College of Engineering, Pune

Dept. of Electronics and Telecommunication Engineering

Course **: RTL Simulation and Synthesis Lab (FY VLSI & ES)**

Year : 2020-21

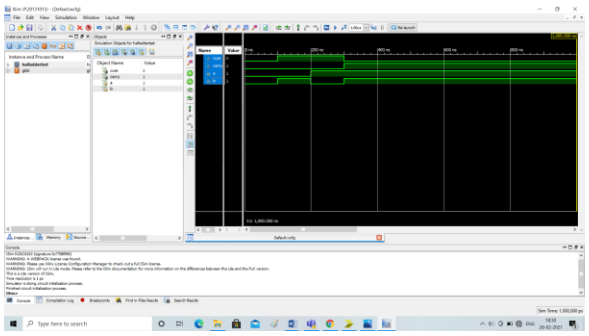
RTL Lab 1,2,3 Report :

Mis : 122035014

Name : Phalke Swapnil Hanmant

Question1:Half Adder

```
module halfadd( a,b,sum,carry);
input a,b;
output sum,carry;
assign sum=a^b;
assign carry=a&b;
endmodule
```



```
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

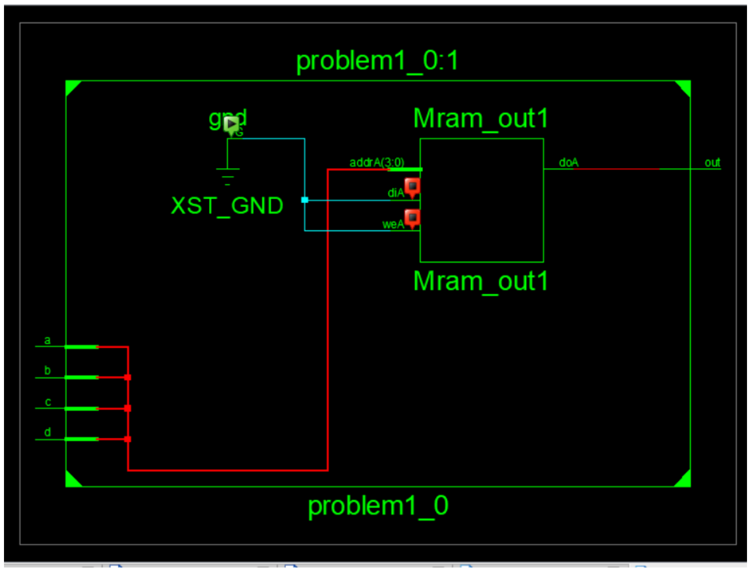
Sim Time: 1,000,000 ps

Question 2:

1.**Draw TT and simplify using k-maps if a 4 input function has conditions that output is high if inputs has exactly 2 zeros or exactly 2 ones or exactly 3 ones**

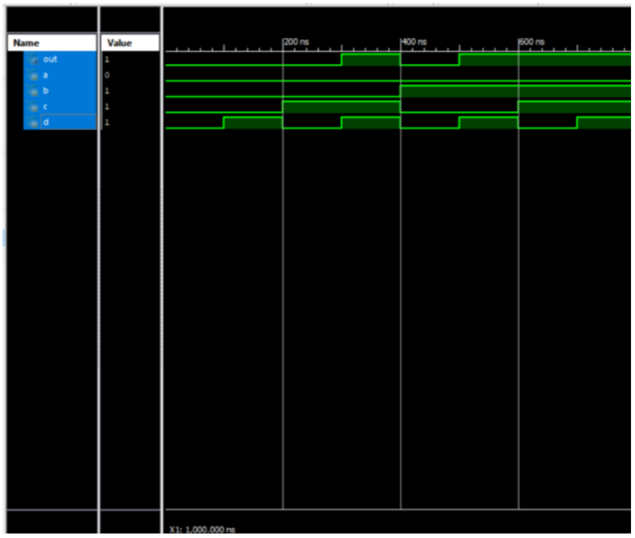
- **Behavioural**

```
module problem1_0(a,b,c,d,out);
input a,b,c,d;
output reg out;
reg [3:0]q;

always@(*)
begin
q[0]=a;
q[1]=b;
q[2]=c;
q[3]=d;
case(q)
4'b0000 : out = 0;
4'b0001 : out = 0;
4'b0010 : out = 0;
4'b0011 : out = 1;
4'b0100 : out = 0;
4'b0101 : out = 1;
4'b0110 : out = 1;
4'b0111 : out = 1;
4'b1000 : out = 0;
4'b1001 : out = 1;
4'b1010 : out = 1;
4'b1011 : out = 1;
4'b1100 : out = 1;
4'b1101 : out = 1;
4'b1110 : out = 1;
4'b1111 : out = 0;
default: out = 0;
endcase
end
endmodule
```



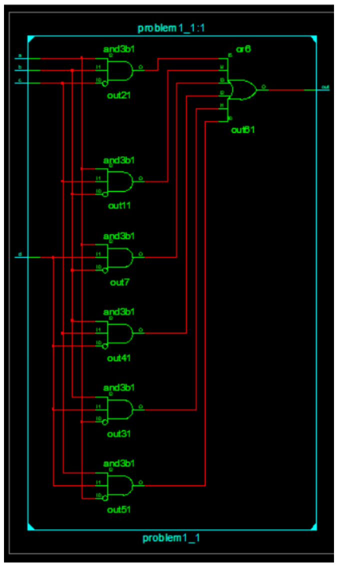
Testbench Code and Simulation Result:

