LDLAB_220405_002_team20_Yang Taehyun_2021-14284

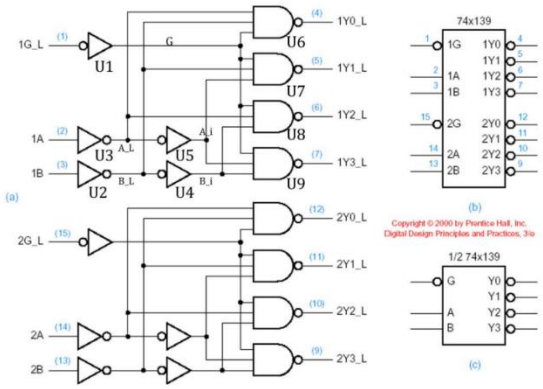
1. Lab

Implementing 74X139

- Truth Table

B	A	$Y0\_L$	$Y1\_L$	$Y2\_L$	$Y3\_L$
F	F	F	T	T	T
F	T	T	F	T	T
T	F	T	T	F	T
T	T	T	T	T	F

- Logic Diagram



Code for v74x139 _a.v

```
module v74x139h_a(
    input G_L,
    input A,
    input B,
    output [3:0] Y_L
    );

        // Structural
        wire N_A, N_B, N_G;

        not T1(N_G, G_L);
        not T2(N_A, A);
        not T3(N_B, B);

        nand T4(Y_L[0], N_G, N_A, N_B);

        nand T5(Y_L[1], N_G, A, N_B);
        nand T6(Y_L[2], N_G, N_A, B);
        nand T7(Y_L[3], N_G, A, B);
endmodule
```

Code for v74x139_b.v

```
module v74x139h_a(
    input G_L,
    input A,
    input B,
    output [3:0] Y_L
    );

        // Dataflow
        wire [1:0] sel;
        wire [3:0] out;

        assign sel = {B, A};
        assign Y_L = ~out;

        assign out =    (sel == 2'b00 && G_L == 1'b0) ? 4'b0001 :
                        (sel == 2'b01 && G_L == 1'b0) ? 4'b0010 :
                        (sel == 2'b10 && G_L == 1'b0) ? 4'b0100 :
                        (sel == 2'b11 && G_L == 1'b0) ? 4'b1000 :
                        4'b0000;
endmodule
```

Code of v74x139_c.v

```
module v74x139h_a(
    input G_L,
    input A,
    input B,
    output [3:0] Y_L
    );

        // Behavioral
        wire [1:0] sel;
        reg [3:0] out;

        assign sel = {B, A};
        assign Y_L = ~out;

        always@(G_L or sel) begin
                if(G_L == 1'b0) begin
                        case(sel)
                                2'b00 : out = 4'b0001;
                                2'b01 : out = 4'b0010;
                                2'b10 : out = 4'b0100;
                                2'b11 : out = 4'b1000;
                        endcase
                end
                else out = 4'b0000;
        end
endmodule
```

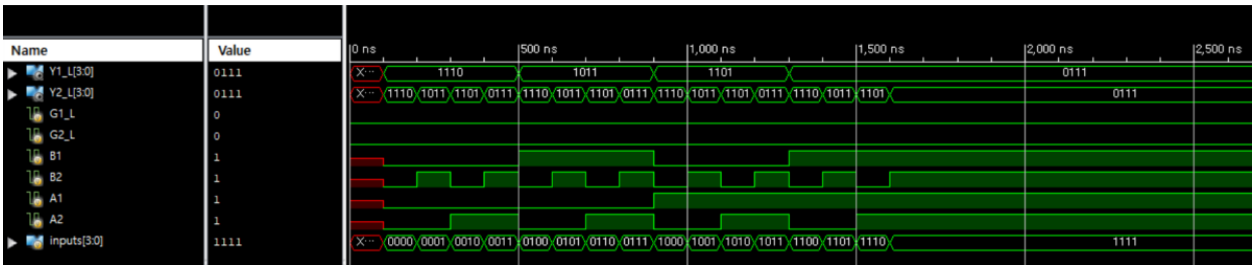
Code of v74x139.v

```
module v74x139(
    input G1_L,
    input G2_L,
    input B1,
    input B2,
    input A1,
    input A2,
    output [3:0] Y1_L,
    output [3:0] Y2_L
    );

          v74x139h_a T1(.G_L(G1_L), .A(A1), .B(B1), .Y_L(Y1_L));
          v74x139h_a T2(.G_L(G2_L), .A(A2), .B(B2), .Y_L(Y2_L));


endmodule
```

