**Q1) 4-bit Incrementor with Top level entity :-** four\_bit\_incrementer

module andgate( output c, input a, b);

wire c\_;

nand n1 (c\_, a ,b);

nand n2 (c, c\_, c\_);

endmodule

module orgate(output c, input a, b);

wire a\_, b\_;

nand n3 (a\_, a, a);

nand n4 (b\_, b, b);

nand n5 (c, a\_, b\_);

endmodule

module xorgate(output c, input a, b);

wire a\_, b\_, w1, w2;

nand n6 (a\_, a, a);

nand n7 (b\_, b, b);

andgate a1 (w1, a\_, b);

andgate a2 (w2, a, b\_);

orgate o1 (c, w1, w2);

endmodule

module half\_add (input a, b, output s, c);

xorgate g1(s, a, b);

andgate g2(c, a, b);

endmodule

module full\_add ( input a, b, cin, output s, c);

wire wire1, wire2, wire3;

half\_add ha1 ( a, b, wire1, wire2);

half\_add ha2 ( wire1, cin, s, wire3);

orgate or1 (c, wire2, wire3);

endmodule

module four\_bit\_incrementer (input [3:0] a, input cin, output [4:0] s);

wire c1, c2, c3;

full\_add fa1 (a[0], 1'b1, 1'b0, s[0], c1);

full\_add fa2 (a[1], 1'b0, c1, s[1], c2);

full\_add fa3 (a[2], 1'b0, c2, s[2], c3);

full\_add fa4 (a[3], 1'b0, c3, s[3], s[4]);

endmodule

**Test bench for 4-bit incrementor:**

module four\_bit\_incrementer\_test;

reg [3:0] a;

reg cin;

wire [4:0] s;

four\_bit\_incrementer gate1 (a, cin, s);

integer i;

initial begin

for (i = 0; i < 16; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q2) 4-bit Decrementer with Top Level entity: -** four\_bit\_decrementer

module andgate( output c, input a, b);

wire c\_;

nand n1 (c\_, a ,b);

nand n2 (c, c\_, c\_);

endmodule

module orgate(output c, input a, b);

wire a\_, b\_;

nand n3 (a\_, a, a);

nand n4 (b\_, b, b);

nand n5 (c, a\_, b\_);

endmodule

module xorgate(output c, input a, b);

wire a\_, b\_, w1, w2;

nand n6 (a\_, a, a);

nand n7 (b\_, b, b);

andgate a1 (w1, a\_, b);

andgate a2 (w2, a, b\_);

orgate o1 (c, w1, w2);

endmodule

module half\_add (input a, b, output s, c);

xorgate g1(s, a, b);

andgate g2(c, a, b);

endmodule

module full\_add ( input a, b, cin, output s, c);

wire wire1, wire2, wire3;

half\_add ha1 ( a, b, wire1, wire2);

half\_add ha2 ( wire1, cin, s, wire3);

orgate or1 (c, wire2, wire3);

endmodule

module four\_bit\_decrementer (input [3:0] a, input cin, output [4:0] s);

wire c1, c2, c3;

full\_add fa1 (a[0], 1'b1, 1'b0, s[0], c1);

full\_add fa2 (a[1], 1'b1, c1, s[1], c2);

full\_add fa3 (a[2], 1'b1, c2, s[2], c3);

full\_add fa4 (a[3], 1'b1, c3, s[3], s[4]);

endmodule

**Test bench for 4 – bit Decrementer:**

module four\_bit\_decrementer\_test;

reg [3:0] a;

reg cin;

wire [4:0] s;

four\_bit\_decrementer gate1 (a, cin, s);

integer i;

initial begin

for (i = 0; i < 16; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q3) 16 – bit incrementer with top level entity :-** sixteen\_bit\_incrementer

module andgate( output c, input a, b);

wire c\_;

nand n1 (c\_, a ,b);

nand n2 (c, c\_, c\_);

endmodule

module orgate(output c, input a, b);

wire a\_, b\_;