```
module tesst;
// Inputs
            reg a;
            reg b;
            reg c;
            reg d;
            // Outputs
            wire out;
// Instantiate the Unit Under Test (UUT)
            problem1_0 uut (
                        .a(a), .b(b), .c(c), .d(d), .out(out));
    initial begin
                        a = 0;
                        b = 0;
                        c = 0;
                        d = 0;
#100;
     a = 0;b = 0;c = 0;d = 1;
#100;
                        a = 0;b = 0;c = 1;d = 0;
#100;
                        a = 0;b = 0;c = 1;d = 1;
#100;
                     a = 0;b = 1;c = 0;d = 0;
#100;
                        a = 0;b = 1;c = 0;d = 1;
#100;
                        a = 0;b = 1;c = 1;d = 0;
#100;
                        a = 0;b = 1;c = 1;d = 1;
                        end
     endmodule
```



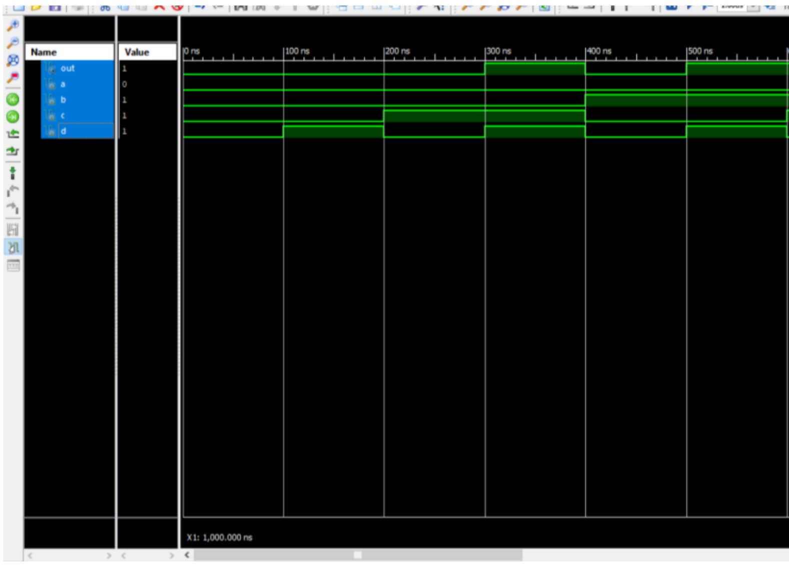
- **Structual**

```
module problem1_1(
    a,b,c,d,out);
            input a,b,c,d;
            output out;
            wire q,w,e,r;
            wire a1,a2,a3,a4,a5,a6;
            not(q,a);
            not(w,b);
            not(e,c);
            not(r,d);
            and(a1,c,d,q);
            and(a2,b,d,q);
            and(a3,b,c,r);
            and(a4,a,d,w);
            and(a5,a,w,c);
            and(a6,a,b,e);
            or(out,a1,a2,a3,a4,a5,a6);
endmodule
```



Testbench Code and Simulation Result:

```
module tesst;
// Inputs
            reg a;
            reg b;
            reg c;
            reg d;
            // Outputs
            wire out;
// Instantiate the Unit Under Test (UUT)
            problem1_1 uut (
                        .a(a),
                        .b(b),
                        .c(c),
                        .d(d),
                        .out(out)
            );
    initial begin
                        // Initialize Inputs
                        a = 0;
                        b = 0;
                        c = 0;
                        d = 0;
            #100;
     a = 0;b = 0;c = 0;d = 1;
#100;
                        a = 0;b = 0;c = 1;d = 0;
#100;
                        a = 0;b = 0;c = 1;d = 1;
#100;
                    a = 0;b = 1;c = 0;d = 0;
#100;
                        a = 0;b = 1;c = 0;d = 1;
#100;
                        a = 0;b = 1;c = 1;d = 0;
#100;
                        a = 0;b = 1;c = 1;d = 1;
                        end
     endmodule
```

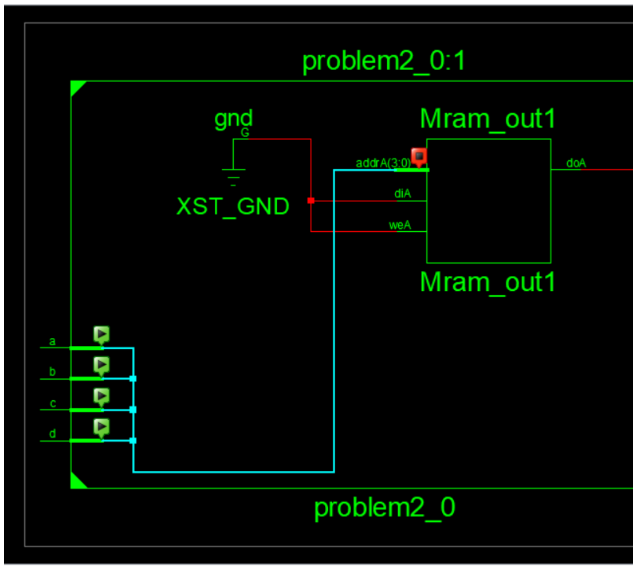


2. In a factory, if relay A is ON then indicator glows except condition that if both B and C are ON, and if A is OFF then indicator glows only when relay D is ON. Find the TT and simplify using k-map.

- **Behavioral**

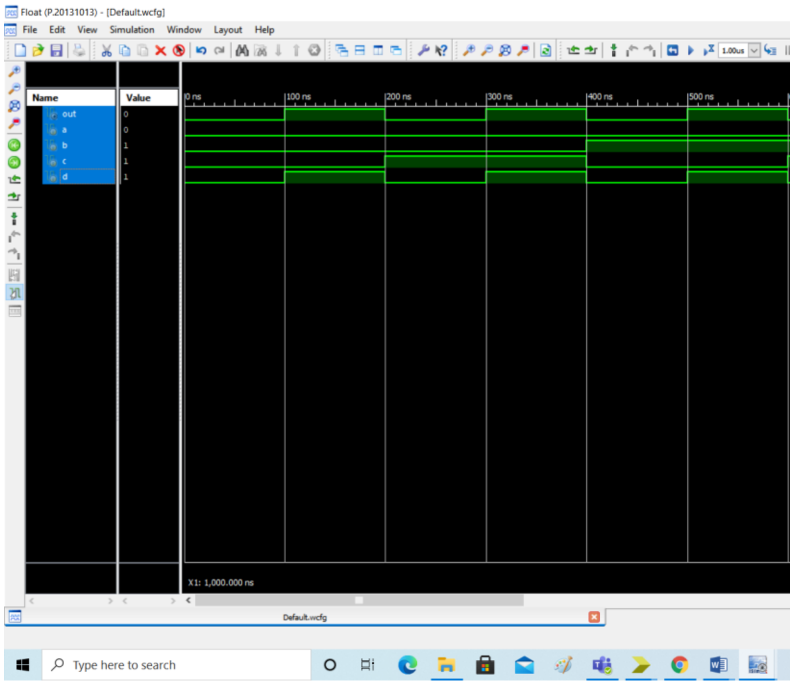
```
module problem2_0(a,b,c,d,out);
input a,b,c,d;
output reg out;
reg [3:0]q;
```

