**Experiment 6** : **Arithmetic Logic Unit (ALU)**

**AIM:** TO simulate and synthesize the alu .

**Apparatus:** A Pc running windows with Xilinx Ise design suite installed in it.

**Verilog Program:**

module alu\_311(

output reg [7:0]Out\_311,

input [3:0]In1\_311,In2\_311,Sel\_311 );

always@(Sel\_311,In1\_311,In2\_311)

begin

if(Sel\_311==0)

Out\_311=In1\_311+In2\_311;

else if(Sel\_311==1)

Out\_311=In1\_311-In2\_311;

else if(Sel\_311==2)

Out\_311=In1\_311\*In2\_311;

else if(Sel\_311==3)

Out\_311=In1\_311/In2\_311;

else if(Sel\_311==4)

Out\_311=In1\_311%In2\_311;

else if(Sel\_311==5)

Out\_311=In1\_311\*In2\_311;

else if(Sel\_311==6)

Out\_311=!In1\_311;

else if(Sel\_311==7)

Out\_311=In1\_311&&In2\_311;

else if(Sel\_311==8)

Out\_311=In1\_311||In2\_311;

else if(Sel\_311==9)

Out\_311=In1\_311|In2\_311;

else if(Sel\_311==10)

Out\_311=In1\_311&In2\_311;

else if(Sel\_311==11)

Out\_311=In1\_311^In2\_311;

else if(Sel\_311==12)

Out\_311=In1\_311<<In2\_311;

else if(Sel\_311==13)

Out\_311=In1\_311>>In2\_311;

else if(Sel\_311==14)

Out\_311=In1\_311<<<In2\_311;

else

Out\_311=In1\_311>>>In2\_311;

end

endmodule

**Test bench :**

module tb\_alu\_311;

reg [3:0] In1\_311;

reg [3:0] In2\_311;

reg [3:0] Sel\_311;

wire [7:0] Out\_311;

alu\_311 uut (

.Out\_311(Out\_311),

.In1\_311(In1\_311),

.In2\_311(In2\_311),

.Sel\_311(Sel\_311)

);

initial begin

$monitor($time,"in1\_311=%b in2\_311=%b sel\_311=%b out\_311=%b",In1\_311,In2\_311,Sel\_311,Out\_311);

In1\_311 = 4'b1010;

In2\_311 = 4'b0011;

for(Sel\_311=0;Sel\_311<=15;Sel\_311=Sel\_311+1)

begin

#10

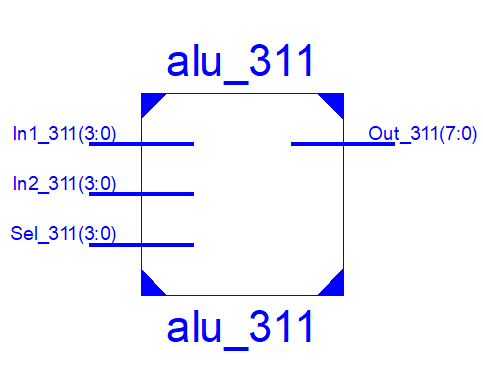
In1\_311=4'b0110;

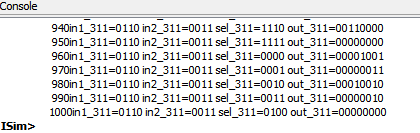
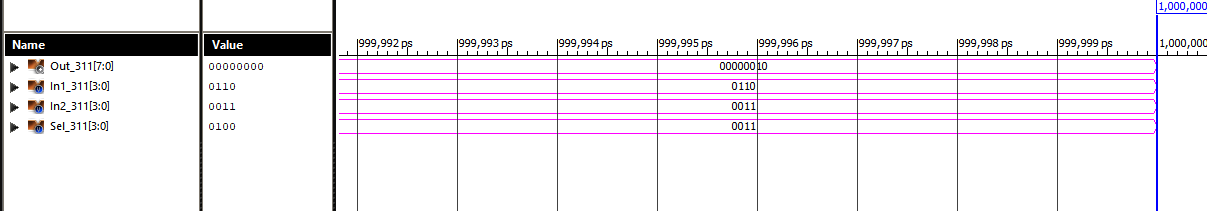
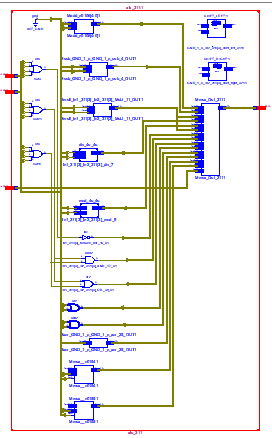
In2\_311=4'b0011;