nand n3 (a\_, a, a);

nand n4 (b\_, b, b);

nand n5 (c, a\_, b\_);

endmodule

module xorgate(output c, input a, b);

wire a\_, b\_, w1, w2;

nand n6 (a\_, a, a);

nand n7 (b\_, b, b);

andgate a1 (w1, a\_, b);

andgate a2 (w2, a, b\_);

orgate o1 (c, w1, w2);

endmodule

module half\_add (input a, b, output s, c);

xorgate g1(s, a, b);

andgate g2(c, a, b);

endmodule

module full\_add ( input a, b, cin, output s, c);

wire wire1, wire2, wire3;

half\_add ha1 ( a, b, wire1, wire2);

half\_add ha2 ( wire1, cin, s, wire3);

orgate or1 (c, wire2, wire3);

endmodule

module sixteen\_bit\_incrementer (input [15:0] a, input cin, output [16:0] s);

wire c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15;

full\_add fa1 (a[0], 1'b1, 1'b0, s[0], c1);

full\_add fa2 (a[1], 1'b0, c1, s[1], c2);

full\_add fa3 (a[2], 1'b0, c2, s[2], c3);

full\_add fa4 (a[3], 1'b0, c3, s[3], c4);

full\_add fa5 (a[4], 1'b0, c4, s[4], c5);

full\_add fa6 (a[5], 1'b0, c5, s[5], c6);

full\_add fa7 (a[6], 1'b0, c6, s[6], c7);

full\_add fa8 (a[7], 1'b0, c7, s[7], c8);

full\_add fa9 (a[8], 1'b0, c8, s[8], c9);

full\_add fa10 (a[9], 1'b0, c9, s[9], c10);

full\_add fa11 (a[10], 1'b0, c10, s[10], c11);

full\_add fa12 (a[11], 1'b0, c3, s[11], c12);

full\_add fa13 (a[12], 1'b0, c11, s[12], c13);

full\_add fa14 (a[13], 1'b0, c12, s[13], c14);

full\_add fa15 (a[14], 1'b0, c13, s[14], c15);

full\_add fa16 (a[15], 1'b0, c14, s[15], s[16]);

endmodule

**Testbench for 16 – bit incrementer:**

module sixteen\_bit\_incrementer\_test;

reg [15:0] a;

reg cin;

wire [16:0] s;

sixteen\_bit\_incrementer gate1 (a, cin, s);

integer i;

initial begin

for (i = 0; i < 65536; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q4) 16 – bit Decrementer with top level entity :-**

module andgate( output c, input a, b);

wire c\_;

nand n1 (c\_, a ,b);

nand n2 (c, c\_, c\_);

endmodule

module orgate(output c, input a, b);

wire a\_, b\_;

nand n3 (a\_, a, a);

nand n4 (b\_, b, b);

nand n5 (c, a\_, b\_);

endmodule

module xorgate(output c, input a, b);

wire a\_, b\_, w1, w2;

nand n6 (a\_, a, a);

nand n7 (b\_, b, b);

andgate a1 (w1, a\_, b);

andgate a2 (w2, a, b\_);

orgate o1 (c, w1, w2);

endmodule

module half\_add (input a, b, output s, c);

xorgate g1(s, a, b);

andgate g2(c, a, b);

endmodule

module full\_add ( input a, b, cin, output s, c);

wire wire1, wire2, wire3;

half\_add ha1 ( a, b, wire1, wire2);

half\_add ha2 ( wire1, cin, s, wire3);

orgate or1 (c, wire2, wire3);

endmodule

module sixteen\_bit\_decrementer (input [15:0] a, input cin, output [16:0] s);

wire c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15;

full\_add fa1 (a[0], 1'b1, 1'b0, s[0], c1);

full\_add fa2 (a[1], 1'b1, c1, s[1], c2);

full\_add fa3 (a[2], 1'b1, c2, s[2], c3);

full\_add fa4 (a[3], 1'b1, c3, s[3], c4);