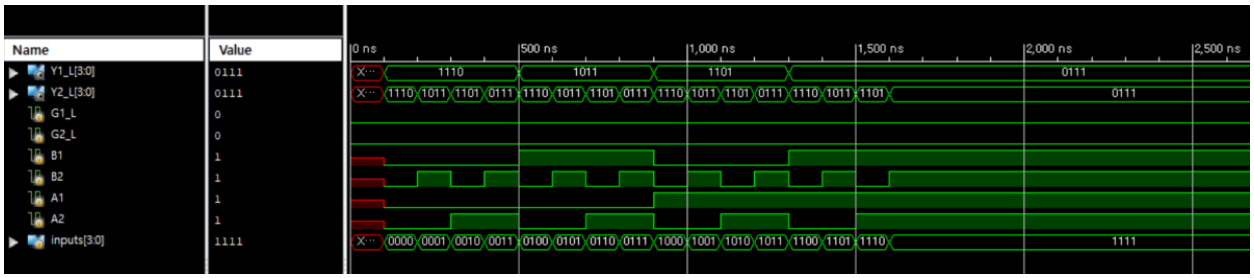
Code for v74x139h_test.v

```
module v74x139_test;

        // Inputs
        reg G1_L;
        reg G2_L;
        reg B1;
        reg B2;
        reg A1;
        reg A2;

        reg [3:0] inputs;

        // Outputs
        wire [3:0] Y1_L;
        wire [3:0] Y2_L;



        // Instantiate the Unit Under Test (UUT)
        v74x139 uut (
                .G1_L(G1_L),
                .G2_L(G2_L),
                .B1(B1),
                .B2(B2),
                .A1(A1),
                .A2(A2),
                .Y1_L(Y1_L),
                .Y2_L(Y2_L)
        );

        initial begin
                // Initialize Inputs
                G1_L = 0;
                G2_L = 0;
                B1 = 0;
                B2 = 0;
                A1 = 0;
                A2 = 0;

                assign {A1, B1, A2, B2} = inputs;

                // Wait 100 ns for global reset to finish
                #100;

                // Add stimulus here
                for(inputs = 0; inputs < 4'b1111; inputs = inputs+1)begin
                        #100; G1_L = 0;
                end

        end

endmodule
```

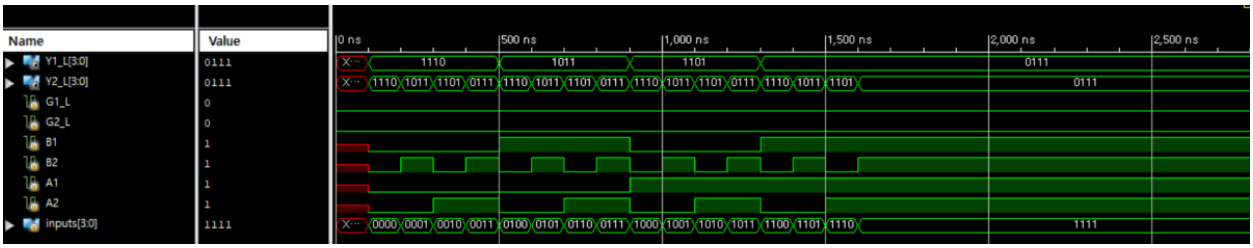
- Simulation Result of v74x139_a.v (structural Description)



- Simulation Result of v74x139_b.v (data-flow style description)



- Simulation Result of v74x139_c.v(behavioral description)



1.2 Implementing a 3 to 8 decoder

Code for v3x8

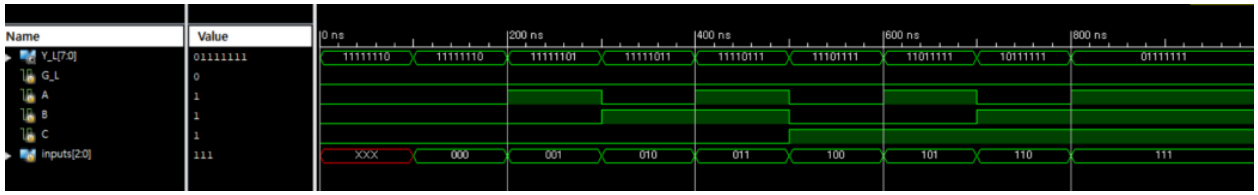
```
module v3x8(
    input G_L,
    input A,
    input B,
    input C,
    output [7:0] Y_L
    );

        wire G;
        wire G1_L, G2_L, N_C;
        wire [3:0] Y1_L, Y2_L;

        assign Y_L = {Y2_L, Y1_L};

        not T1(N_C, C);
        not T2(G, G_L);

        nand T3(G1_L, G, N_C);
        nand T4(G2_L, G, C);

        v74x139 T5(.G1_L(G1_L), .G2_L(G2_L),
        .A1(A), .A2(A), .B1(B), .B2(B), .Y1_L(Y1_L), .Y2_L(Y2_L));
endmodule
```

