**CSE460: VLSI Design**

**Lab Report 01**

1. Design a 4 to 1 MUX using Verilog HDL and verify using timing diagram.

**Code:**

module mux\_4\_1 (

input in1, in2, in3, in4,

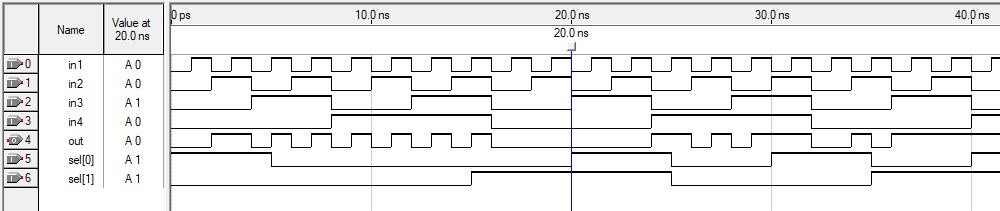
input [1:0] sel,

output out);

assign out = sel[1] ? (sel[0] ? in4 : in3) : (sel[0] ? in2 : in1);

endmodule

**Output:**



1. Design a priority encoder (3>1>0>2) using Verilog HDL and verify using timing diagram.

**Code:**

module priority\_encoder (

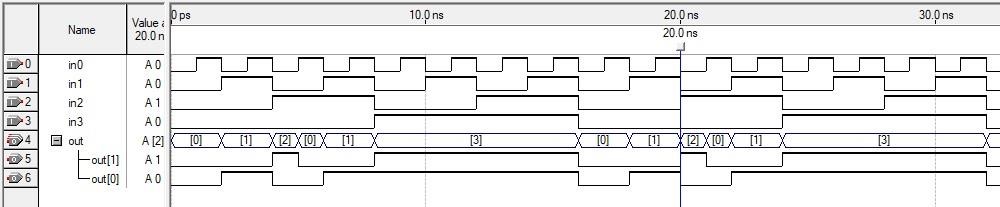
input in0, in1, in2, in3,

output[1:0] out);

assign out = in3 ? 3 : (in1 ? 1 : (in0 ? 0 : (in2 ? 2 : 0 ) ) );

endmodule

**Output:**



1. Design a 2 to 4 Decoder using Verilog HDL and verify using timing diagram.

**Code:**

module decoder\_2\_4 (

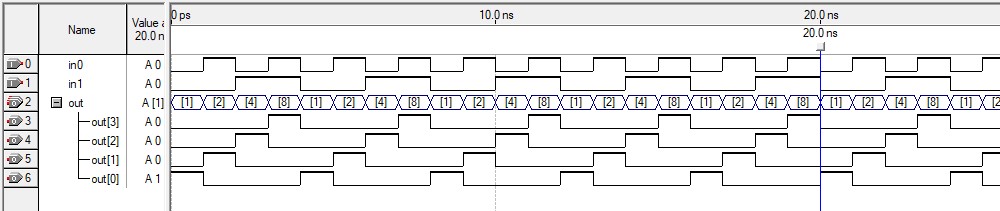
input in0, in1,

output [3:0] out );

assign out = in1 ? (in0 ? 8 : 4) : (in0 ? 2 : 1);

endmodule

**Output:**



1. Design a 4 bit adder-subtractor using Verilog HDL and verify using timing diagram.

**Code:**

module

adder\_subtractor(A0,A1,A2,A3,B0,B1,B2,B3,S0,S1,S2,S3,C0,C4,S);

input A0, A1, A2, A3;

input B0, B1, B2, B3;

input C0;

output S0, S1, S2, S3;

output C4, S;

wire A0, A1, A2, A3;

wire B0, B1, B2, B3;

wire C0;

wire S0, S1, S2, S3;

wire C4, S;

wire C1, C2, C3;

assign X0 = C0^B0;

assign X1 = C0^B1;

assign X2 = C0^B2;

assign X3 = C0^B3;

assign S = C3^C4;

fulladder Y0(C0,X0,A0,S0,C1);

fulladder Y1(C1,X1,A1,S1,C2);

fulladder Y2(C2,X2,A2,S2,C3);

fulladder Y3(C3,X3,A3,S3,C4);

endmodule

module fulladder(a,b,cin,sum,cout);

