**Experiment 1** : REALISATION OF LOGIC GATES

**AIM:** TO simulate all gates using gate level modelling.

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

**Verilog Program:**

module g\_311(

input a\_311,

input b\_311,

output or\_311,

output and\_311,

output nand\_311,

output nor\_311,

output xor\_311,

output not\_311,

output xnor\_311

);

not(not\_311,a\_311);

or(or\_311,a\_311,b\_311);

and(and\_311,a\_311,b\_311);

nand(nand\_311,a\_311,b\_311);

nor(nor\_311,a\_311,b\_311);

xor(xor\_311,a\_311,b\_311);

xnor(xnor\_311,a\_311,b\_311);

endmodule

**Testbench :**

module tb\_g\_311;

// Inputs

reg a\_311;

reg b\_311;

// Outputs

wire or\_311;

wire not\_311;

wire and\_311;

wire nand\_311;

wire nor\_311;

wire xor\_311;

wire xnor\_311;

// Instantiate the Unit Under Test (UUT)

g\_311 uut (

.a\_311(a\_311),

.b\_311(b\_311),

.not\_311(not\_311),

.or\_311(or\_311),

.and\_311(and\_311),

.nand\_311(nand\_311),

.nor\_311(nor\_311),

.xor\_311(xor\_311),

.xnor\_311(xnor\_311)

);

initial begin

// Initialize Inputs

a\_311 = 0;

b\_311 = 0;

#20

a\_311 = 0;

b\_311 = 1;

#30

a\_311 = 1;

b\_311 = 0;

#40

a\_311 = 1;

b\_311 = 1;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

$finish;

end

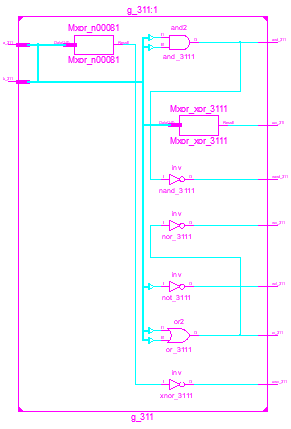
initial begin

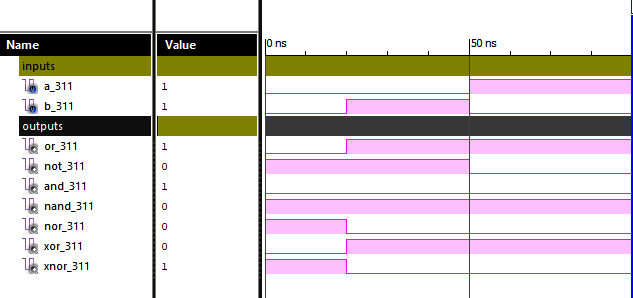
// Initialize Inputs

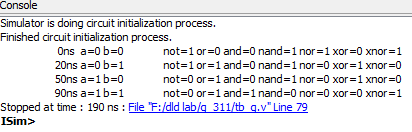
$monitor($time,"ns a=%b b=%b not=%b or=%b and=%b nand=%b nor=%b xor=%b xnor=%b",a\_311,b\_311,not\_311,or\_311,and\_311,nand\_311,nor\_311,xor\_311,xnor\_311);

end

endmodule







**Result:** Realisation and simulation of all gates are done.

**Experiment 2** : ADDERS AND SUBSTRACTORS

**AIM:** To Write a Verilog HDL program for various adders and Substractors using gate level and data flow modelling and simulate synthesis using Xilinx Ise design suite.

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

**Verilog Program:**

1. **Half adder using data flow modelling:**

**Data flow**

module ha\_311(

input a\_311,

input b\_311,

output s\_311,

output c\_311

);

assign s\_311= a\_311 ^ b\_311;

assign c\_311= a\_311 & b\_311;

endmodule

**Test Bench :**

module tb\_ha;

// Inputs

reg a\_311;

reg b\_311;

// Outputs

wire s\_311;

wire c\_311;

// Instantiate the Unit Under Test (UUT)

ha\_311 uut (

.a\_311(a\_311),

.b\_311(b\_311),

.s\_311(s\_311),

.c\_311(c\_311)

);

initial begin

// Initialize Inputs

a\_311 = 0;

b\_311 = 0;

#50

a\_311 = 0;

b\_311 = 1;

#50

a\_311 = 1;

b\_311 = 0;

#50

a\_311 = 1;

b\_311 = 1;

// Wait 100 ns for global reset to finish

#100;

$finish;

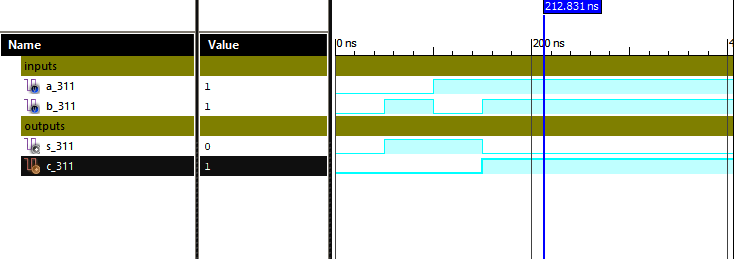
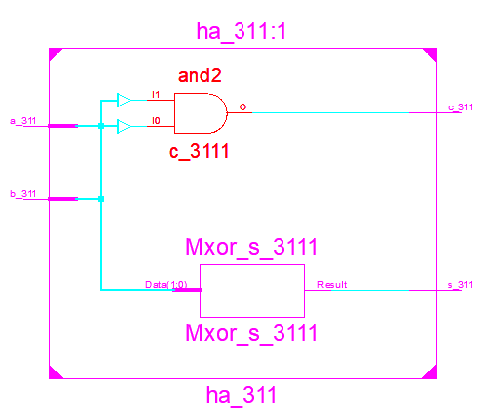
end

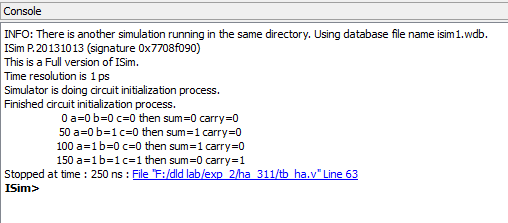
initial begin

$monitor($time," a=%b b=%b c=%b then sum=%b carry=%b",a\_311,b\_311,c\_311,s\_311,c\_311);

end

endmodule





1. **Half subtractor using data flow modelling:**

**Verilog program:**

**Data flow**

module hs\_311(

input x\_311,

input y\_311,

output d\_311,

output b\_311

);

assign d\_311= x\_311 ^ y\_311;

assign b\_311= (~x\_311)& y\_311;

endmodule

**Test Bench :**

module tb\_hs;

// Inputs

reg x\_311;

reg y\_311;

// Outputs

wire d\_311;

wire b\_311;

// Instantiate the Unit Under Test (UUT)

hs\_311 uut (

.x\_311(x\_311),

.y\_311(y\_311),

.d\_311(d\_311),

.b\_311(b\_311)

);

initial begin

// Initialize Inputs

x\_311 = 0;

y\_311 = 0;

#50

x\_311 = 0;

y\_311 = 1;

#50

x\_311 = 1;

y\_311 = 0;

#50

x\_311 = 1;

y\_311 = 1;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

$finish;

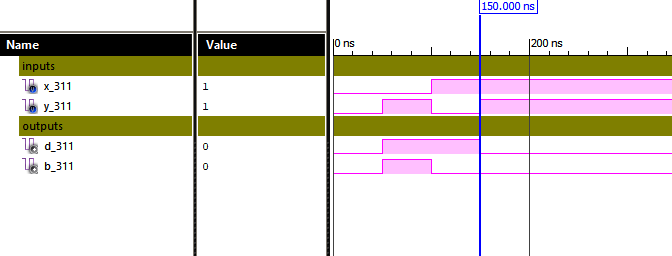
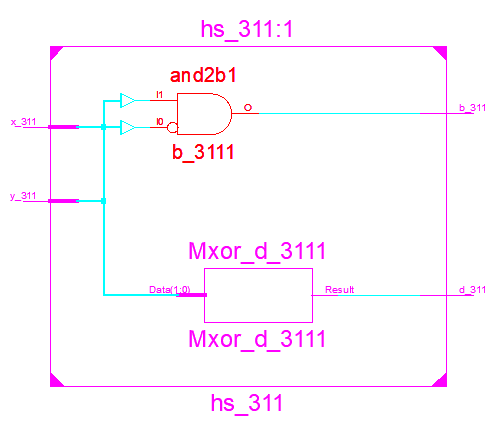
end