end

end

endmodule





**Experiment 7** : **Flip Flops**

**AIM:** TO simulate and synthesize the Flip Flops.

**Apparatus:** A Pc running windows with Xilinx Ise design suite installed in it.

**Verilog Program:**

1. **D flip flop:**

module dff\_311(

input d\_311,

input clk,

input reset\_311,

output reg q\_311,

output reg qb\_311

);

always@(posedge clk)

begin

if(reset\_311==1)

begin

q\_311 <=0;

qb\_311 <=1;

end

else

begin

q\_311<=d\_311;

qb\_311<=~d\_311;

end

end

endmodule

**Test bench :**

module tb\_dff\_311;

reg d\_311;

reg clk;

reg reset\_311;

wire q\_311;

wire qb\_311;

dff\_311 uut (

.d\_311(d\_311),

.clk(clk),

.reset\_311(reset\_311),

.q\_311(q\_311),

.qb\_311(qb\_311)

);

initial begin

$monitor($time," Clk=%b,Reset\_311=%b,D\_311=%b,Q\_311=%b,Q\_311'=%b",clk,reset\_311,d\_311,q\_311,qb\_311);

d\_311 = 0; clk = 0; reset\_311 = 1;

#8 reset\_311=0; #10 d\_311=1; #10 d\_311=0;

#40; $stop;

end

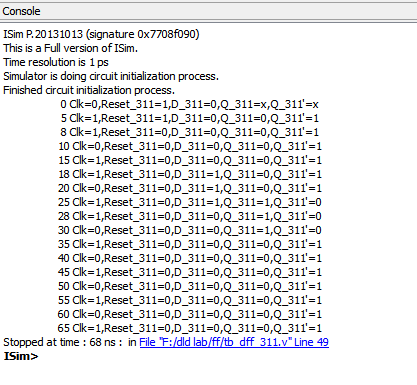
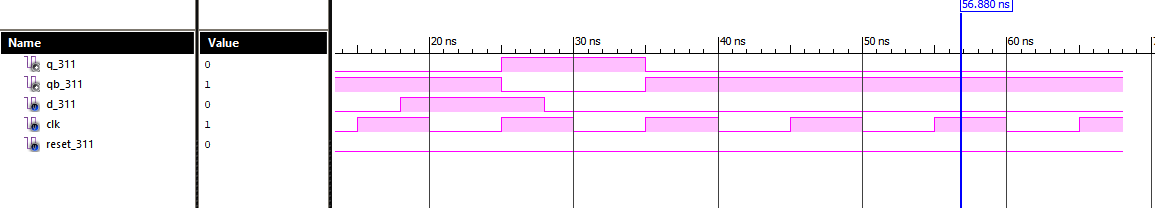
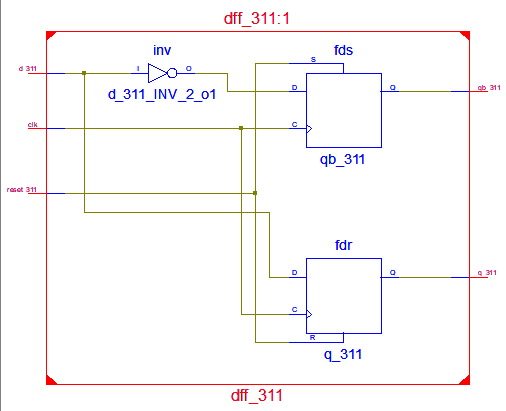
initial begin

forever

#5 clk=~clk;

end

endmodule



**b) jk flip flop**

**Verilog Program:**

module jkff\_311(

input j\_311,

input k\_311,

input clk\_311,

input reset,

output reg q\_311,

output reg qb\_311

);

always@(negedge clk\_311)

begin

case({reset,j\_311,k\_311})

3'b100: begin

q\_311<=0; qb\_311<=1;

end

3'b001: begin

q\_311<=0; qb\_311<=1;

end

3'b010: begin

q\_311<=1; qb\_311<=0;

end

3'b011: begin

q\_311<=~q\_311; qb\_311<=~qb\_311;

end

endcase

end

endmodule

**Test bench :**

module tb\_jkff\_311;

reg j\_311;

reg k\_311;

reg clk\_311;

reg reset;

wire q\_311;

wire qb\_311;

jkff\_311 uut (

.j\_311(j\_311),

.k\_311(k\_311),

.clk\_311(clk\_311),

.reset(reset),

.q\_311(q\_311),

.qb\_311(qb\_311)

);

initial

begin

$monitor($time,"clk\_311=%b,Reset=%b,j\_311=%b,k\_311=%b,Q\_311=%b,Qb\_311=%b",clk\_311,reset,j\_311,k\_311,q\_311,qb\_311);

j\_311 = 0;

k\_311 =0;

clk\_311 = 0;

reset = 1;