```
always@(*)
begin
q[0]=a;
q[1]=b;
q[2]=c;
q[3]=d;
case(q)
4'b0000 : out = 0;
4'b0001 : out = 1;
4'b0010 : out = 0;
4'b0011 : out = 1;
4'b0100 : out = 0;
4'b0101 : out = 1;
4'b0110 : out = 0;
4'b0111 : out = 1;
4'b1000 : out = 1;
4'b1001 : out = 1;
4'b1010 : out = 1;
4'b1011 : out = 1;
4'b1100 : out = 1;
4'b1101 : out = 1;
4'b1110 : out = 0;
4'b1111 : out = 0;
default: out = 0;
endcase
end
endmodule
```



Testbench Code and Simulation Result:

```
module tesst;
// Inputs
		reg a;
		reg b;
		reg c;
		reg d;
		// Outputs
		wire out;
// Instantiate the Unit Under Test (UUT)
		Problem2_0 uut (
				.a(a),
				.b(b),
				.c(c),
				.d(d),
				.out(out)
		);
    initial begin
				// Initialize Inputs
						a = 0;
						b = 0;
						c = 0;
						d = 0;
				#100;
      a = 0;b = 0;c = 0;d = 1;
#100;
						a = 0;b = 0;c = 1;d = 0;
#100;
						a = 0;b = 0;c = 1;d = 1;
#100;
					a = 0;b = 1;c = 0;d = 0;
#100;
						a = 0;b = 1;c = 0;d = 1;
#100;
						a = 0;b = 1;c = 1;d = 0;
#100;
						a = 0;b = 1;c = 1;d = 1;
						end
    endmodule
```



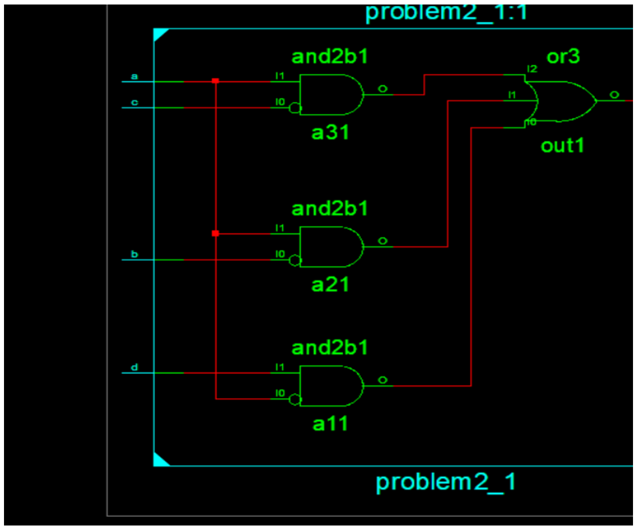
Name	Value
out	0
a	0
b	1
c	1
d	1

X1: 1,000.000 ns

- **Structual**

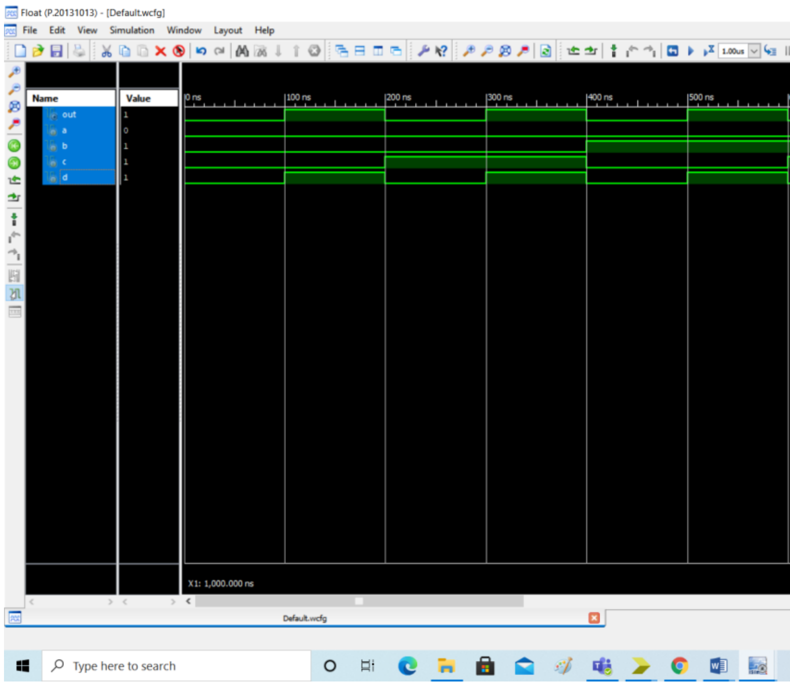
```
module problem2_1(
    a,b,c,d,out);