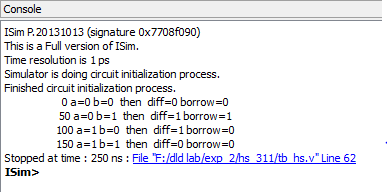
initial begin

$monitor($time," a=%b b=%b then diff=%b borrow=%b",x\_311,y\_311,d\_311,b\_311);

end

endmodule





1. **Full adder using Gate level modelling:**

**Verilog program:**

**Gate level**

module fa\_311(

input a\_311,

input b\_311,

input c\_311,

output s\_311,

output cy\_311,

inout x\_311,

inout y\_311,

inout z\_311

);

xor g1(s\_311,a\_311,b\_311,c\_311);

and a1(x\_311,a\_311,c\_311);

and a2(y\_311,a\_311,b\_311);

and a3(z\_311,b\_311,c\_311);

or o1(cy\_311,x\_311,y\_311,z\_311);

endmodule

**Test Bench :**

module tb\_fa;

// Inputs

reg a\_311;

reg b\_311;

reg c\_311;

// Outputs

wire s\_311;

wire cy\_311;

wire x\_311;

wire y\_311;

wire z\_311;

// Instantiate the Unit Under Test (UUT)

fa\_311 uut (

.a\_311(a\_311),

.b\_311(b\_311),

.c\_311(c\_311),

.s\_311(s\_311),

.cy\_311(cy\_311),

.x\_311(x\_311),

.y\_311(y\_311),

.z\_311(z\_311)

);

initial begin

// Initialize Inputs

a\_311 = 0; b\_311 = 0; c\_311 = 0;

#50

a\_311 = 0; b\_311 = 0; c\_311 = 1;

#50

a\_311 = 0; b\_311 = 1; c\_311 = 0;

#50

a\_311 = 0; b\_311 = 1; c\_311 = 1;

#50

a\_311 = 1; b\_311 = 0; c\_311 = 0;

#50

a\_311 = 1; b\_311 = 0; c\_311 = 1;

#50

a\_311 = 1; b\_311 = 1; c\_311 = 0;

#50

a\_311 = 1; b\_311 = 1; c\_311 = 1;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

$finish;

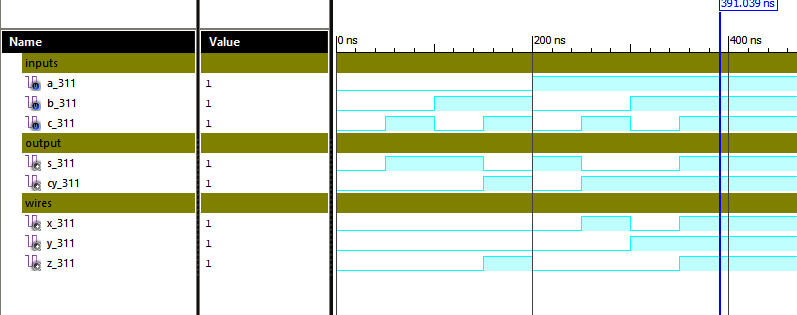
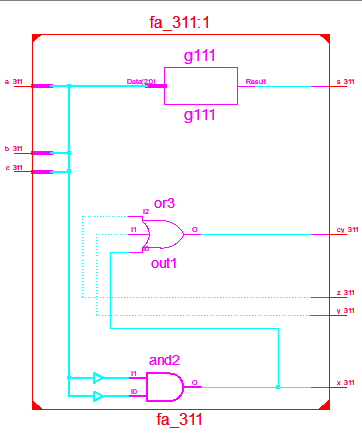
end

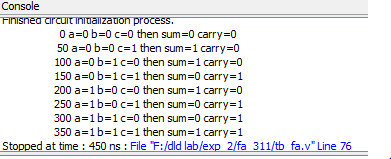
initial begin

$monitor($time," a=%b b=%b c=%b then sum=%b carry=%b",a\_311,b\_311,c\_311,s\_311,cy\_311);

end

endmodule





1. **Full substractor using data flow modelling:**

**Verilog program:**

**Data flow**

module fs\_311(input x\_311, input y\_311,input z\_311,output d\_311,output b\_311 );

assign d\_311= x\_311^y\_311^z\_311;

assign b\_311= ((~x\_311)&y\_311) | (y\_311&z\_311) | ((~x\_311)&z\_311);

endmodule

**Test Bench :**

module tb\_fs;

// Inputs

reg x\_311;

reg y\_311;

reg z\_311;

// Outputs

wire d\_311;

wire b\_311;

// Instantiate the Unit Under Test (UUT)

fs\_311 uut (

.x\_311(x\_311),

.y\_311(y\_311),

.z\_311(z\_311),

.d\_311(d\_311),

.b\_311(b\_311)

);

initial begin

// Initialize Inputs

x\_311 = 0; y\_311 = 0; z\_311 = 0;

#50

x\_311 = 0; y\_311 = 0; z\_311 = 1;

#50

x\_311 = 0; y\_311 = 1; z\_311 = 0;

#50

x\_311 = 0; y\_311 = 1; z\_311 = 1;

#50

x\_311 = 1; y\_311 = 0; z\_311 = 0;

#50

x\_311 = 1; y\_311 = 0; z\_311 = 1;

#50

x\_311 = 1; y\_311 = 1; z\_311 = 0;

#50

x\_311 = 1; y\_311 = 1; z\_311 = 1;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

$finish;

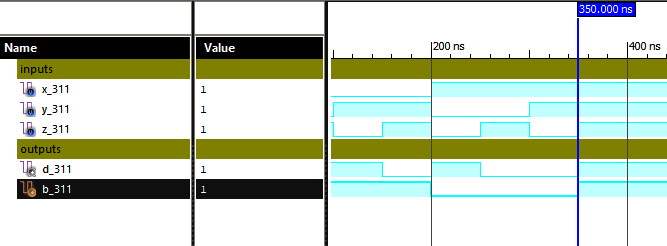
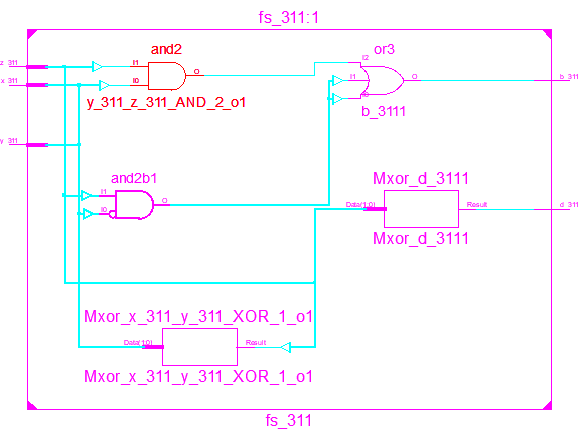
end

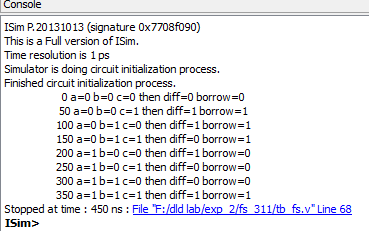
initial begin

$monitor($time," a=%b b=%b c=%b then diff=%b borrow=%b",x\_311,y\_311,z\_311,d\_311,b\_311);

end

endmodule





1. **Full adder using two half adders:**

**Verilog program:**

module fa\_2ha\_311(

input a,

input b,

input c,

output s\_311,

wire w1,w2,w3,

output cy\_311

);

ha\_311 g1(a,b,w1,w2);

ha\_311 g2(c,w1,s\_311,w3);

or g3(cy\_311,w2,w3);

endmodule

**Test Bench :**

module tb\_fa\_2ha\_311;

// Inputs

reg a;

reg b;

reg c;

// Outputs

wire s\_311;

// Instantiate the Unit Under Test (UUT)

fa\_2ha\_311 uut (

.a(a),

.b(b),

.c(c),

.s\_311(s\_311),

.cy\_311(cy\_311)