#8 reset=0;

#10 j\_311=0; k\_311=0;

#10 j\_311=0; k\_311=1;

#10 j\_311=1;k\_311=0;

#10 j\_311=1;k\_311=1;

#20;

$stop;

end

initial

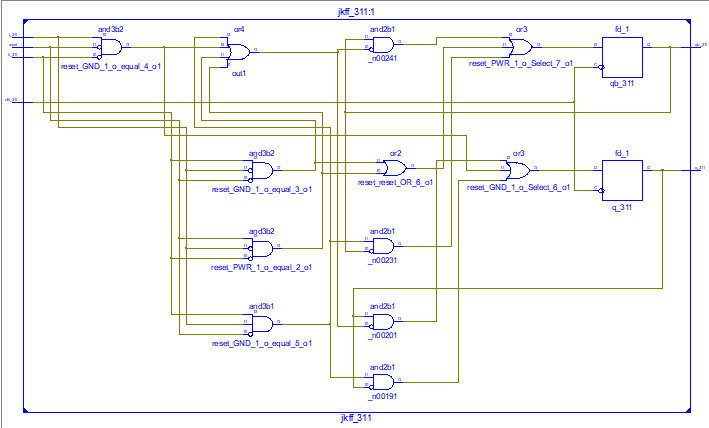
begin

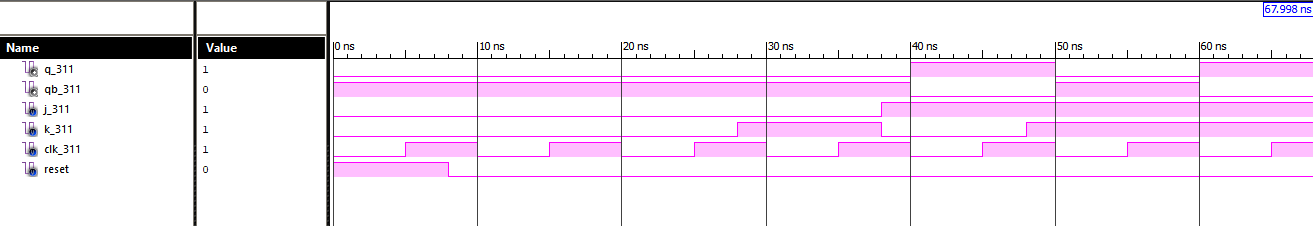
forever

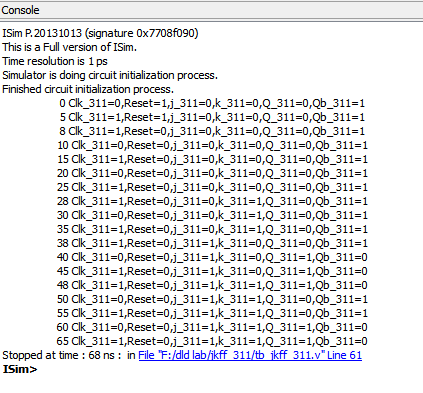
#5 clk\_311=~clk\_311;

end

endmodule







**c) sr flip flop**

**Verilog Program:**

module srff\_311 (

input s\_311,

input r\_311,

input clk,

input reset,

output reg q\_311,

output reg qb\_311

);

always@(posedge clk)

begin

if((reset==1)||((s\_311==0)&&(r\_311==1)))

begin

q\_311<=0;

qb\_311<=1;

end

else if(((s\_311==1)&&(r\_311==0))&&(reset==0))

begin

q\_311<=1;

qb\_311<=0;

end

end

endmodule

**Test bench :**

module tb\_srff\_311;

reg s\_311;

reg r\_311;

reg clk;

reg reset;

wire q\_311;

wire qb\_311;

srff\_311 uut (

.s\_311(s\_311),

.r\_311(r\_311),

.clk(clk),

.reset(reset),

.q\_311(q\_311),

.qb\_311(qb\_311)

);

initial

begin

$monitor($time," Clk=%b, Reset=%b, S\_311=%b, R\_311=%b, q\_311=%b, qb\_311=%b",clk,reset,s\_311,r\_311,q\_311,qb\_311);

s\_311 = 0;

r\_311=0;

clk = 0;

reset= 1;

#8 reset=0;

#10 s\_311=0; r\_311=0;

#10 s\_311=0; r\_311=1;

#10 s\_311=1;r\_311=0;

#50;