		input a,b,c,d;
		output out;
		wire q,w,e,r;
		wire a1,a2,a3;

		not(q,a);
		not(w,b);
		not(e,c);
		not(r,d);
		and(a1,q,d);
		and(a2,a,w);
		and(a3,a,e);
  or(out,a1,a2,a3);
```

endmodule



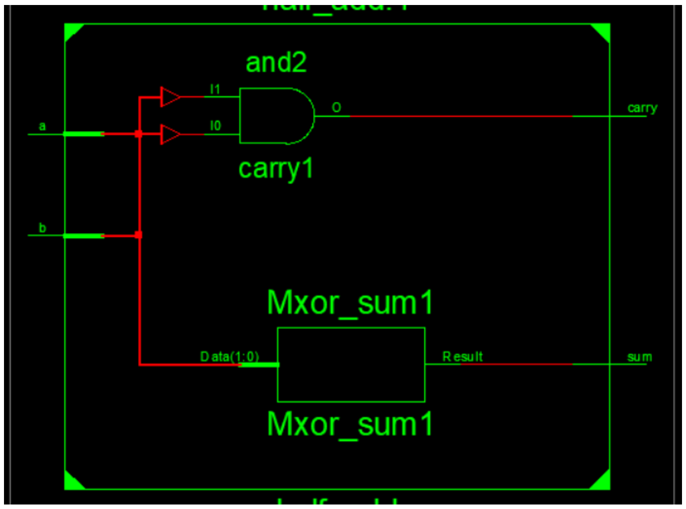
Testbench Code and Simulation Result:

```
module tesst;
// Inputs
                reg a;
                reg b;
                reg c;
                reg d;
                // Outputs
                wire out;
// Instantiate the Unit Under Test (UUT)
                Problem2_1 uut (
                                .a(a),
                                .b(b),
                                .c(c),
                                .d(d),
                                .out(out)
                );
     initial begin
                                // Initialize Inputs
                                a = 0;
                                b = 0;
                                c = 0;
                                d = 0;
#100;
      a = 0;b = 0;c = 0;d = 1;
#100;
                                a = 0;b = 0;c = 1;d = 0;
#100;
                                a = 0;b = 0;c = 1;d = 1;
#100;
                           a = 0;b = 1;c = 0;d = 0;
#100;
                                a = 0;b = 1;c = 0;d = 1;
#100;
                                a = 0;b = 1;c = 1;d = 0;
#100;
                                a = 0;b = 1;c = 1;d = 1;
                                end
      endmodule
```



Half adder

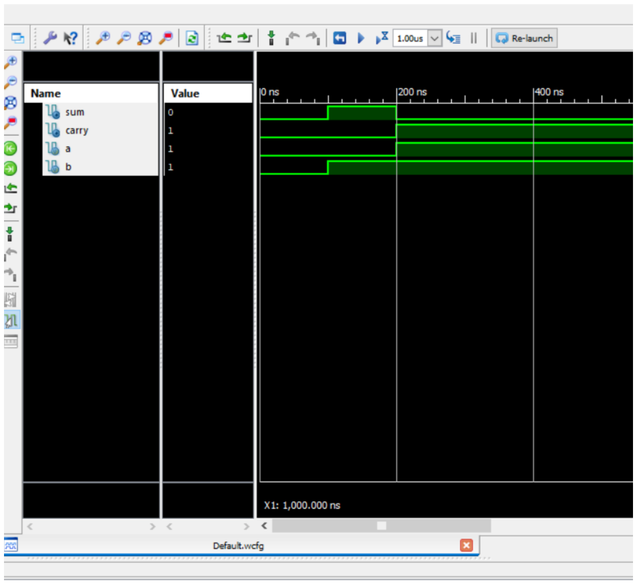
```
module half_add(
                input a,b,
                output sum,carry
                );

                xor(sum,a,b);
                and(carry,a,b);
Endmodule
```



Testbench Code and Simulation Result:

```
module half_add_tb;
                // Inputs
                reg a;
                reg b;

                // Outputs
                wire sum;
                wire carry;