);

initial begin

// Initialize Inputs

a = 0; b = 0; c = 0;

#50

a = 0; b = 0; c = 1;

#50

a = 0; b = 1; c = 0;

#50

a = 0; b = 1; c = 1;

#50

a = 1; b = 0; c = 0;

#50

a = 1; b= 0; c= 1;

#50

a = 1; b = 1; c= 0;

#50

a = 1; b = 1; c = 1;

#100;

// Add stimulus here

$finish;

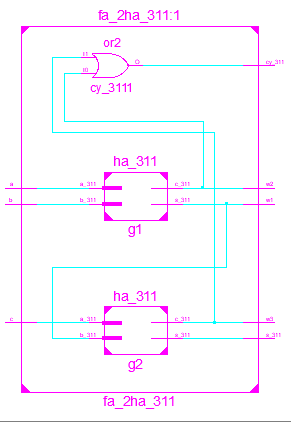
end

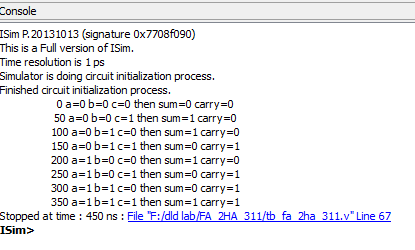
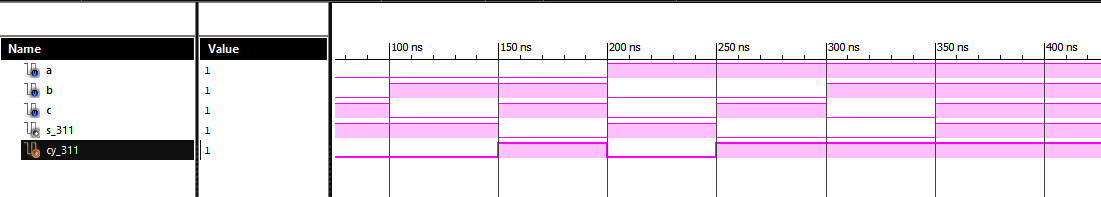
initial begin

$monitor($time," a=%b b=%b c=%b then sum=%b carry=%b",a,b,c,s\_311,cy\_311);

end

endmodule





1. **4 bit adder using 4 Full adder:**

**Verilog program:**

module bitadder\_4fa\_311(

input [3:0] a,

input [3:0] b,

input cin,

output cout\_311,

output [3:0] s\_311,

inout w1,

inout w2,

inout w3

);

fa\_311 g1(a[0],b[0],cin,s\_311[0],w1);

fa\_311 g2(w1,a[1],b[1],s\_311[1],w2);

fa\_311 g3(w2,a[2],b[2],s\_311[2],w3);

fa\_311 g4(w3,a[3],b[3],s\_311[3],cout\_311);

endmodule

**Testbench:**

module tb\_bitadder\_4fa\_311;

reg [3:0] a;

reg [3:0] b;

reg cin;

wire cout\_311; wire [3:0] s\_311;

wire w1; wire w2; wire w3;

bitadder\_4fa\_311 uut (

.a(a),

.b(b),

.cin(cin),

.cout\_311(cout\_311),

.s\_311(s\_311),

.w1(w1),

.w2(w2),

.w3(w3)

);

initial begin

$monitor($time,"a=%b b=%b cin=%b sum=%b cout=%b ",a,b,cin,s\_311,cout\_311);

// Initialize Inputs

a = 0; b = 0; cin = 0;

#10

a = 0; b = 1; cin = 0;

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

a = 1; b = 0; cin = 0;

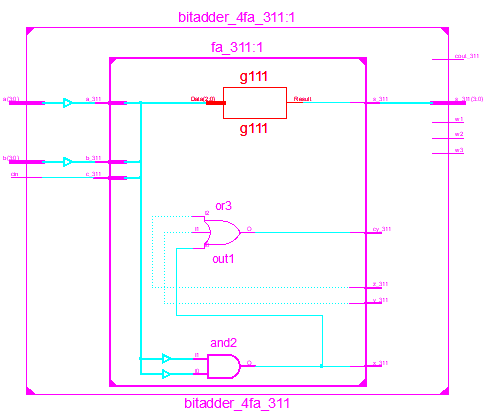
#10

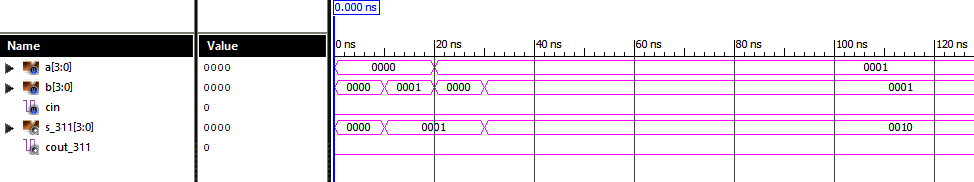
a = 1; b = 1; cin = 0;

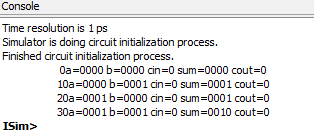
#100;

end

endmodule







**Experiment 3** : MULTIPLEXERS AND DEMULTIPLEXERS

**AIM:** TO simulate and synthesize the multiplexers and demultiplexers using Verilog hdl.

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

1. **2x1 mux:**

**Verilog Program:**

module mux2x1(

input i0\_311,

input i1\_311,

output y\_311,

input s\_311

);

assign y\_311=((i0\_311)\*(~s\_311)|(i1\_311)\*(s\_311));

endmodule

**Testbench:**

module tb\_mux2x1\_311;

reg i0\_311;

reg i1\_311;

reg s\_311;

wire y\_311;

mux2x1 uut (

.i0\_311(i0\_311),

.i1\_311(i1\_311),

.y\_311(y\_311),

.s\_311(s\_311)

);

initial begin

i0\_311 = 0; i1\_311 = 0; s\_311 = 0;

#10

i0\_311 = 1; i1\_311 = 0; s\_311 = 0;

#10

i0\_311 = 0; i1\_311 = 1; s\_311 = 1;

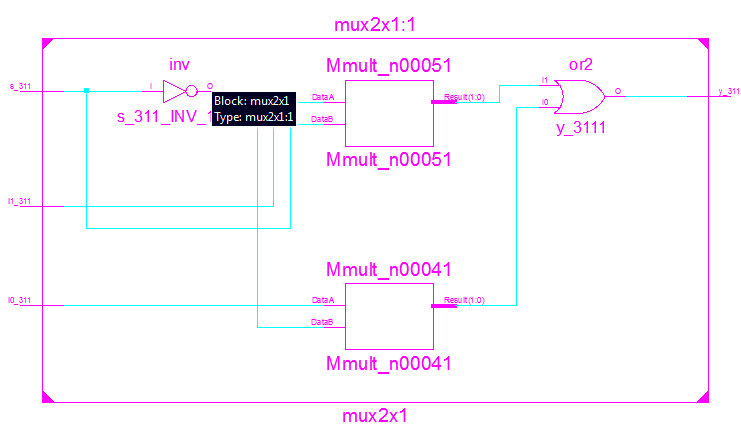
$finish;

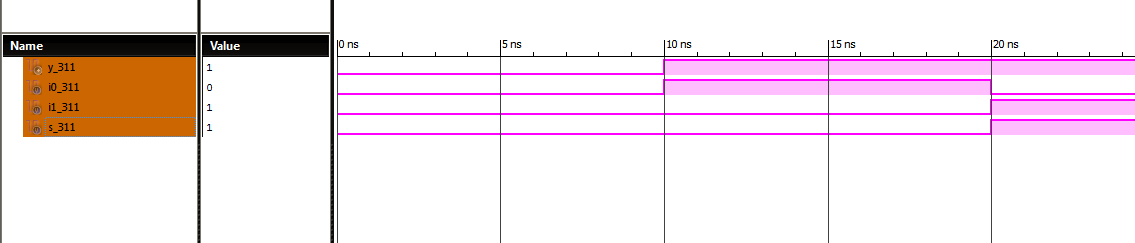
end

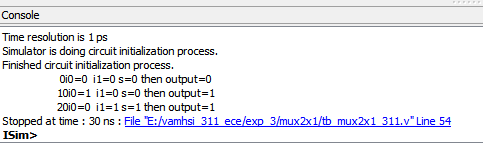
initial begin

$monitor($time,"i0=%b i1=%b s=%b then output=%b",i0\_311,i1\_311,s\_311,y\_311);

end

endmodule





**b) 4x1 mux:**

**Verilog program:**

module mux4x1\_311(

input i0\_311,

input i1\_311,

input i2\_311,

input i3\_311,

input s0\_311,

input s1\_311,

output y\_311

);

assign y\_311=(((i0\_311)&(~s1\_311)&(~s0\_311))

|((i1\_311)&(~s1\_311)&(s0\_311))

|((i2\_311)&(s1\_311)&(~s0\_311))

|((i3\_311)&(s1\_311)&(s0\_311)));