Code for v3x8_test

```
module v3x8_test;

        // Inputs
        reg G_L;
        reg A;
        reg B;
        reg C;

        reg [2:0] inputs;

        // Outputs
        wire [7:0] Y_L;

        // Instantiate the Unit Under Test (UUT)
        v3x8 uut (
                .G_L(G_L),
                .A(A),
                .B(B),
                .C(C),
                .Y_L(Y_L)
        );

```

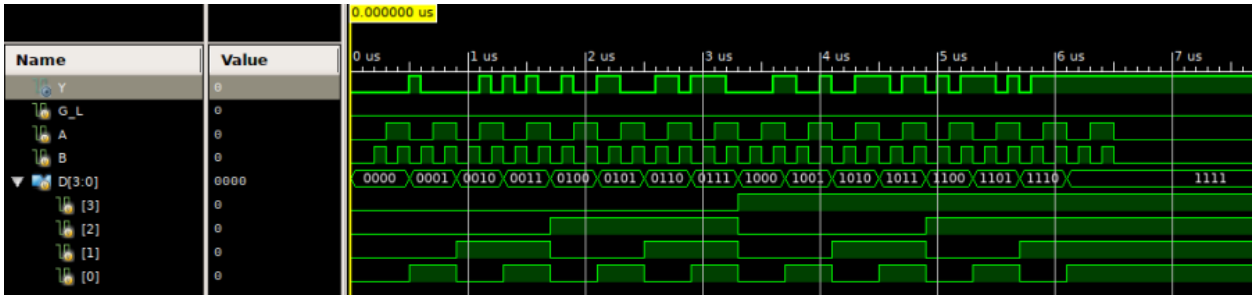
```
23          initial begin
24                  // Initialize Inputs
25                  G_L = 0;
26                  A = 0;
27                  B = 0;
28                  C = 0;
29
30                  // Wait 100 ns for global reset to finish
31                  #100;
32
33                  assign {C, B, A} = inputs;
34
35                  for (inputs = 0; inputs < 3'b111; inputs = inputs+1) begin
36                          #100 G_L = 0;
37                  end
38          end
39  endmodule
```

Simulation Result

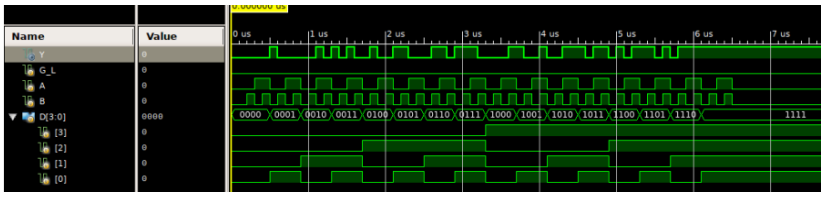


2. Implementing a 4 to 1 MUX

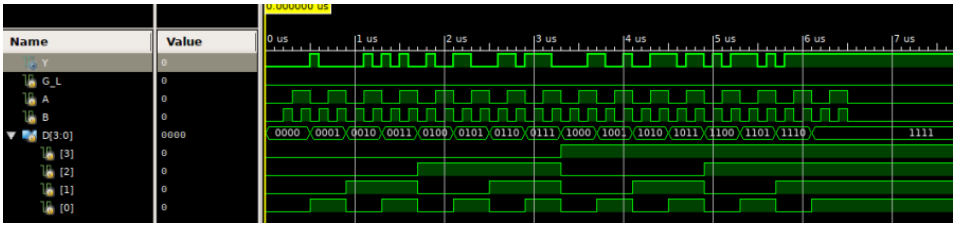
result of v74x153 a.v



result of v74x153 b.v



Result of v74x153 c.



3. Discussion.

Structural Description- The biggest pro for a structural description is that we can just implement it straight from a logical circuit. It is very intuitive, however the moment the logical circuit becomes more complex, it might be hard to implicate it and the code may also become a bit messy

Data flow style description- We can simply convert the truth table onto the data flow style description sheet. We can use assign to show the relationship between the inputs and outputs. Relatively simple and intuitive just like the structural description but once input increases, the possible outcomes for truth table also exponentially increases thus making it a bit difficult to implicate complex logics.

Behavioral Description- Implicating based on time logical control statements and block/nonblocking functions. This makes it easier to implicate larger and more complex logics. However, since it is a chronologically controlled description, there might be unexpected outputs and it is hard to alter errors for this problem.