                // Instantiate the Unit Under Test (UUT)
```

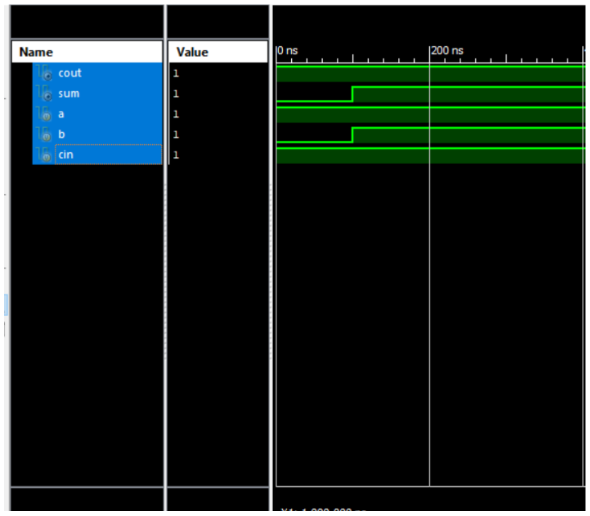
```
	half_add uut (
		.a(a),
		.b(b),
		.sum(sum),
		.carry(carry)
	);
	initial begin
		// Initialize Inputs
		a = 0;
		b = 0;
		// Wait 100 ns for global reset to finish
		#100;
		a=0;
		b=1;
		#100
		a=1;
		b=1;
	end
endmodule
```

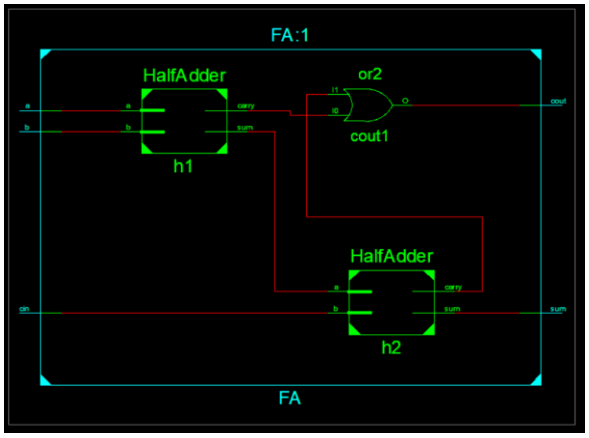


Name	Value
sum	0
carry	1
a	1
b	1

Question4:FA using HA

```
module FA( a,b,cin,cout,sum
    );
input a,b,cin;
output cout,sum;
wire x,y,z;
HalfAdder h1(a,b,x,y);
HalfAdder h2(x,cin,sum,z);
assign cout=y|z;
endmodule


module HalfAdder( a,b,sum,carry
    );
input a,b;
output sum,carry;
assign sum=a^b;
assign carry=a&b;

endmodule
```

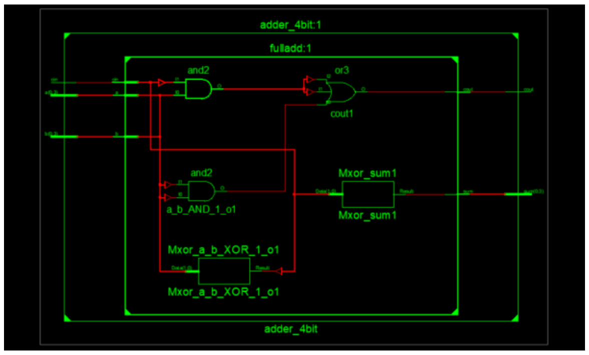




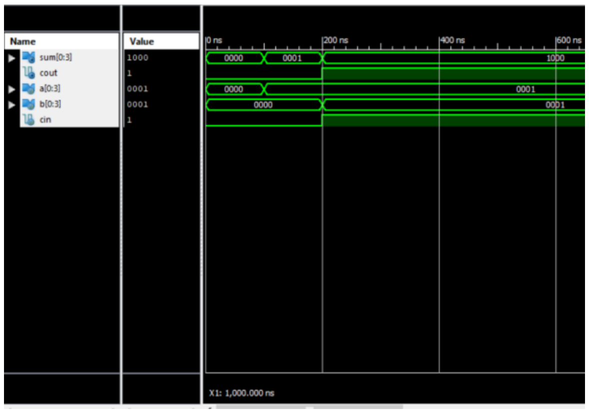
```
        initial begin
    a = 1;
                b = 0;
                cin = 1;
    #100;
                a = 1;
                b = 1;
                cin = 1;
    #100;
end
```

Question 5:4 bit ripple carry adder

```
module adder_4bit( a,b,sum,cin,cout
    );
input[0:3] a,b;
input cin;
output [0:3]sum;
output cout;
wire [1:3]x;

fulladd f1(a[0],b[0],cin,x[1],sum[0]);
fulladd f2(a[1],b[1],x[1],x[2],sum[1]);
fulladd f3(a[2],b[2],x[2],x[3],sum[2]);
fulladd f4(a[3],b[3],x[3],cout,sum[3]);
endmodule

module fulladd( a,b,cin,cout,sum
    );
input a,b,cin;
output cout,sum;

assign sum=a^b^cin;
assign cout=(a&b)|(b&cin)|(a&cin);