Endmodule

**Testbench:**

module tb\_mux4x1\_311;

reg i0\_311;

reg i1\_311;

reg i2\_311;

reg i3\_311;

reg s0\_311;

reg s1\_311;

wire y\_311;

mux4x1\_311 uut (

.i0\_311(i0\_311),

.i1\_311(i1\_311),

.i2\_311(i2\_311),

.i3\_311(i3\_311),

.s0\_311(s0\_311),

.s1\_311(s1\_311),

.y\_311(y\_311)

);

initial begin

i0\_311 = 1; i1\_311 = 0; i2\_311 = 1; i3\_311 = 0; s0\_311 = 0; s1\_311 = 0;

#10

i0\_311 = 1; i1\_311 = 0; i2\_311 = 1; i3\_311 = 0; s0\_311 = 1; s1\_311 = 0;

#10

i0\_311 = 1; i1\_311 = 0; i2\_311 = 1; i3\_311 = 0; s0\_311 = 0; s1\_311 = 0;

#10

i0\_311 = 1; i1\_311 = 0; i2\_311 = 1; i3\_311 = 0; s0\_311 = 1; s1\_311 = 1;

#100;

$finish;

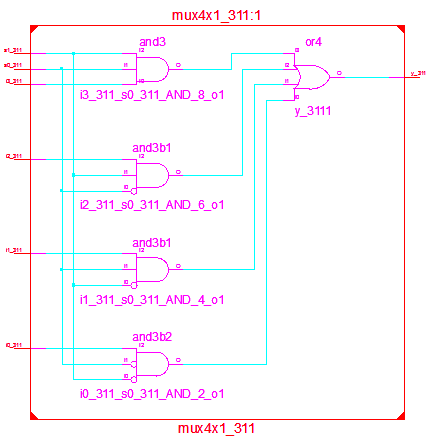
end

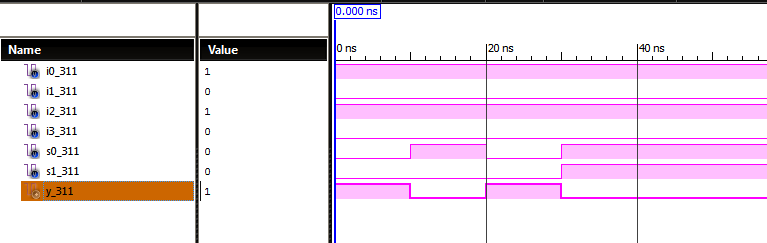
initial begin

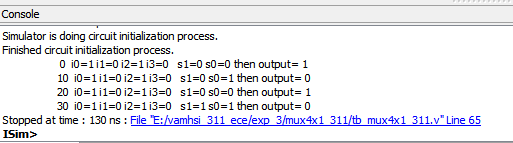
$monitor($time," i0=%d i1=%d i2=%d i3=%d s1=%d s0=%d then output= %d",i0\_311,i1\_311,i2\_311,i3\_311,s1\_311,s0\_311,y\_311);

end

endmodule







**8x1 mux using 4x1 mux:**

**Verilog program :**

module mux\_8by1\_311(

input d0\_311,d1\_311,d2\_311,d3\_311,d4\_311,d5\_311,d6\_311,d7\_311,

input s0\_311,s1\_311,s2\_311,

output y\_311

);

wire w1,w2;

mux\_4by1\_311 g1(d0\_311,d1\_311,d2\_311,d3\_311,s1\_311,s0\_311,w1);

mux\_4by1\_311 g2(d4\_311,d5\_311,d6\_311,d7\_311,s1\_311,s0\_311,w2);

mux\_2by1\_311 g3(w1,w2,s2\_311,y\_311);

endmodule

**Test bench :**

module test\_mux\_8by1\_311;

reg d0\_311;

reg d1\_311;

reg d2\_311;

reg d3\_311;

reg d4\_311;

reg d5\_311;

reg d6\_311;

reg d7\_311;

reg s0\_311;

reg s1\_311;

reg s2\_311;

wire y\_311;

mux\_8by1\_311 uut (

.d0\_311(d0\_311),

.d1\_311(d1\_311),

.d2\_311(d2\_311),

.d3\_311(d3\_311),

.d4\_311(d4\_311),

.d5\_311(d5\_311),

.d6\_311(d6\_311),

.d7\_311(d7\_311),

.s0\_311(s0\_311),

.s1\_311(s1\_311),

.s2\_311(s2\_311),

.y\_311(y\_311)

);

initial begin

$monitor($time,"ns s2=%b s1=%b s0=%b d0=%b d1=%b d2=%b d3=%b d4=%b d5=%b d6=%b d7=%b y=%b",s2\_311,s1\_311,s0\_311,d0\_311,d1\_311,d2\_311,d3\_311,d4\_311,d5\_311,d6\_311,d7\_311,y\_311);

d0\_311 = 1; d1\_311 = 0; d2\_311 = 0; d3\_311 = 0; d4\_311 = 0;

d5\_311 = 0; d6\_311 = 0; d7\_311 = 0; s0\_311 = 0 s1\_311 = 0;

s2\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 1;

d2\_311 = 0;

d3\_311 = 0;

d4\_311 = 0;

d5\_311 = 0;

d6\_311 = 0;

d7\_311 = 0;

s0\_311 = 1;

s1\_311 = 0;

s2\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 0;

d2\_311 = 1;

d3\_311 = 0;

d4\_311 = 0;

d5\_311 = 0;

d6\_311 = 0;

d7\_311 = 0;

s0\_311 = 0;

s1\_311 = 1;

s2\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 0;

d2\_311 = 0;

d3\_311 = 1;

d4\_311 = 0;

d5\_311 = 0;

d6\_311 = 0;

d7\_311 = 0;

s0\_311 = 1;

s1\_311 = 1;

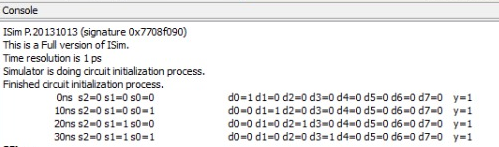
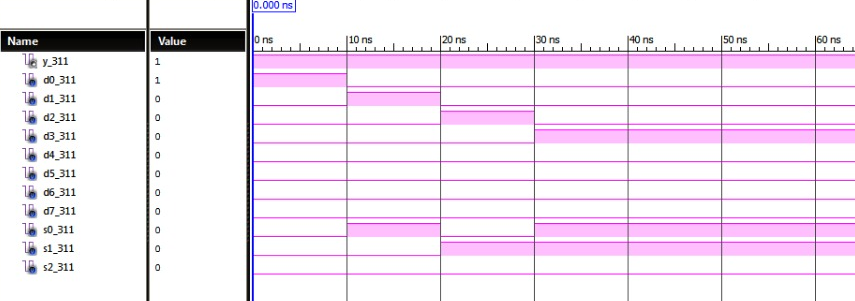
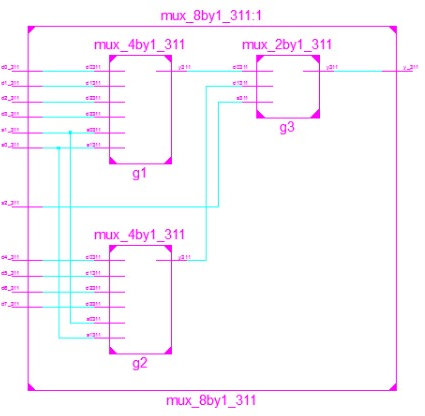
s2\_311 = 0;