$stop;

end

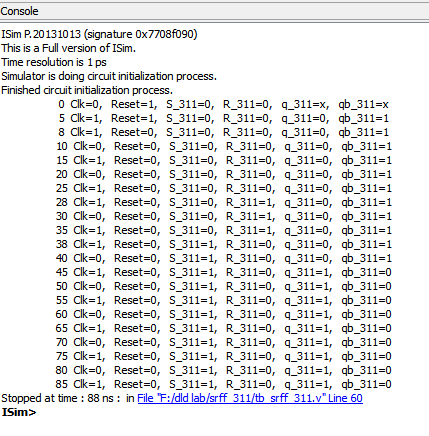
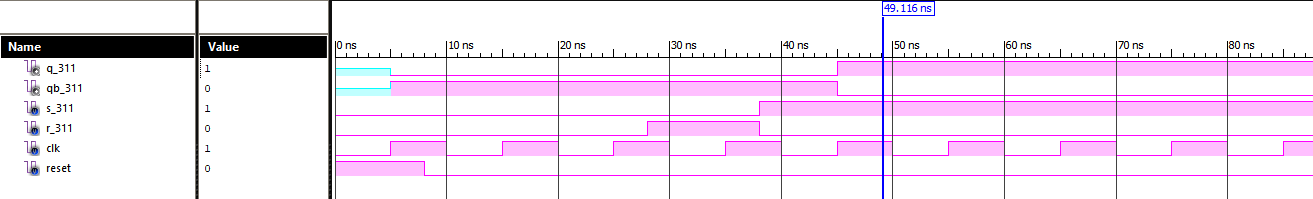
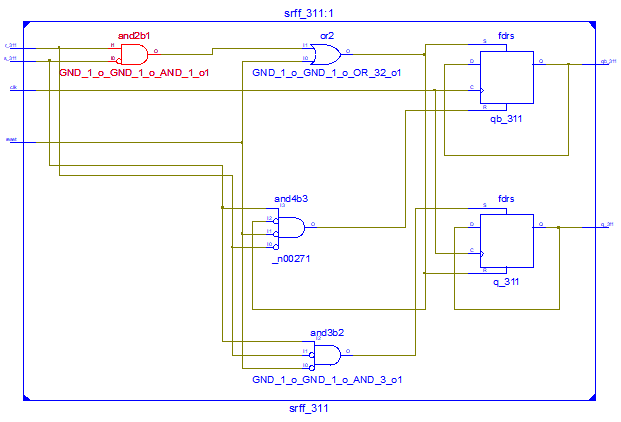
initial

begin

forever

#5 clk=~clk;

end

endmodule

**d) t flip flop**

**Verilog Program:**

module tff\_311(

input t\_311,

input clk\_311,

input reset,

output reg q\_311,

output reg qb\_311

);

always@(negedge clk\_311)

begin

if(reset==1)

begin

q\_311<=0;

qb\_311<=1;

end

if(t\_311==1)

begin

q\_311 <= ~q\_311;

qb\_311 <= ~qb\_311;

end

end

endmodule

**Test bench :**

module tb\_tff\_311;

reg t\_311;

reg clk\_311;

reg reset;

wire q\_311;

wire qb\_311;

tff\_311 uut (

.t\_311(t\_311),

.clk\_311(clk\_311),

.reset(reset),

.q\_311(q\_311),

.qb\_311(qb\_311)

);

initial

begin

monitor($time,"Clock=%b,reset=%b,T\_311=%b,Q\_311=%b,Q\_311'=%b",clk\_311,reset,t\_311,q\_311,qb\_311);

t\_311 = 0; clk\_311 = 0; reset = 1;

#8 reset=0;

#10 t\_311=1;

#10 t\_311=0;

#10 t\_311=1;

#10 t\_311=0;

#10 t\_311=1;

#40; $stop;