endmodule
```

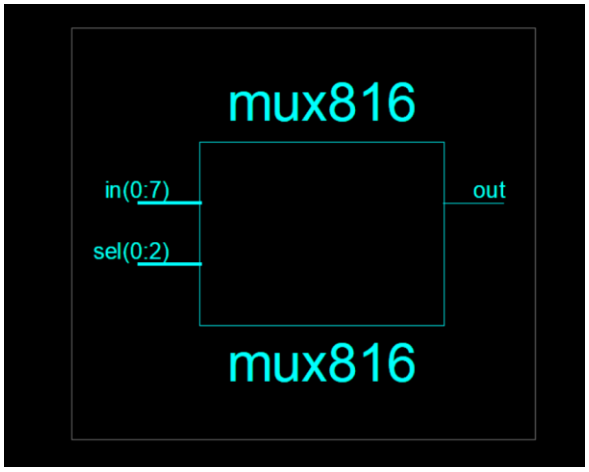


```
initial begin
		// Initialize Inputs
		a = 0;
		b = 0;
		cin = 0;
		#100;
		a = 1;
		b = 0;
		cin = 1;
		#100;
		a = 1;
		b = 1;
		cin = 1;
end
```



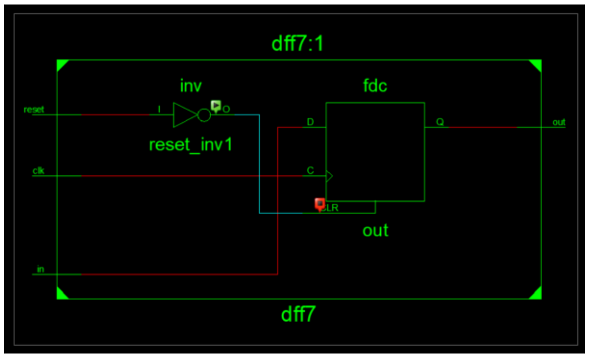
Question 6: 8:1 using 4:1 and 2:1

```
module mux816(in,sel,out
    );
input [0:7]in;
input [0:2]sel;
output out;
wire x,y;
mux416 m1(in[0],in[1],in[2],in[3],sel[1:2],x);
mux416 m2(in[0],in[1],in[2],in[3],sel[1:2],y);
mux216 m3(x,y,sel[0],oit);
endmodule

module mux416(a,b,c,d,sel,out
    );
input a,b,c,d;
input [0:1]sel;
output out;
assign out=(~sel[0] & ~sel[1]&a) |  (~sel[0] & sel[1]&b)
| (sel[0] & ~sel[1]&c) | (sel[0] & sel[1]&d);
endmodule

module mux216(a,b,sel,out
    );
input a,b;
input sel;
output out;
reg x,y;
assign out=(~sel &a) | (sel &b) ;
endmodule
```

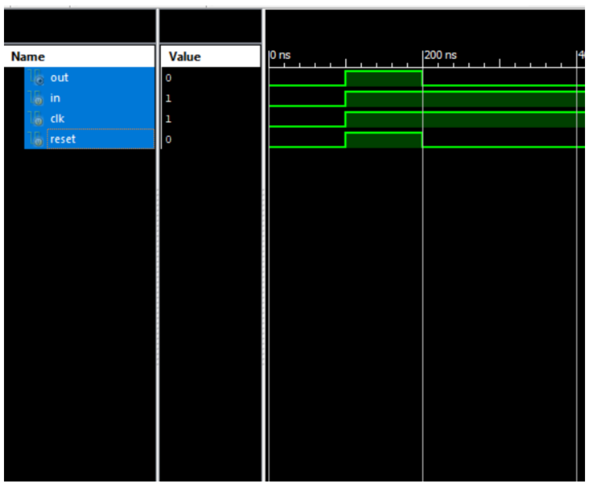


Question7: Dff using asynchronous reset

```
module dff7( in,clk,reset,out
    );
input in,clk,reset;
output reg out;
always@(posedge clk, negedge reset)
begin
if(!reset)
out=0;
else
out=in;
end
endmodule
```

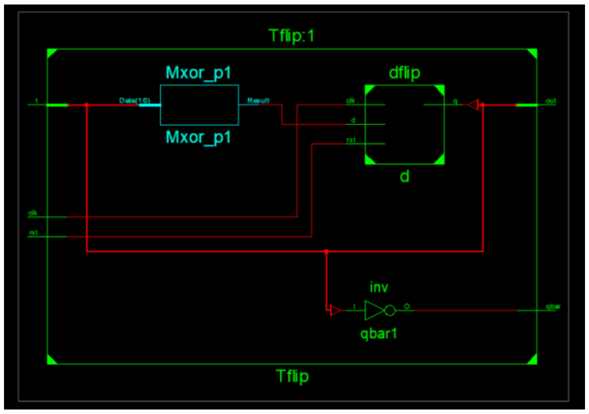


```
initial begin
                // Initialize Inputs
                in = 0;
                clk = 0;
                reset = 0;

                // Wait 100 ns for global reset to
finish
                #100;
     in = 1;
                clk = 1;
                reset = 1;
                // Add stimulus here
    #100;
    in = 1;
                clk = 1;
                reset = 0;
        end
```



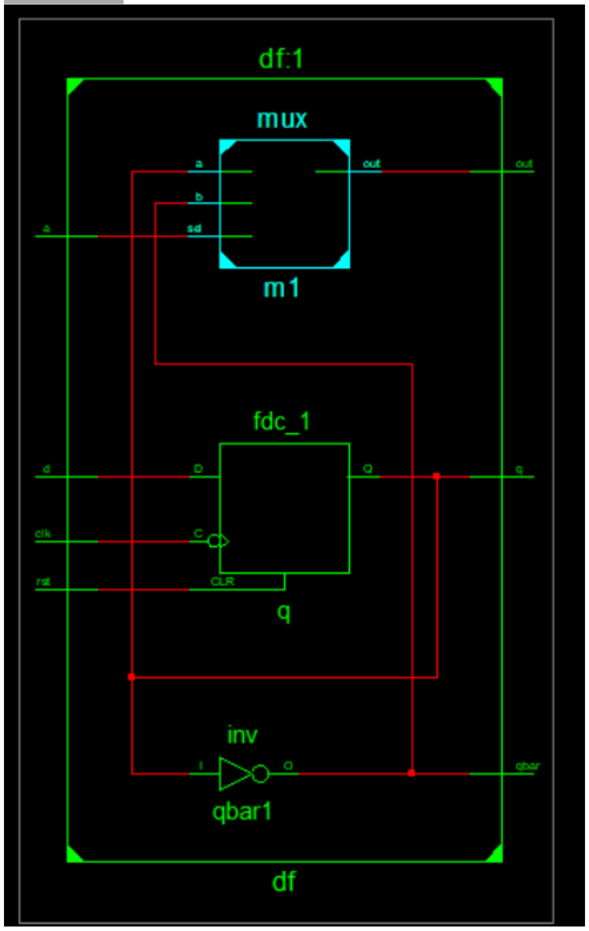
Question 8: TFF using DFF