#10

#100;

end

endmodule



**1x4 De-mux:**

**Verilog program :**

module demux1x4\_311(

input i,

input s0,

input s1,

output y0\_311,

output y1\_311,

output y2\_311,

output y3\_311

);

assign y0\_311=((~s1)&(~s0)&(i));

assign y1\_311=((~s1)&(s0)&(i));

assign y2\_311=((s1)&(~s0)&(i));

assign y3\_311=((s1)&(s0)&(i));

endmodule

**Test bench :**

module tb\_demux4x1\_311;

reg i;

reg s0;

reg s1;

wire y0\_311;

wire y1\_311;

wire y2\_311;

wire y3\_311;

demux1x4\_311 uut (

.i(i),

.s0(s0),

.s1(s1),

.y0\_311(y0\_311),

.y1\_311(y1\_311),

.y2\_311(y2\_311),

.y3\_311(y3\_311)

);

initial begin

$monitor($time,"i =%b s0=%b s1=%b then y0=%b y1=%b y2=%b y3=%b",i,s0,s1,y0\_311,y1\_311,y2\_311,y3\_311);

i = 1; s0 = 0; s1 = 0;

#5

i = 1; s0 = 0; s1 = 1;

#5

i = 1; s0 = 1; s1 = 0;

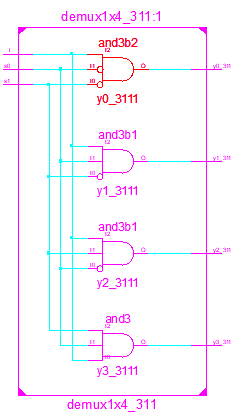
#5

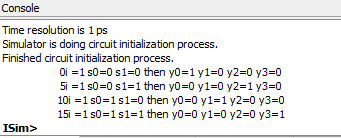
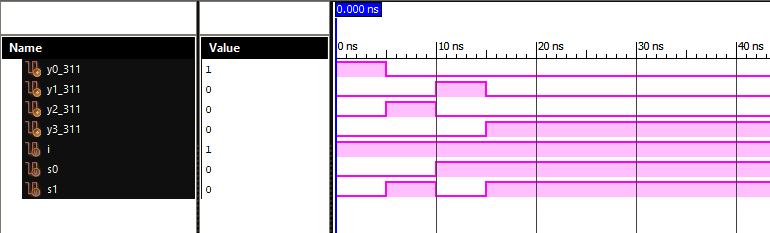
i = 1; s0 = 1; s1 = 1;

#5;

end

endmodule





**Experiment 4** : ENCODERS DECODERS ,PRIORITY AND COMPAEATOR

**AIM:** TO simulate and synthesize the encoder and decoder using Verilog hdl also priority encoder,comparator.

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

1. **4 to 2 encoder :**

**Verilog Program:**

module encoder4to2(

input d0,

input d1,

input d2,

input d3,

output a\_311,

output b\_311

);

assign a\_311 =((~d3)&(d2)&(~d1)&(~d0)|(d3)&(~d2)&(~d1)&(~d0));

assign b\_311 =((~d3)&(~d2)&(d1)&(~d0)|(d3)&(~d2)&(~d1)&(~d0));

endmodule

**Test bench :**

module tb\_encoder4to2\_311;

reg d0; reg d1;

reg d2; reg d3;

wire a\_311; wire b\_311;

encoder4to2 uut (

.d0(d0),

.d1(d1),

.d2(d2),

.d3(d3),

.a\_311(a\_311), .b\_311(b\_311)

);

initial begin

$monitor($time,"d0=%b d1=%b d2=%b d3=%b a=%b b=%b",d0,d1,d2,d3,a\_311,b\_311);

d0 = 1; d1 = 0; d2 = 0; d3 = 0;

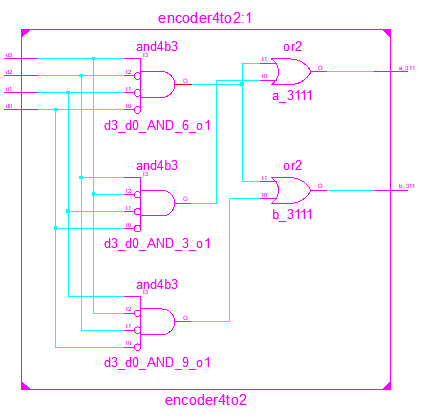
#10 d0 = 0; d1 = 1; d2 = 0; d3 = 0;

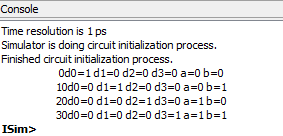
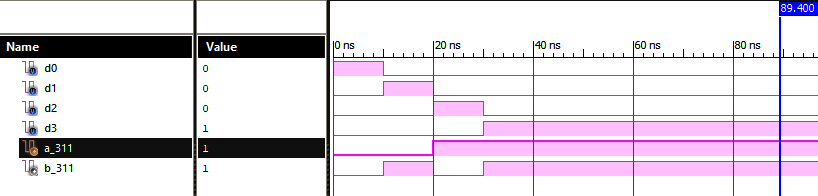
#10 d0 = 0; d1 = 0; d2 = 1; d3 = 0;

#10 d0 = 0; d1 = 0; d2 = 0; d3 = 1;

#100 ; end

endmodule

****



1. **2 to 4 decoder :**

**Verilog Program:**

module enable2to4decoder\_311(

input e,

input a,

input b,

output yo\_311,

output y1\_311,

output y2\_311,

output y3\_311

);

assign yo\_311=((!e)&(!a)&(!b));

assign y1\_311=((!e)&(a)&(!b));

assign y2\_311=((!e)&(!a)&(b));

assign y3\_311=((!e)&(a)&(b));

endmodule

**Test bench :**

module tb\_enable2to4decoder\_311;

reg e;

reg a;

reg b;

wire yo\_311;

wire y1\_311;

wire y2\_311;

wire y3\_311;

enable2to4decoder\_311 uut (

.e(e),

.a(a),

.b(b),

.yo\_311(yo\_311),

.y1\_311(y1\_311),

.y2\_311(y2\_311),

.y3\_311(y3\_311)

);

initial begin

e = 1; a = 0; b = 0;

#5

e = 1; a = 0; b = 1;

#5

e = 1; a = 1; b = 0;

#5

e = 1; a = 1; b = 1;

#5

e = 0; a = 0; b = 0;

#5

e = 0; a = 0; b = 1;

#5 e = 0; a = 1; b = 0;

#5 e = 0; a = 1; b = 1;

#10

$finish;

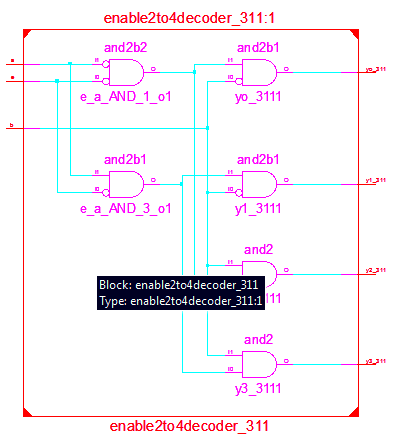
end

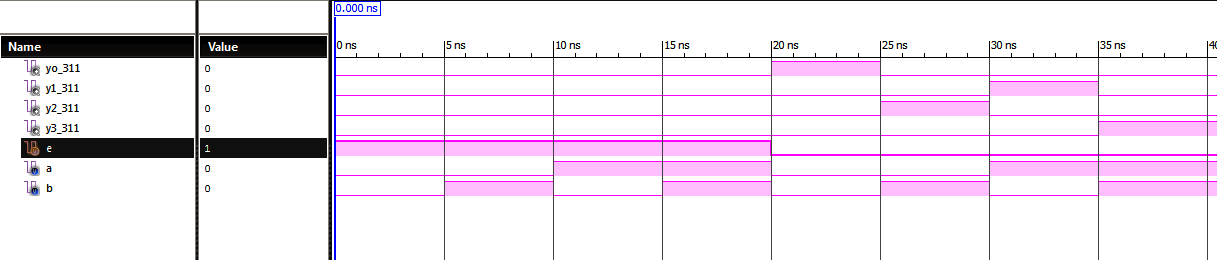
initial begin

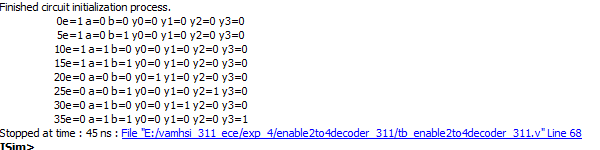
$monitor($time,"e=%b a=%b b=%b y0=%b y1=%b y2=%b y3=%b",e,a,b,yo\_311,y1\_311,y2\_311,y3\_311);