full\_add fa5 (a[4], 1'b1, c4, s[4], c5);

full\_add fa6 (a[5], 1'b1, c5, s[5], c6);

full\_add fa7 (a[6], 1'b1, c6, s[6], c7);

full\_add fa8 (a[7], 1'b1, c7, s[7], c8);

full\_add fa9 (a[8], 1'b1, c8, s[8], c9);

full\_add fa10 (a[9], 1'b1, c9, s[9], c10);

full\_add fa11 (a[10], 1'b1, c10, s[10], c11);

full\_add fa12 (a[11], 1'b1, c3, s[11], c12);

full\_add fa13 (a[12], 1'b1, c11, s[12], c13);

full\_add fa14 (a[13], 1'b1, c12, s[13], c14);

full\_add fa15 (a[14], 1'b1, c13, s[14], c15);

full\_add fa16 (a[15], 1'b1, c14, s[15], s[16]);

endmodule

**Test bench for 16-bit Incrementer :-**

module sixteen\_bit\_decrementer\_test;

reg [15:0] a;

reg cin;

wire [16:0] s;

sixteen\_bit\_decrementer gate1 (a, cin, s);

integer i;

initial begin

for (i = 0; i < 65536; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q5) 4-bit negator with top level entity :-** four\_bit\_negator

module notgate(output b, input a);

nand n1 (b, a, a);

endmodule

module four\_bit\_negator(input [3:0] a, output [3:0] b);

notgate g1(b[0],a[0]);

notgate g2(b[1],a[1]);

notgate g3(b[2],a[2]);

notgate g4(b[3],a[3]);

endmodule

**Test bench for 4 – bit negator :-**

module four\_bit\_negator\_test;

reg [3:0] a;

wire [3:0] b;

four\_bit\_negator gate1 (a, b);

integer i;

initial begin

for (i = 0; i < 16; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q6) 16 bit negator with top level entity :-** sixteen\_bit\_negator

module notgate(output b, input a);

nand n1 (b, a, a);

endmodule

module sixteen\_bit\_negator(input [15:0] a, output [15:0] b);

notgate g1(b[0],a[0]);

notgate g2(b[1],a[1]);

notgate g3(b[2],a[2]);

notgate g4(b[3],a[3]);

notgate g5(b[4],a[4]);

notgate g6(b[5],a[5]);

notgate g7(b[6],a[6]);

notgate g8(b[7],a[7]);

notgate g9(b[8],a[8]);

notgate g10(b[9],a[9]);

notgate g11(b[10],a[10]);

notgate g12(b[11],a[11]);

notgate g13(b[12],a[12]);

notgate g14(b[13],a[13]);

notgate g15(b[14],a[14]);

notgate g16(b[15],a[15]);

endmodule

**Testbench for 16 bit negator :-**

module sixteen\_bit\_negator\_test;

reg [15:0] a;

wire [15:0] b;

sixteen\_bit\_negator gate1 (a, b);

integer i;

initial begin

for (i = 0; i < 65536; i = i + 1)

begin

{a} = i;

#20;

end

end

endmodule

**Q7) SR latch with top level entity :-**  sr\_nor\_latch

module sr\_nor\_latch(input s, r, output q, q\_);

nor(q\_, s, q);

nor(q, r, q\_);

endmodule

**Test bench for the SR latch :-**

module sr\_nor\_latch\_test;

reg s, r;

wire q, q\_;

sr\_nor\_latch sr1 (s, r, q, q\_);

initial begin

s = 1'b0; r = 1'b0; #10;

s = 1'b0; r = 1'b1; #10;

s = 1'b1; r = 1'b0; #10;

s = 1'b0; r = 1'b0; #10;

s = 1'b1; r = 1'b1; #10;

end

endmodule

**Q8) Gated SR Latch with top level entity :-** gated\_sr\_nor\_latch

module andgate(output c, input a, b);

wire a\_, b\_;

nor n1 (a\_, a, a);

nor n2 (b\_, b, b);

nor n3 (c, a\_, b\_);