```
module Tflip(t,out,clk,rst,qbar);
input t,clk,rst;
output out,qbar;
wire o,p;
assign o=out;
assign p=t^o;
dflip d(p,rst,clk,out);
assign qbar=~out;
endmodule

module dflip(
    d,rst,clk,q);
input d,rst,clk;
output reg q;
always@(negedge clk or posedge rst )
begin
if(rst)
q=1'b0;
else
q=d;
end
endmodule
```



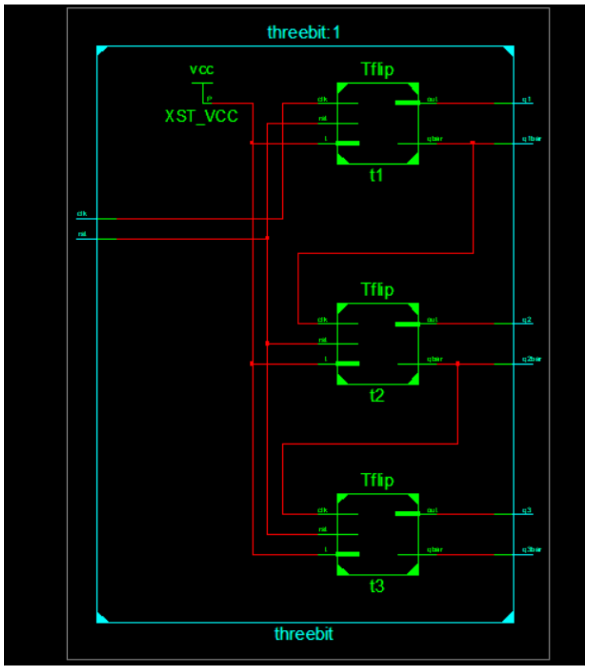
Question9: Dff+Mux

```
module df(
    d,rst,clk,q,a,qbar,out);
input d,rst,clk,a;
output reg q;
output out;
output qbar;
always@(negedge clk or posedge rst )
begin
if(rst)
q=1'b0;
else
q=d;
end
assign qbar=~q;
mux m1(q,qbar,a,out);
endmodule
```



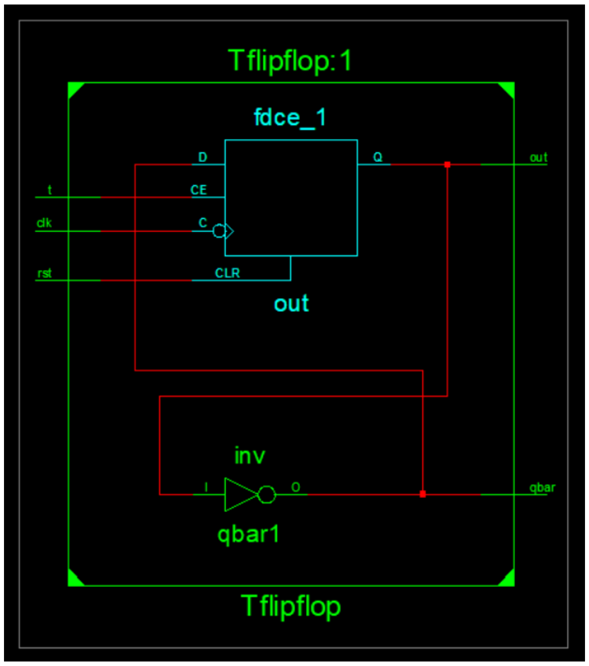
Question 10:3bit counter using TFF

```
module threebit(q1,q2,q3,q1bar,q2bar,q3bar,clk,rst
    );
input clk,rst;
output q1,q2,q3,q1bar,q2bar,q3bar;

Tflip t1(1'b1,q1,clk,rst,q1bar);
Tflip t2(1'b1,q2,q1bar,rst,q2bar);
Tflip t3(1'b1,q3,q2bar,rst,q3bar);

endmodule
```



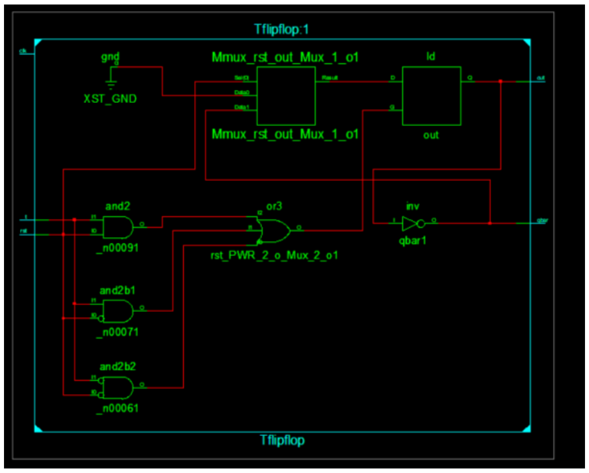
Question 16:TFF using behavioral logic