End

endmodule







1. **3 to 8 decoder :**

**Verilog Program:**

module decoder3to8\_311(

input a,

input b,

input c,

output y0\_311,

output y1\_311,

output y2\_311,

output y3\_311,

output y4\_311,

output y5\_311,

output y6\_311,

output y7\_311

);

wire w1;

not g1(w1,a);

enable2to4decoder\_311 g2(a,b,c,y0\_311,y1\_311,y2\_311,y3\_311);

enable2to4decoder\_311 g3(w1,b,c,y4\_311,y5\_311,y6\_311,y7\_311);

endmodule

**Test bench :**

module tb\_decoder3x8\_311;

reg a;

reg b;

reg c;

wire y0\_311;

wire y1\_311;

wire y2\_311;

wire y3\_311;

wire y4\_311;

wire y5\_311;

wire y6\_311;

wire y7\_311;

decoder3to8\_311 uut (

.a(a),

.b(b),

.c(c),

.y0\_311(y0\_311),

.y1\_311(y1\_311),

.y2\_311(y2\_311),

.y3\_311(y3\_311),

.y4\_311(y4\_311),

.y5\_311(y5\_311),

.y6\_311(y6\_311),

.y7\_311(y7\_311)

);

initial begin

$monitor($time," e=%d b=%b c=%b yo=%b y1=%b y2=%b y3=%b y4=%b y5=%b y6=%b y7=%b",a,b,c,y0\_311,y1\_311,y2\_311,y3\_311,y4\_311,y5\_311,y6\_311,y7\_311);

a = 0; b = 0; c = 0;

#10

a = 0; b = 0; c = 1;

#10

a = 0; b = 1; c = 0;

#10

a = 0; b = 1; c = 1;

#10

a = 1; b = 0; c = 0;

#10

a = 1; b = 0; c = 1;

#10

a = 1; b = 1; c = 0;

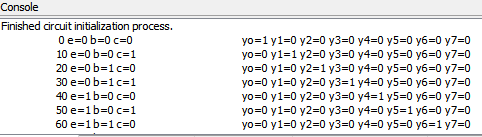
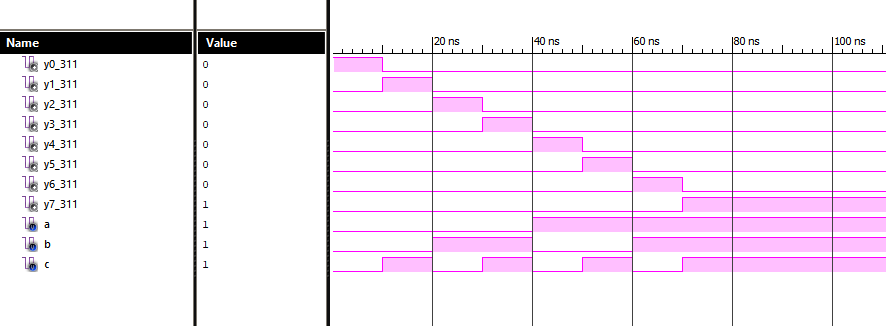
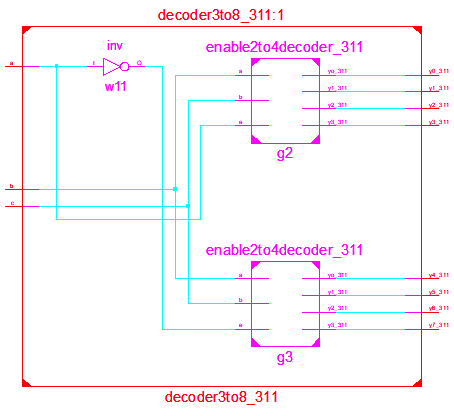
#10

a = 1; b = 1; c = 1;

#10 ;

end

endmodule



1. **priority encoder:**

**Verilog program :**

module ptiority\_311(

input d0\_311,d1\_311,d2\_311,d3\_311,

output a\_311,b\_311,v\_311

);

wire w1,w2;

or g1(a\_311,d2\_311,d3\_311);

or g2(w1,~d2\_311,d3\_311);

or g3(w2,d1\_311,d3\_311);

and g4(b\_311,w1,w2);

or g5(v\_311,d0\_311,d1\_311,d2\_311,d3\_311);

endmodule

**Test bench :**

module tb\_priority\_311;

// Inputs

reg d0\_311;

reg d1\_311;

reg d2\_311;

reg d3\_311;

// Outputs

wire a\_311;

wire b\_311;

wire v\_311;

// Instantiate the Unit Under Test (UUT)

ptiority\_311 uut (

.d0\_311(d0\_311),

.d1\_311(d1\_311),

.d2\_311(d2\_311),

.d3\_311(d3\_311),

.a\_311(a\_311),

.b\_311(b\_311),

.v\_311(v\_311)

);

initial begin

$monitor($time,"ns d0=%b d1=%b d2=%b d3=%b a=%b b=%b v=%b",d0\_311,d1\_311,d2\_311,d3\_311,a\_311,b\_311,v\_311);

// Initialize Inputs

d0\_311 = 0;

d1\_311 = 0;

d2\_311 = 0;

d3\_311 = 0;

#10

d0\_311 = 1;

d1\_311 = 0;

d2\_311 = 0;

d3\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 1;

d2\_311 = 0;

d3\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 0;

d2\_311 = 1;

d3\_311 = 0;

#10

d0\_311 = 0;

d1\_311 = 0;

d2\_311 = 0;

d3\_311 = 1;

#10

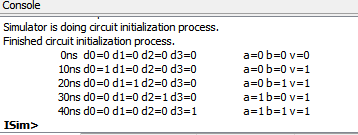
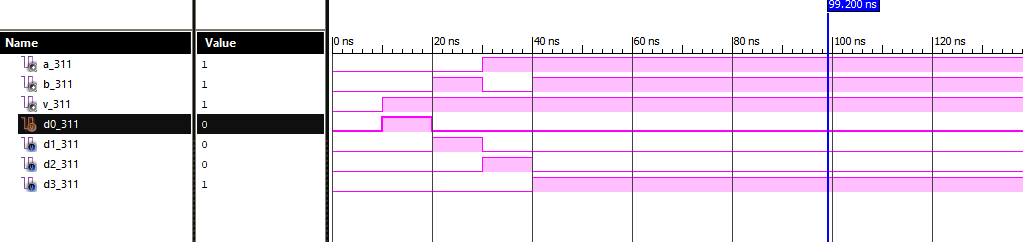
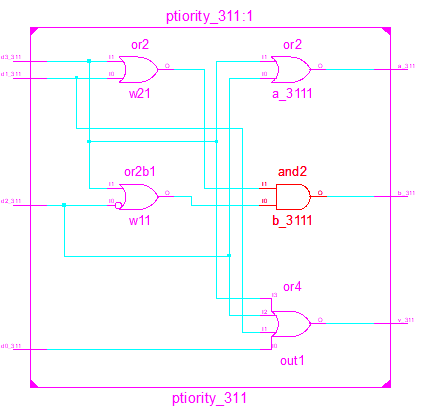
// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

end

endmodule



1. **1-bit comparator:**

**Verilog program :**

module comparator\_311(

input a311,b311,gt311,lt311,eq311,

output aGTb311,aLTb311,aEQb311

);

assign aGTb311 = (gt311&(~b311))|(gt311&a311)|(a311&(~b311));

assign aLTb311 = (lt311&(~a311))|(lt311&b311)|((~a311)&b311);

assign aEQb311 = ((eq311)&(~a311)&(~b311))|((eq311)&(a311)&(b311)) ;

endmodule

**Test bench :**

module tb\_comparator\_311;

// Inputs

reg a311;

reg b311;

reg gt311;

reg lt311;

reg eq311;