endmodule

module gated\_sr\_nor\_latch(input s, r, en, output q, q\_);

wire es, er;

andgate a1 (es, s ,en);

andgate a2 (er, r, en);

nor n4 (q\_, es, q);

nor n5 (q, er, q\_);

endmodule

**Test bench for the gated sr latch:-**

module gated\_sr\_nor\_latch\_test;

reg s, r, en;

wire q, q\_;

gated\_sr\_nor\_latch gsr1 (s, r, en, q, q\_);

initial begin

en = 1'b1;

s = 1'b0; r = 1'b0; #10;

s = 1'b0; r = 1'b1; #10;

s = 1'b1; r = 1'b0; #10;

s = 1'b0; r = 1'b0; #10;

s = 1'b1; r = 1'b1; #10;

en = 1'b0;

s = 1'b0; r = 1'b0; #10;

s = 1'b0; r = 1'b1; #10;

s = 1'b1; r = 1'b0; #10;

s = 1'b0; r = 1'b0; #10;

s = 1'b1; r = 1'b1;

end

endmodule

**Q9) ALU with top level module :-** alu

module alu(input [15:0] x, y, input [3:0] ctrl, output [16:0] result);

wire [16:0] result\_1, result\_2, result\_3, result\_4, result\_5, result\_6, result\_7, result\_8, result\_9, result\_10, result\_11, result\_12, result\_13, result\_14, result\_15, result\_16;

wire [15:0] temp\_y, temp\_x;

wire [15:0] temp\_1 = 16'b1111111111111110;

sixteen\_bit\_negator n1(temp\_y, y);

sixteen\_bit\_negator n2(temp\_x, x);

sixteen\_adder a (result\_1, x, y, 1'b0);

sixteen\_adder b (result\_2, x, temp\_y, 1'b1);

sixteen\_adder c (result\_3, y, temp\_x, 1'b1);

assign result\_4 = 17'b00000000000000000;

assign result\_5 = 17'b00000000000000001;

assign result\_6 = 17'b01111111111111111;

sixteen\_adder d (result\_7, result\_4[15:0], temp\_x, 1'b1);

sixteen\_adder e (result\_8, result\_4[15:0], temp\_y, 1'b1);

assign result\_9 = temp\_x;

assign result\_10 = temp\_y;

sixteen\_adder f (result\_11, result\_4[15:0], x, 1'b1);

sixteen\_adder g (result\_12, result\_4[15:0], y, 1'b1);

sixteen\_adder h (result\_13, x, temp\_1, 1'b1);

sixteen\_adder i (result\_14, y, temp\_1, 1'b1);

sixteen\_bit\_and j (result\_15, x, y);

sixteen\_bit\_or k (result\_16, x, y);

multiplexer m0 (result, result\_1, result\_2, result\_3, result\_4, result\_5, result\_6, result\_7, result\_8, result\_9, result\_10, result\_11, result\_12, result\_13, result\_14, result\_15, result\_16, ctrl);

endmodule

module andg (output wire c, input wire a, b);

wire w1;

nand(w1, a, b);

nand(c, w1, w1);

endmodule

module org (output wire c, input wire a, b);

wire a\_, b\_;

nand(a\_, a, a);

nand(b\_, b, b);

nand(c, a\_, b\_);

endmodule

module xorg (output wire c, input wire a, b);

wire tr1, tr2;

org o1 (tr1, a, b);

nand(tr2, a, b);

andg a1(c, tr1, tr2);

endmodule

module fulladder (output wire s, cout, input wire a, b, cin);

wire w1, w2, w3;

xorg xor1(w1, a, b);

xorg xor2(s, w1, cin);

andg a1(w2, a, b);

andg a2(w3, w1, cin);

org a3(cout, w2, w3);

endmodule

module notg(output wire b, input wire a);

nand n7 (b, a, a);

endmodule

module sixteen\_adder(output [16:0] result, input [15:0] x, y, input cin);

wire c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14;