end

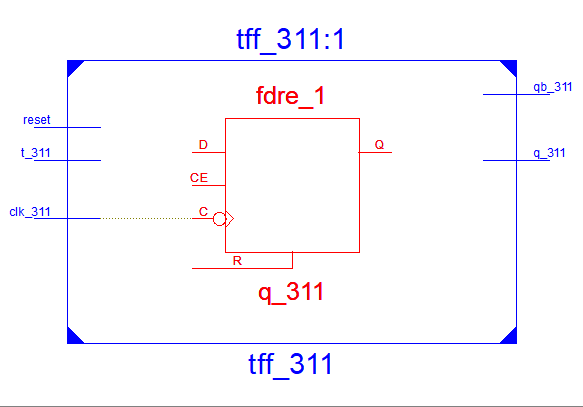
initial

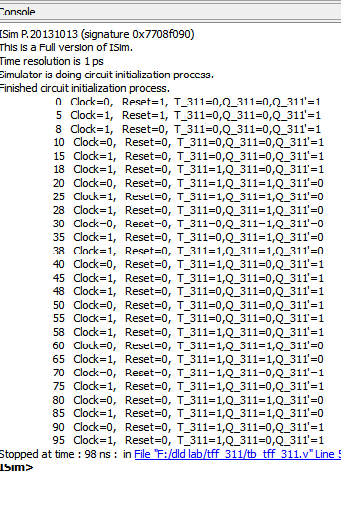
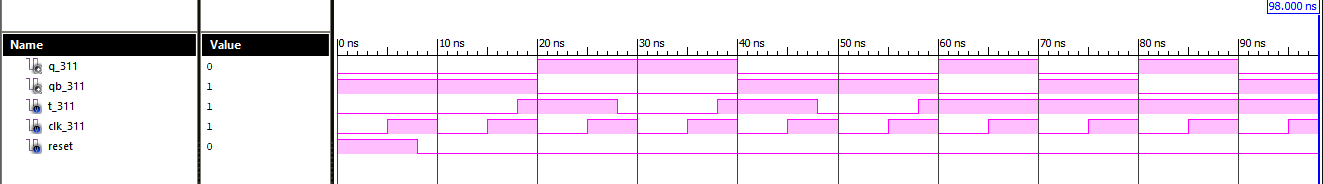
begin

forever

#5 clk\_311=~clk\_311;

end

endmodule****



**Experiment 8** **: counters**

**AIM:** TO simulate and synthesize the counters.

**Apparatus:** A Pc running windows with Xilinx Ise design suite installed in it.

1. **Up counter:**

**Verilog Program:**

module up\_counter\_311(

output reg [7:0] count\_311,

input clk\_311,

input reset\_311

);

always@(negedge clk\_311)

begin

if(reset\_311==1)

count\_311=0;

else

count\_311=count\_311+1;

end

endmodule

**Test bench :**

module tb\_up\_count\_311;

reg clk\_311;

reg reset\_311;

wire [7:0] count\_311;

up\_counter\_311 uut (

.count\_311(count\_311),

.clk\_311(clk\_311),

.reset\_311(reset\_311)

);

initial begin

$monitor($time," clk\_311=%b, reset\_311=%b, count\_311=%b",clk\_311,reset\_311,count\_311);

clk\_311 = 0;

reset\_311 = 1;

#8 reset\_311=0;

#3000; $stop;

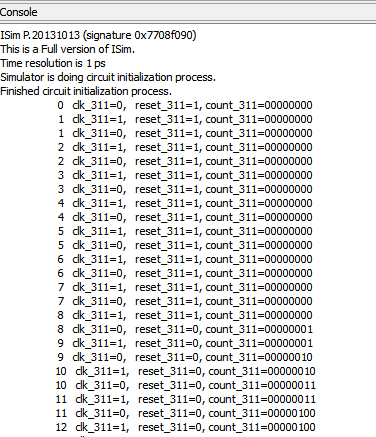
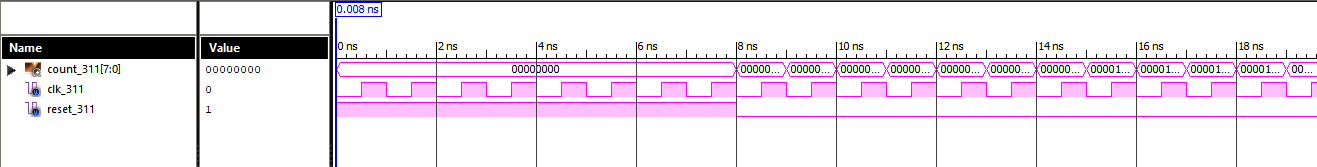
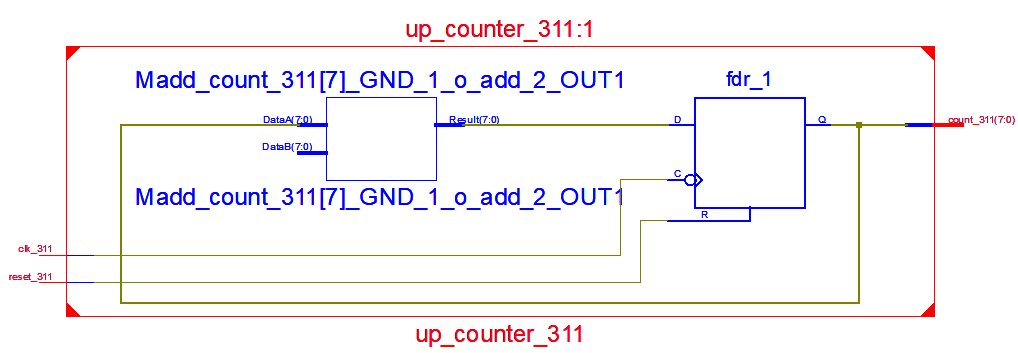
end