```
module Tflipflop(t,out,clk,rst,qbar);
input t,clk,rst;
output reg out;
output qbar;
always@(negedge clk or posedge rst )
begin
if(rst)
out=0;
else if(t==0)
out=out;
else
out=~out;
end
assign qbar=~out;
endmodule
```



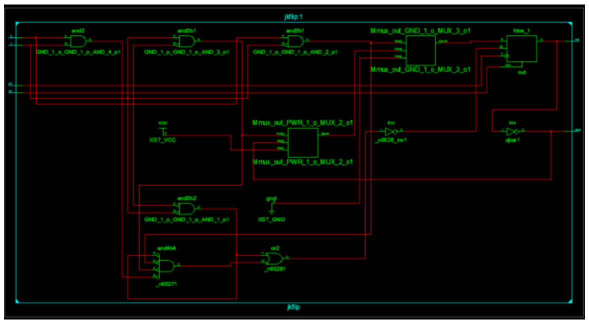
Question 17:TFF using case logic

```
module Tflipflop(t,out,clk,rst,qbar);
input t,clk,rst;
output reg out;
output qbar;
always@(negedge clk or posedge rst )
begin
case({rst,t})
2'b00 : out=0;
2'b01 : out=0;
2'b10 : out=out;
2'b11 : out=~out;
endcase
end
assign qbar=~out;
endmodule
```



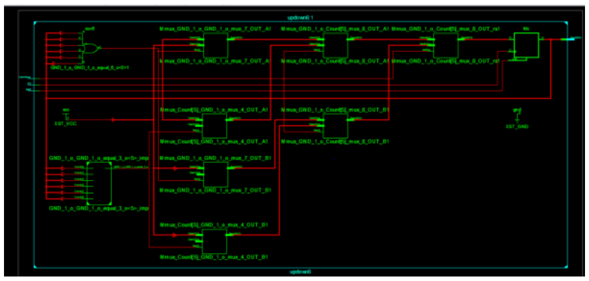
Question18: JK using if else

```
module jkflip(j,k,out,clk,rst,qbar);
input j,k,clk,rst;
output reg out;
output qbar;
always@(negedge clk or posedge rst )
begin
if(rst)
out=0;
else if(j==0 && k==0)
out=out;
else if(j==0 && k==1)
out=0;
else if(j==1 && k==0)
out=1;
else if(j==1 && k==1)
out=~out;
end
assign qbar=~out;
endmodule
```



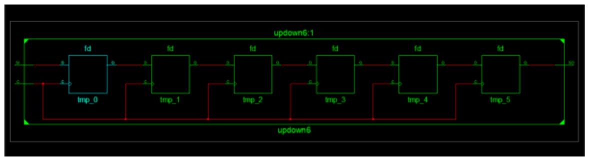
Question 11-12 6bit up down

```
module updown6(
Clk,
reset,
    UpOrDown,  //high for UP counter and low for Down counter
    Count
    );
//input ports and their sizes
    input Clk,reset,UpOrDown;
    //output ports and their size
    output [5 : 0] Count;
    //Internal variables
    reg [5 : 0] Count = 0;
always @(posedge(Clk) or posedge(reset))
     begin
        if(reset == 1)
            Count <= 0;
        else
            if(UpOrDown == 1)   //Up mode selected
                if(Count == 15)
                    Count <= 0;
                else
                    Count <= Count + 1; //Incremend Counter
            else  //Down mode selected
                if(Count == 0)
                    Count <= 15;
                else
                    Count <= Count - 1; //Decrement counter
     end

endmodule
```

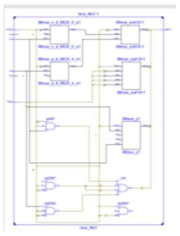


Question14: 6 bit shift register

```
module updown6 (C, SI, SO);
input C,SI;
output SO;
reg [5:0] tmp;

  always @(posedge C)
    begin
      tmp = tmp << 1;
      tmp[0] = SI;
    end
    assign SO  = tmp[5];
endmodule
```



Question3: 4:1 MUX with 5 different coding Styles



```
module mux_4to1(a,b,c,d,sel,out,in
   );
input [0:3]in;
input a,b,c,d;
input [0:1]sel;
output [0:4]out;
reg x,y;
assign out[0]=(~sel[0] & ~sel[1]&a) |  (~sel[0] & sel[1]&b) | (sel[0] & ~sel[1]&c) | (sel[0] & sel[1]&d);
always@(*)
begin
case(sel)
2'b00 : y=a;
2'b01 : y=b;
2'b10 : y=c;
2'b11 : y=d;
endcase
end
assign out[1]=y;
assign out[2] = sel[1] ? (sel[0] ? b : a ) : (sel[0] ? d : c );
assign out[3]=in[sel];
wire NS0, NS1;
wire Y0, Y1, Y2, Y3;
not N1(NS0, sel[0]);
not N2(NS1, sel[1]);
and A1(Y0, a, NS1, NS0);
and A2(Y1, b, NS1, sel[0]);
and A3(Y2, c, sel[1], NS0);
and A4(Y3, d, sel[1], sel[0]);
or O1(out[4], Y0, Y1, Y2, Y3);
endmodule
```