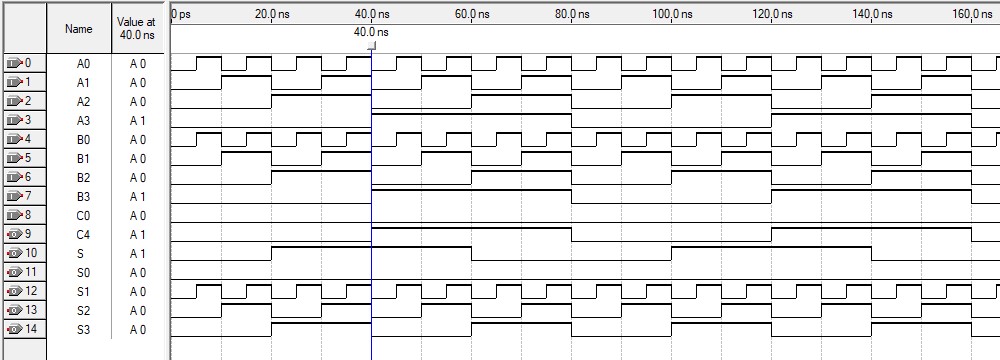
input a,b,cin;

output sum,cout;

assign {cout,sum} = a+b+cin;

endmodule

**Output:**



5. Write down the behavioral representation of Verilog HDL code 1 to 4 Demultiplexer and Up-Down Counter.

**1 to 4 Demultiplexer:**

**Code:**

module demultiplexer\_1\_4 (in, s0, s1, o1, o2, o3, o4);

output o1, o2, o3, o4;

input in;

input s0;

input s1;

assign o1 = in & (~s0) & (~s1);

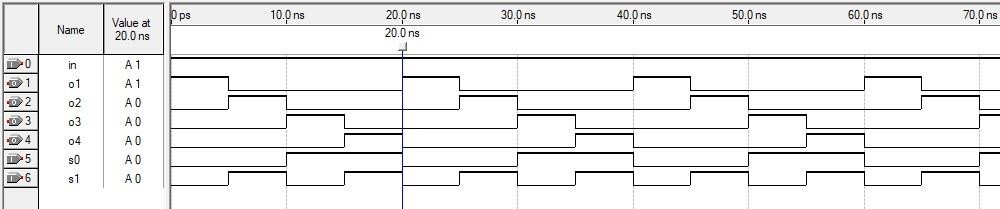
assign o2 = in & (~s0) & s1;

assign o3 = in & s0 & (~s1);

assign o4 = in & s0 & s1;

endmodule

**Output:**



**Up-Down Counter:**

**Code:**

module up\_down\_counter ( clk, up, out);

input clk, up; output[3:0] out;

reg [3:0] out = 4'b0000;

|  |
| --- |
|  |

always @(posedge clk) if(up)

begin

out = out + 4'b0001;

end

else

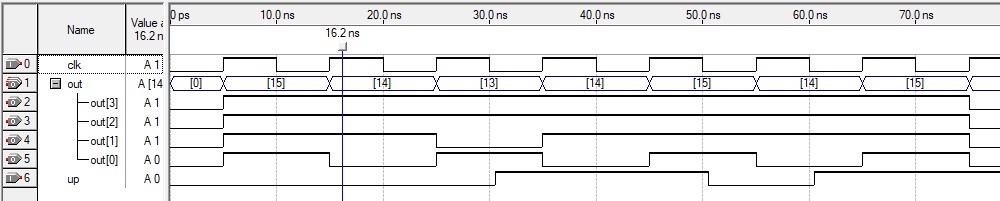
begin

out = out - 4'b0001;

end

endmodule

**Output:**



6. Write down the behavioral representation of Verilog HDL code for Ring Counter and Johnson Counter.

**Ring Counter:**

**Code:**

module ring\_counter (out, clk, clr);

input clk, clr;

output [3:0] out;

reg [3:0] out\_t;

always @(posedge clk)

if(clr==1)

out\_t <= 4'b1000;

else

begin

out\_t[3] <= out\_t[0];

out\_t[2] <= out\_t[3];

out\_t[1] <= out\_t[2];

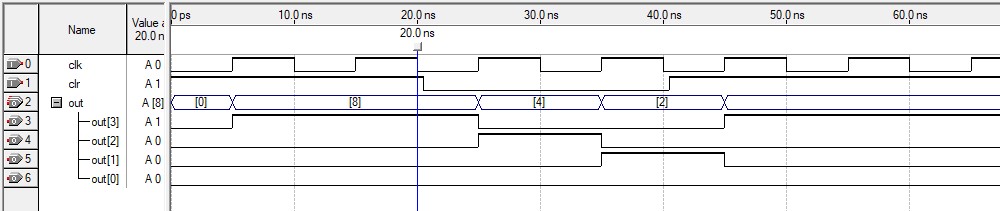
out\_t[0] <= out\_t[1];

end

assign out = out\_t;

endmodule

**Output:**



**Johnson Counter:**

**Code:**

module johnson\_counter(clk, clr, out);

input clk, clr;

output [3:0] out;

reg [3:0] out\_t;

always @(posedge(clk) or posedge(clr))

begin

if(clr == 1'b1)

begin

out\_t = 4'b0000;

end

else if(clk == 1'b1)

begin

out\_t = {out\_t[2:0],~out\_t[3]};