initial

begin

forever #0.5 clk\_311=~clk\_311;

end

endmodule

1. **Down counter:**

**Verilog Program:**

module down\_counter\_311(

output reg [7:0] count\_311,

input clk\_311,

input reset\_311

);

always@(negedge clk\_311)

begin

if(reset\_311==1)

count\_311=255;

else

count\_311=(count\_311)-1;

end

endmodule

**Test bench :**

module tb\_down\_counter\_311;

reg clk\_311;

reg reset\_311;

wire [7:0] count\_311;

down\_counter\_311 uut (

.count\_311(count\_311),

.clk\_311(clk\_311),

.reset\_311(reset\_311)

);

initial begin $monitor($time,"clk\_311=%b,reset\_311=%b,count\_311=%b",clk\_311,reset\_311,count\_311);

clk\_311 = 0;

reset\_311 = 1;

#8 reset\_311=0;

#3000;

$stop;

end

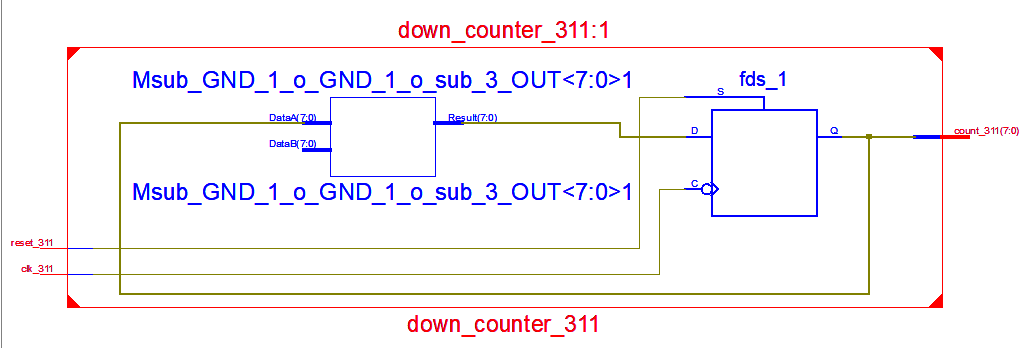
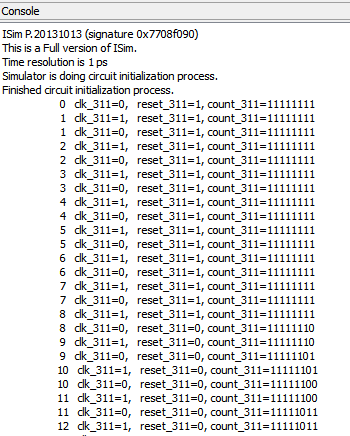
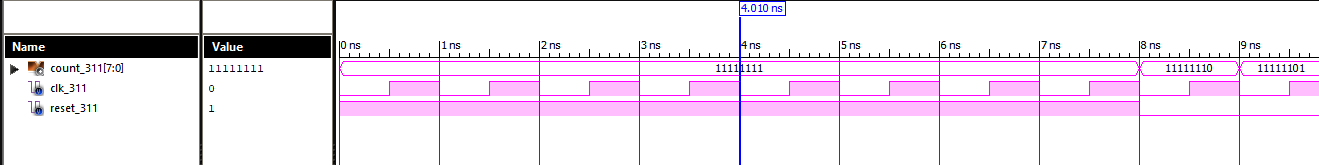
initial

begin

forever #0.5 clk\_311=~clk\_311;

end

endmodule

****

1. **Up-down counter:**

**Verilog Program:**

module up\_down\_311(

output reg [3:0] count\_311,

input ud\_311,

input clk\_311,

input reset\_311

);

always@(negedge clk\_311)

begin

if(ud\_311==1)

begin

if(reset\_311==1)

count\_311=0;

else

count\_311=count\_311+1;

end

else

begin

if(reset\_311==1)

count\_311=255;

else

count\_311=count\_311-1;

end

end

endmodule

**Test bench :**

module tb\_up\_down\_311;

reg ud\_311;

reg clk\_311;

reg reset\_311;

wire [3:0] count\_311;

up\_down\_311 uut (

.count\_311(count\_311),

.ud\_311(ud\_311),

.clk\_311(clk\_311),

.reset\_311(reset\_311)