// Outputs

wire aGTb311;

wire aLTb311;

wire aEQb311;

// Instantiate the Unit Under Test (UUT)

comparator\_311 uut (

.a311(a311),

.b311(b311),

.gt311(gt311),

.lt311(lt311),

.eq311(eq311),

.aGTb311(aGTb311),

.aLTb311(aLTb311),

.aEQb311(aEQb311)

);

initial begin

$monitor($time,"a=%b b=%b agtb=%b altb=%b aeqb=%b",a311,b311,aGTb311,aLTb311,aEQb311);

a311 = 0; b311 = 0; gt311 = 0; lt311 = 0; eq311 = 1;

#10

a311 = 0; b311 = 1; gt311 = 0; lt311 = 1; eq311 = 0;

#10

a311 = 1; b311 = 0; gt311 = 1; lt311 = 0; eq311 = 0;

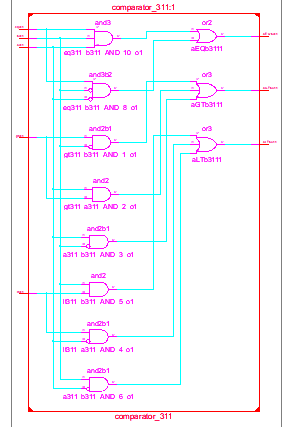
#10

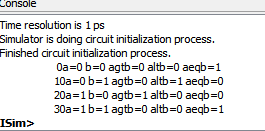
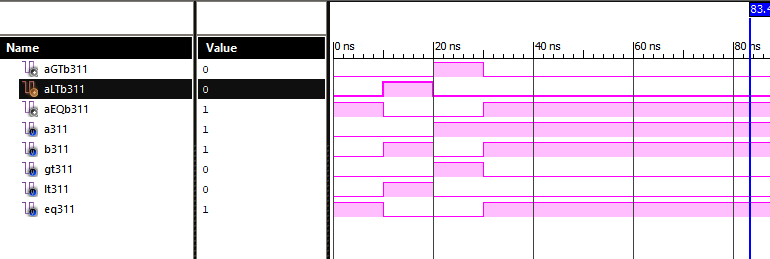
a311 = 1; b311 = 1; gt311 = 0; lt311 = 0; eq311 = 1;

#10;

end

endmodule





1. **4-bit comparator using 1-bit comparator:**

**Verilog program :**

module comparator4bit\_311(

input [3:0]a\_311,b\_311,

input gt\_311,lt\_311,eq\_311,

output aGTb\_311,aLTb\_311,aEQb\_311

);

wire gt1,gt2,gt3,lt1,lt2,lt3,eq1,eq2,eq3;

comparator\_311 C1(a\_311[0],b\_311[0],gt\_311,lt\_311,eq\_311,gt1,lt1,eq1);

comparator\_311 C2(a\_311[1],b\_311[1],gt1,lt1,eq1,gt2,lt2,eq2);

comparator\_311 C3(a\_311[2],b\_311[2],gt2,lt2,eq2,gt3,lt3,eq3);

comparator\_311 C4(a\_311[3],b\_311[3],gt3,lt3,eq3,aGTb\_311,aLTb\_311,aEQb\_311);

endmodule

**Test bench :**

module tb\_comparator4bit\_311;

// Inputs

reg [3:0] a\_311;

reg [3:0] b\_311;

reg gt\_311;

reg lt\_311;

reg eq\_311;

// Outputs

wire aGTb\_311;

wire aLTb\_311;

wire aEQb\_311;

// Instantiate the Unit Under Test (UUT)

comparator4bit\_311 uut (

.a\_311(a\_311),

.b\_311(b\_311),

.gt\_311(gt\_311),

.lt\_311(lt\_311),

.eq\_311(eq\_311),

.aGTb\_311(aGTb\_311),

.aLTb\_311(aLTb\_311),

.aEQb\_311(aEQb\_311)

);

initial begin

$monitor($time,"ns a=%b b=%b gt=%b lt=%b eq=%b aGTb=%b aLTb=%b aEQb=%b",a\_311,b\_311,gt\_311,lt\_311,eq\_311,aGTb\_311,aLTb\_311,aEQb\_311);

// Initialize Inputs

a\_311 = 1;

b\_311 = 2;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 1;

#10

a\_311 = 2;

b\_311 = 1;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 1;

#10

a\_311 = 2;

b\_311 = 2;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 1;

#10

a\_311 = 0;

b\_311 = 0;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 1;

#10

a\_311 = 1;

b\_311 = 2;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 0;

#10

a\_311 = 2;

b\_311 = 1;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 0;

#10

a\_311 = 2;

b\_311 = 2;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 0;

#10

a\_311 = 0;

b\_311 = 0;

gt\_311 = 0;

lt\_311 = 0;

eq\_311 = 0;

#10

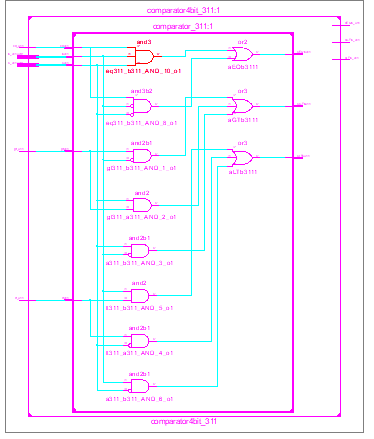
// Wait 100 ns for global reset to finish

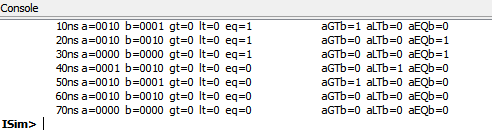
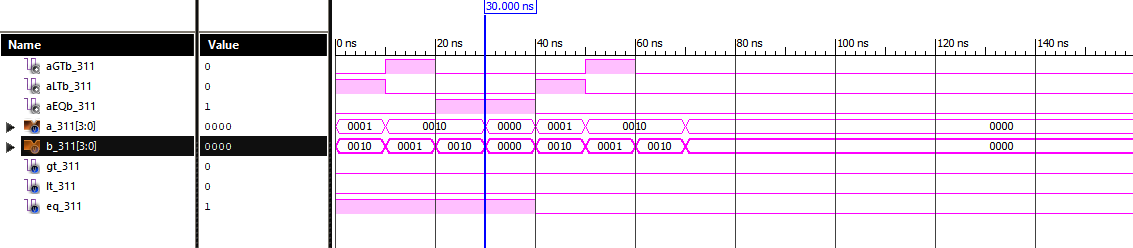
#100;

// Add stimulus here

end

endmodule





1. **4-bit comparator using 1-bit comparator:**

**Verilog program :**

module comp4311(

input a0,a1,a2,a3,

input b0,b1,b2,b3,

input gt,

input lt,

input eq,

output agtb,

output altb,

output aeqb

);

wire [8:0]w;

comp1bit c0(a0,b0,gt,lt,eq,w[0],w[1],w[2]);

comp1bit c1(a1,b1,w[0],w[1],w[2],w[3],w[4],w[5]);

comp1bit c2(a2,b2,w[3],w[4],w[5],w[6],w[7],w[8]);

comp1bit c3(a3,b3,w[6],w[7],w[8],agtb,altb,aeqb);

endmodule

**Test bench :**

module tb\_comp4311;

reg a0;

reg a1;

reg a2;

reg a3;

reg b0;

reg b1;

reg b2;

reg b3;

reg gt;

reg lt;

reg eq;

wire agtb;

wire altb;

wire aeqb;

comp4311 uut (

.a0(a0),

.a1(a1),

.a2(a2),

.a3(a3),

.b0(b0),

.b1(b1),

.b2(b2),

.b3(b3),

.gt(gt),

.lt(lt),

.eq(eq),

.agtb(agtb),

.altb(altb),

.aeqb(aeqb)

);

initial begin

a0 = 1;

a1 = 1;

a2 = 1;

a3 = 0;

b0 = 1;

b1 = 0;

b2 = 0;

b3 = 1;

gt = 0;

lt = 1;

eq = 0;

#10

a0 = 1;

a1 = 1;

a2 = 1;

a3 = 1;

b0 = 1;

b1 = 1;

b2 = 0;

b3 = 1;

gt = 1;

lt = 0;

eq = 0;