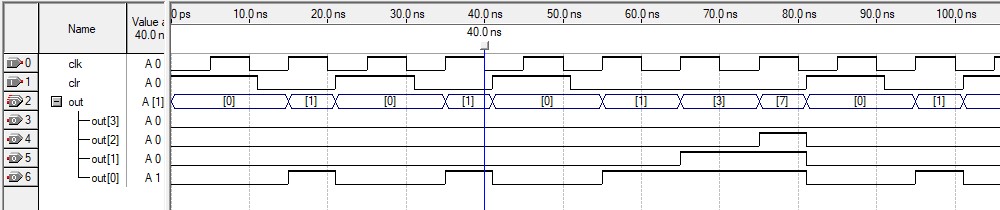
end

end

assign out = out\_t;

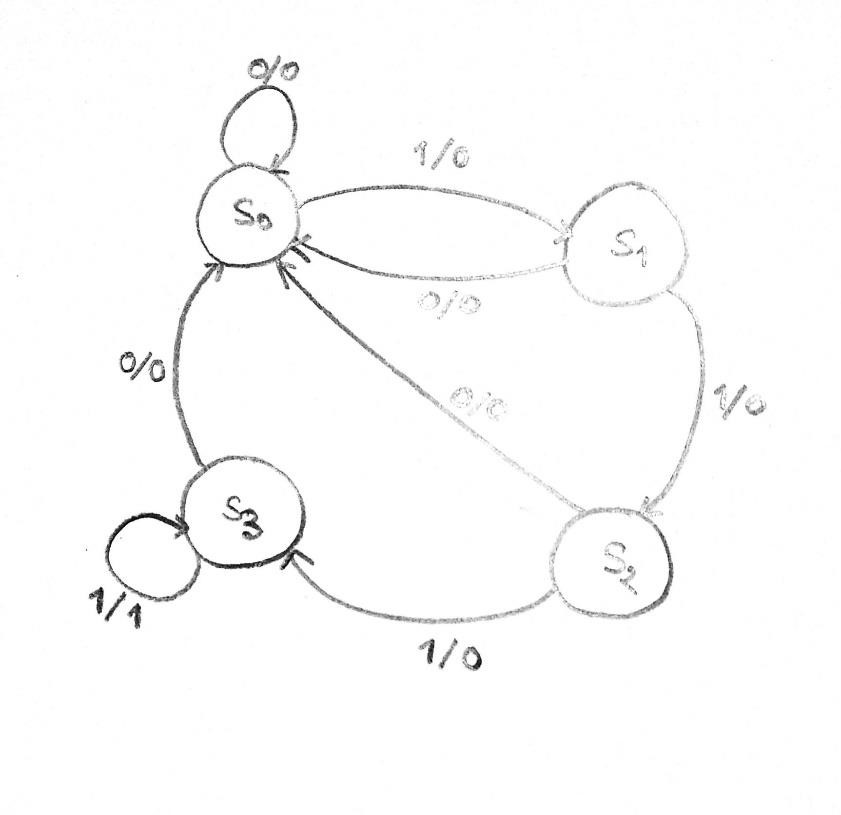
endmodule

**Output:**



7. A sequential circuit has two inputs, w1 and w2, and an output, z. Its function is to compare the input sequences on the two inputs. If w1 = w2 during any four consecutive clock cycles, the circuit produces z = 1; otherwise, z = 0. For example

w1 : 0 1 1 0 1 1 1 0 0 0 1 1 0 w2 : 1 1 1 0 1 0 1 0 0 0 1 1 1 z : 0 0 0 0 1 0 0 0 0 1 1 1 0 **State Diagram:**



**Code:**

module prob\_7 (Clock, Resetn, w1, w2, z);

input Clock, Resetn, w1, w2;

output reg z;

reg [1:0]y, Y;

parameter [2:0] S0 = 0, S1 = 1, S2 = 2, S3 = 3;

assign w = w1~^w2;

always @(w, y)

case (y)

S0: if (w)

begin

z = 0;

Y = S1;

end

else

begin

z = 0;

Y = S0;

end

S1: if (w)

begin

z = 0;

Y = S2;

end

else

begin

z = 0;

Y = S0;

end

S2: if (w)

begin

z = 0;