);

initial begin

$monitor($time," clk\_311=%b, reset\_311=%b, count\_311=%b",clk\_311,reset\_311,count\_311);

ud\_311 = 1;

clk\_311 = 0;

reset\_311 = 1;

#8 reset\_311=0;

#100; ud\_311=0;

#3000; $stop;

end

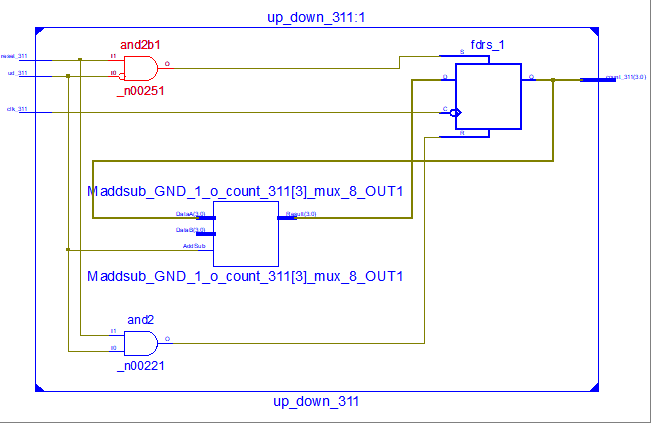
initial

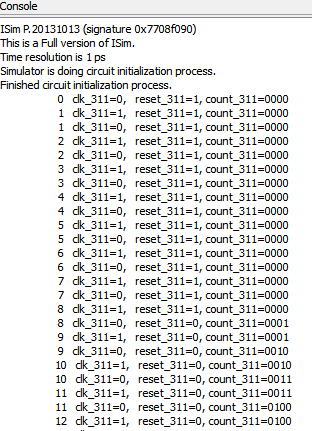
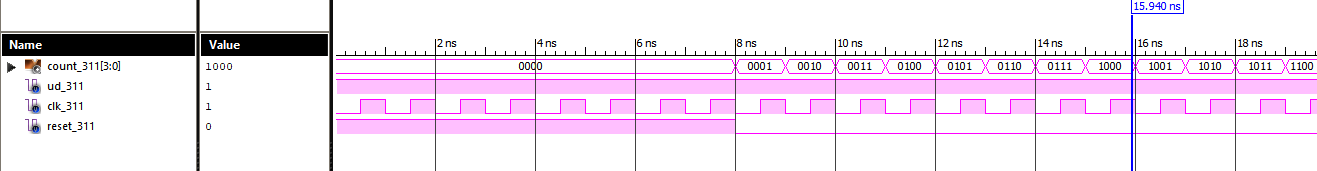
begin

forever

#0.5 clk\_311=~clk\_311;

End

endmodule



**Experiment 9** : **Mealy and Moore**

**AIM:** TO simulate and synthesize the countersMealy and Moore.

**Apparatus:** A Pc running windows with Xilinx Ise design suite installed in it.

1. **Mealy**

**Verilog Program:**

module mealy\_311 (out\_311,in\_311,clk\_311,rst\_311);

input in\_311,clk\_311,rst\_311;

output reg out\_311;

reg [1:0] state\_311;

parameter s0\_311=0,s1\_311=1,s2\_311=2,s3\_311=3;

always@(posedge clk\_311 or posedge rst\_311)

if(rst\_311)

state\_311<=s0\_311;

else

case(state\_311)

s0\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <=s0\_311;

s1\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <=s2\_311;

s2\_311: if(in\_311) state\_311 <=s3\_311;

else state\_311 <=s0\_311;

s3\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <=s2\_311;

endcase

always@(state\_311,in\_311)

case(state\_311)

s0\_311: if(in\_311) out\_311 <=0;

else out\_311 <=0;

s1\_311: if(in\_311) out\_311 <=0;

else out\_311 <=0;

s2\_311: if(in\_311) out\_311 <=0;

else out\_311 <=0;

s3\_311: if(in\_311) out\_311 <=1;

else out\_311 <=0;

endcase

endmodule

**Test bench :**

module tb\_melay\_311;

reg in\_311;

reg clk\_311;

reg rst\_311;

wire out\_311;

mealy\_311 uut (

.out\_311(out\_311),

.in\_311(in\_311),

.clk\_311(clk\_311),

.rst\_311(rst\_311)

);

initial begin

$monitor($time,"in\_311=%b clk\_311=%b rst\_311=%b out\_311=%b",in\_311,clk\_311,rst\_311,out\_311);

in\_311 = 0; clk\_311 = 0; rst\_311 = 1;

#8 rst\_311 =0;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#20 in\_311=1;

#10 in\_311=0;

#10 in\_311=1;

#30 in\_311=1;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#10 in\_311=1;

