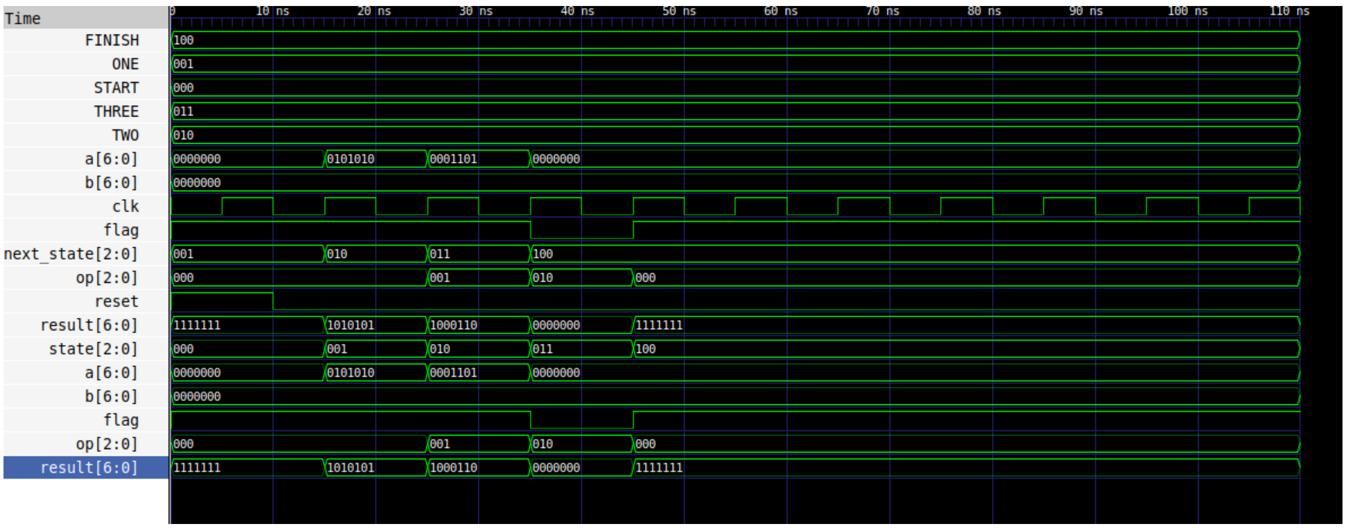
```
module CONTROLLER(
    input wire clk, reset,
    output reg [6:0] a, b,
    output reg [2:0] op,
    input wire [6:0] result,
    input wire flag
);

reg [2:0] state, next_state;

parameter START  = 3'b000,
          ONE    = 3'b001,
          TWO    = 3'b010,
          THREE  = 3'b011,
          FINISH = 3'b100;

always @(posedge clk or posedge reset) begin
    if (reset) state <= START;
    else state <= next_state;
end

always @(*) begin
    a = 7'd0; b = 7'd0; op = 3'd0;
    case (state)
        START:  next_state = ONE;
        ONE: begin
            a = 7'b0101010; op = 3'd0; // NOT
            next_state = TWO;
        end
        TWO: begin
            a = 7'b0001101; op = 3'd1; // ROTATE RIGHT
            next_state = THREE;
        end
        THREE: begin
            a = 7'd0; op = 3'd2;       // NOP or final test
case
            next_state = FINISH;
        end
        FINISH: next_state = FINISH;
        default: next_state = START;
    endcase
end

endmodule
```

CONTROLLER_TESTBENCH.v

```
module TOP(
    input wire clk, reset,
    output wire [6:0] result,
    output wire flag
);

wire [6:0] a, b;
wire [2:0] op;
wire [6:0] res;
wire f;

CONTROLLER controller(
    .clk(clk), .reset(reset),
    .a(a), .b(b), .op(op),
    .result(res),
    .flag(f)
);

CONTROLLER_TESTBENCH datapath(
    .a(a), .b(b), .op(op),
    .result(res), .flag(f)
);

assign result = res;
assign flag = f;

endmodule
```

RTL Timing Diagram:

Check List: Have you added all the timing diagrams of CONTROLLER_TESTBENCH, TOP_TESTBENCH?	YES


Time
0
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
FINISH
100
ONE
001
START
000
THREE
011
TWO
010
a[6:0]
0000000
0101010
0001101
0000000
b[6:0]
0000000
clk
flag
next_state[2:0]
001
010
011
100
op[2:0]
000
001
010
000
reset
result[6:0]
1111111
1010101
1000110
0000000
1111111
state[2:0]
000
001
010
011
100
a[6:0]
0000000
0101010
0001101
0000000
b[6:0]
0000000
flag
op[2:0]
000
001
010
000
result[6:0]
1111111
1010101
1000110
0000000
1111111