Y = S3;

end

else

begin

z = 0;

Y = S0;

end

S3: if (w)

begin

z = 1;

Y = S3;

end

else

begin

z = 0;

Y = S0;

end

default: begin z=1'bx; Y=3'bxxx; end

endcase

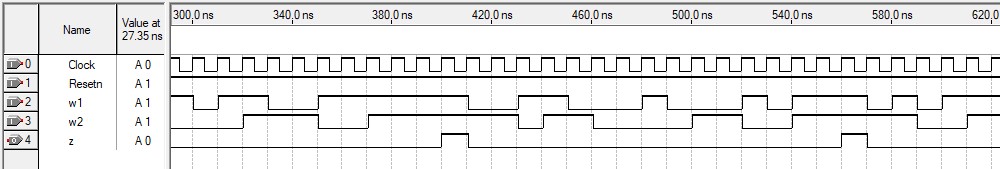
always @(negedge Resetn, posedge Clock)

if (Resetn == 0) y <= S0;

else y <= Y;

endmodule

**Output:**

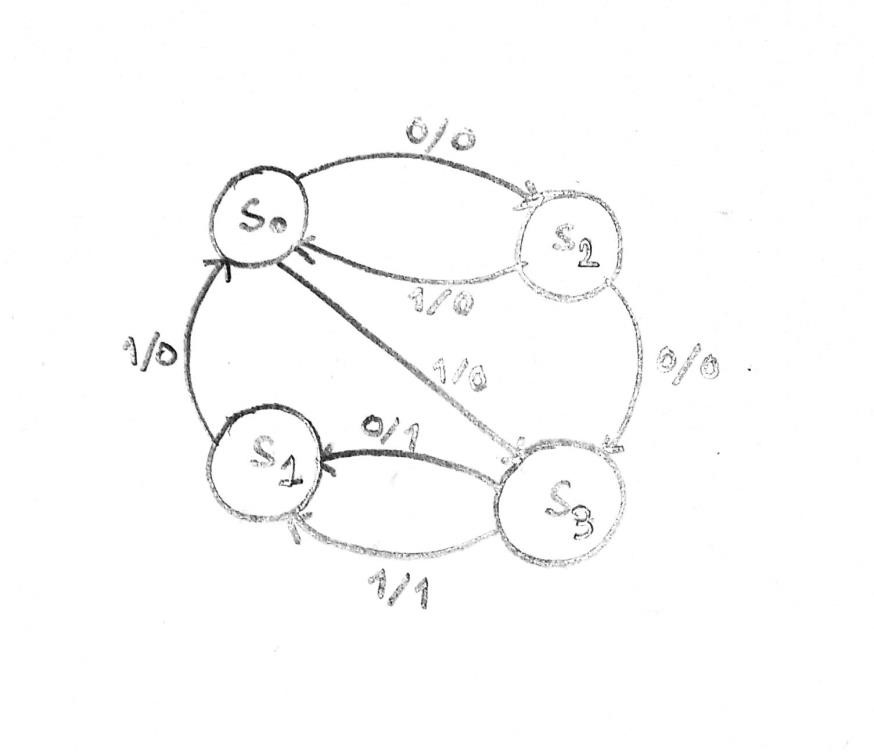


8. An FSM is defined by the state-assigned table in Figure below.

|  |  |  |  |
| --- | --- | --- | --- |
| Present State | Input (w=0) | Input (w=1) | Output |
| S0 | S2 | S3 | 0 |
| S1 | S1 | S0 | 0 |
| S2 | S3 | S0 | 0 |
| S3 | S1 | S1 | 1 |

Draw the state diagram and implement the system using Verilog code

**State Diagram:**



**Code:**

module prob\_8 (Clock, Resetn, w, z);

input Clock, Resetn, w;

output reg z;

reg [1:0]y, Y;

parameter [1:0] S0 = 0, S1 = 1, S2 = 2, S3 = 3;

always @(w, y)

case (y)

S0: if (w)

begin

z = 0;

Y = S3;

end

else

begin

z = 0;

Y = S2;

end

S1: if (w)

begin

z = 0;

Y = S0;

end

else

begin

z = 0;

Y = S1;

end

S2: if (w)

begin

z = 0;

Y = S0;

end

else

begin

z = 0;

Y = S3;

end

S3: if (w)

begin

z = 1;

Y = S1;

end

else

begin

z = 1;

Y = S1;

end

default: begin z=1'bx; Y=3'bxxx; end

endcase

always @(negedge Resetn, posedge Clock)

if (Resetn == 0) y <= S0;

else y <= Y;

endmodule

**Output:**