#10 in\_311=1;

#100; $stop;

end

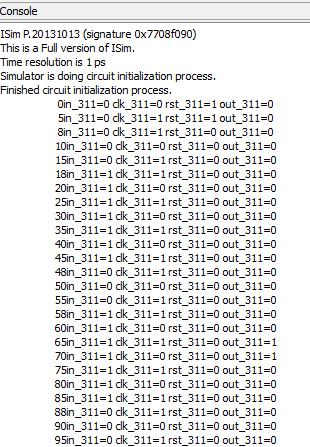
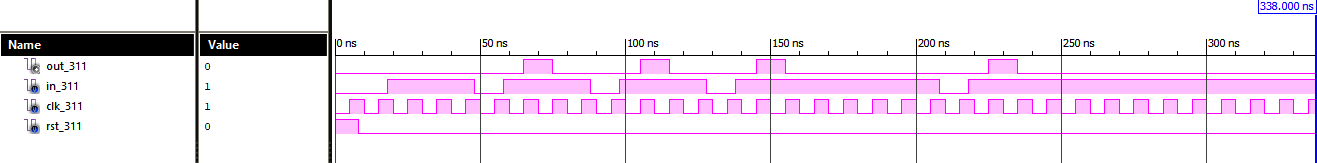
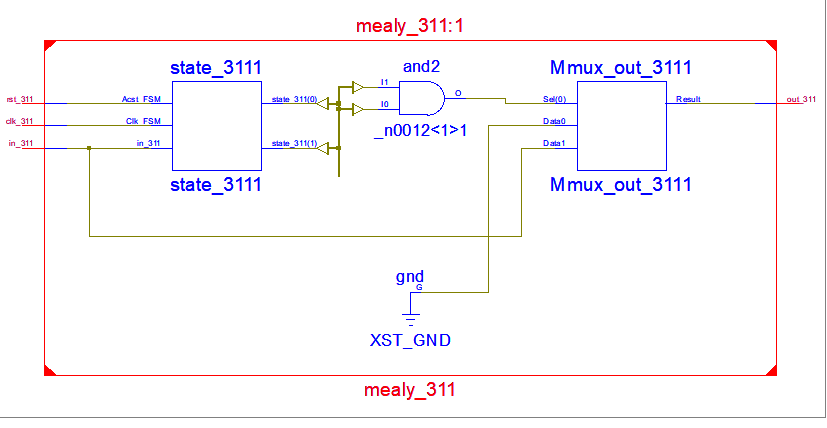
initial

begin

forever#5 clk\_311 = ~clk\_311;

end

endmodule



1. **Moore:**

**Verilog Program:**

module moore\_311 (out\_311,in\_311, clk\_311,rst\_311);

output reg out\_311;

input in\_311, clk\_311, rst\_311;

reg [2:0] state\_311;

parameter s0\_311=0,s1\_311=1,s2\_311=2,s3\_311=3,s4\_311=4;

always@(negedge clk\_311 or posedge rst\_311)

if(rst\_311)

state\_311<=s0\_311;

else

case(state\_311)

s0\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <= s0\_311;

s1\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <= s2\_311;

s2\_311: if(in\_311) state\_311 <=s3\_311;

else state\_311 <= s0\_311;

s3\_311: if(in\_311) state\_311 <=s4\_311;

else state\_311 <= s1\_311;

s4\_311: if(in\_311) state\_311 <=s1\_311;

else state\_311 <=s2\_311;

endcase

always@(state\_311)

case(state\_311)

s0\_311: out\_311 <=0;

s1\_311: out\_311 <=0;

s2\_311: out\_311 <=0;

s3\_311: out\_311 <=0;

s4\_311: out\_311 <=1;

endcase

endmodule

**Test bench :**

module tb\_moore\_311;

reg in\_311;

reg clk\_311;

reg rst\_311;

wire out\_311;

moore\_311 uut (

.out\_311(out\_311),

.in\_311(in\_311),

.clk\_311(clk\_311),

.rst\_311(rst\_311)

);

initial begin

$monitor($time,"in\_311=%b clk\_311=%b rst\_311=%b out\_311=%b",in\_311,clk\_311,rst\_311,out\_311);

in\_311 = 0;

clk\_311 = 0;

rst\_311 = 1;

#8 rst\_311 =0;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#20 in\_311=1;

#10 in\_311=0;

#10 in\_311=1;

#30 in\_311=1;

#10 in\_311=1;

#30 in\_311=0;

#10 in\_311=1;

#10 in\_311=1;

#10 in\_311=1;

#100; $stop;

end

initial begin

forever#5 clk\_311 = ~clk\_311;

end

endmodule