#10

a0 = 1;

a1 = 0;

a2 = 1;

a3 = 0;

b0 = 1;

b1 = 0;

b2 = 1;

b3 = 0;

gt = 0;

lt = 0;

eq = 1;

#10

$finish;

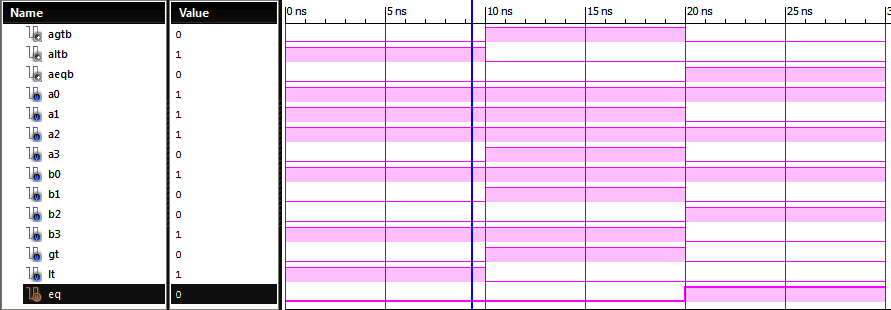
end

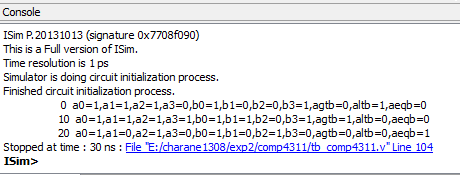
initial begin

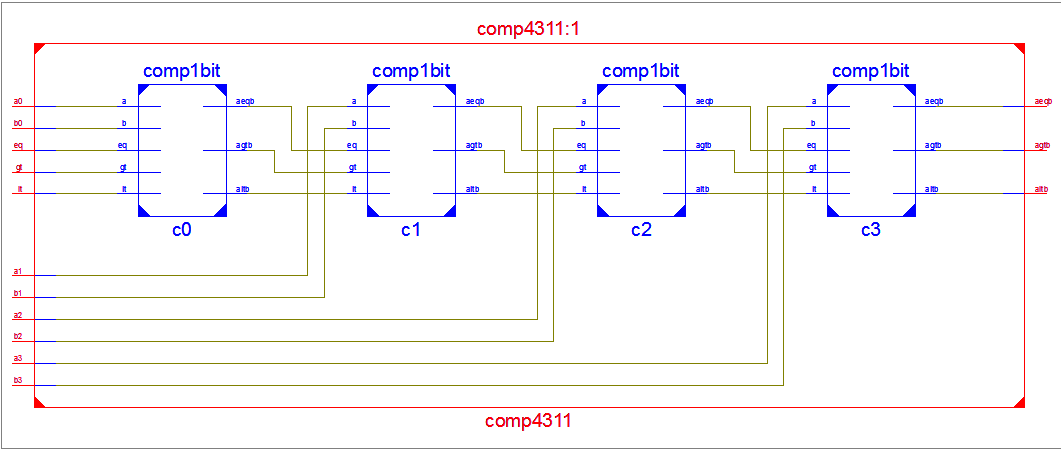
$monitor($time, " a0=%b,a1=%b,a2=%b,a3=%b,b0=%b,b1=%b,b2=%b,b3=%b,agtb=%b,altb=%b,aeqb=%b",a0,a1,a2,a3,b0,b1,b2,b3,agtb,altb,aeqb);

end

endmodule







**Logical functions realisation using mux and decoders**

1. **Full adder using 3 to 8 decoder:**

**Verilog program :**

module fa\_decoder\_311(

input a,

input b,

input c,

inout y0\_311,

inout y1\_311,

inout y2\_311,

inout y3\_311,

inout y4\_311,

inout y5\_311,

inout y6\_311,

inout y7\_311,

output s\_311,

output cy\_311

);

decoder3to8\_311 g1(a,b,c,y0\_311,y1\_311,y2\_311,y3\_311,y4\_311,y5\_311,y6\_311,y7\_311);

or g2(s\_311,y1\_311,y2\_311,y4\_311,y7\_311);

or g3(cy\_311,y3\_311,y5\_311,y6\_311,y7\_311);

endmodule

**Test bench :**

module tb\_fa\_decoder\_311;

reg a;

reg b;

reg c;

wire s\_311;

wire cy\_311;

wire y0\_311;

wire y1\_311;

wire y2\_311;

wire y3\_311;

wire y4\_311;

wire y5\_311;

wire y6\_311;

wire y7\_311;

fa\_decoder\_311 uut (

.a(a),

.b(b),

.c(c),

.y0\_311(y0\_311),

.y1\_311(y1\_311),

.y2\_311(y2\_311),

.y3\_311(y3\_311),

.y4\_311(y4\_311),

.y5\_311(y5\_311),

.y6\_311(y6\_311),

.y7\_311(y7\_311),

.s\_311(s\_311),

.cy\_311(cy\_311)

);

initial begin

$monitor($time,"a=%b b=%b c=%b : sum=%b,carry=%b",a,b,c,s\_311,cy\_311);

a = 0; b = 0; c = 0;

#20 a = 0; b = 0; c = 1;

#20 a = 0; b = 1; c = 0;

#20 a = 0; b = 1; c = 1;

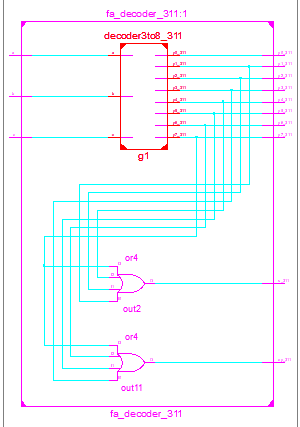
#20 a = 1; b = 0; c = 1;

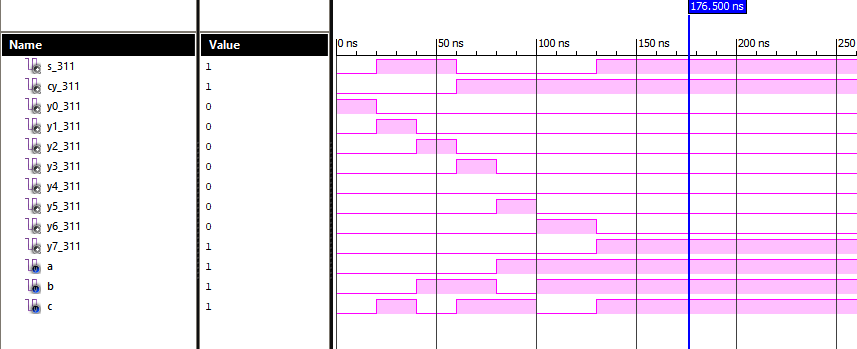
#20 a = 1; b = 1; c = 0;

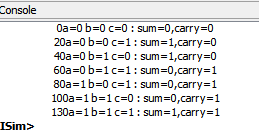
#30 a = 1; b = 1; c = 1;

end

endmodule







1. **Full adder using mux:**

**Verilog program :**

module fa\_mux\_311(

input a,

input b,

input c,

inout c1,

output s\_311,

output cy\_311

);

not g3(c1,c);

mux4x1\_311 g1(c,c1,c1,c,a,b,s\_311);

mux4x1\_311 g2(0,c,c,1,a,b,cy\_311);

endmodule

**Test bench :**

module tb\_fa\_mux\_311;

reg a;

reg b;

reg c;

wire s\_311;

wire cy\_311;

wire c1;

fa\_mux\_311 uut (

.a(a),

.b(b),

.c(c),

.c1(c1),

.s\_311(s\_311),

.cy\_311(cy\_311)

);

initial begin

$monitor($time,"a=%b b=%b c=%b : sum=%b,carry=%b",a,b,c,s\_311,cy\_311);

a = 0;

b = 0;

c = 0;

#20

a = 0;

b = 0;

c = 1;

#20

a = 0;

b = 1;

c = 0;

#20

a = 0;

b = 1;

c = 1;

#20

a = 1;

b = 0;

c = 1;

#20

a = 1;

b = 1;

c = 0;

#30

a = 1;

b = 1;

c = 1;

#100;

end

endmodule