fulladder f0 (result[0], c0, x[0], y[0], cin);

fulladder f1 (result[1], c1, x[1], y[1], c0);

fulladder f2 (result[2], c2, x[2], y[2], c1);

fulladder f3 (result[3], c3, x[3], y[3], c2);

fulladder f4 (result[4], c4, x[4], y[4], c3);

fulladder f5 (result[5], c5, x[5], y[5], c4);

fulladder f6 (result[6], c6, x[6], y[6], c5);

fulladder f7 (result[7], c7, x[7], y[7], c6);

fulladder f8 (result[8], c8, x[8], y[8], c7);

fulladder f9 (result[9], c9, x[9], y[9], c8);

fulladder f10 (result[10], c10, x[10], y[10], c9);

fulladder f11 (result[11], c11, x[11], y[11], c10);

fulladder f12 (result[12], c12, x[12], y[12], c11);

fulladder f13 (result[13], c13, x[13], y[13], c12);

fulladder f14 (result[14], c14, x[14], y[14], c13);

fulladder f15 (result[15], result[16], x[15], y[15], c14);

endmodule

module sixteen\_bit\_negator(output [15:0] b, input [15:0] a);

notg g1(b[0], a[0]);

notg g2(b[1], a[1]);

notg g3(b[2], a[2]);

notg g4(b[3], a[3]);

notg g5(b[4], a[4]);

notg g6(b[5], a[5]);

notg g7(b[6], a[6]);

notg g8(b[7], a[7]);

notg g9(b[8], a[8]);

notg g10(b[9], a[9]);

notg g11(b[10], a[10]);

notg g12(b[11], a[11]);

notg g13(b[12], a[12]);

notg g14(b[13], a[13]);

notg g15(b[14], a[14]);

notg g16(b[15], a[15]);

endmodule

module sixteen\_bit\_and(output [16:0] c, input [15:0] a, b);

andg g1(c[0], b[0], a[0]);

andg g2(c[1], b[1], a[1]);

andg g3(c[2], b[2], a[2]);

andg g4(c[3], b[3], a[3]);

andg g5(c[4], b[4], a[4]);

andg g6(c[5], b[5], a[5]);

andg g7(c[6], b[6], a[6]);

andg g8(c[7], b[7], a[7]);

andg g9(c[8], b[8], a[8]);

andg g10(c[9], b[9], a[9]);

andg g11(c[10], b[10], a[10]);

andg g12(c[11], b[11], a[11]);

andg g13(c[12], b[12], a[12]);

andg g14(c[13], b[13], a[13]);

andg g15(c[14], b[14], a[14]);

andg g16(c[15], b[15], a[15]);

assign c[16]=1'b0;

endmodule

module sixteen\_bit\_or(output [16:0] c, input [15:0] a, b);

org g1(c[0], b[0], a[0]);

org g2(c[1], b[1], a[1]);

org g3(c[2], b[2], a[2]);

org g4(c[3], b[3], a[3]);

org g5(c[4], b[4], a[4]);

org g6(c[5], b[5], a[5]);

org g7(c[6], b[6], a[6]);

org g8(c[7], b[7], a[7]);

org g9(c[8], b[8], a[8]);

org g10(c[9], b[9], a[9]);

org g11(c[10], b[10], a[10]);

org g12(c[11], b[11], a[11]);

org g13(c[12], b[12], a[12]);

org g14(c[13], b[13], a[13]);

org g15(c[14], b[14], a[14]);

org g16(c[15], b[15], a[15]);

assign c[16]=1'b0;

endmodule

module four\_bit\_and(output [3:0] c, input [3:0] a, b);

andg g1(c[0], b[0], a[0]);

andg g2(c[1], b[1], a[1]);

andg g3(c[2], b[2], a[2]);

andg g4(c[3], b[3], a[3]);

endmodule

module mux(output wire ou, input wire in1, in2, ctrl);

wire w1, w2, ctrl\_;

notg n1 (ctrl\_, ctrl);

andg a1 (w1, in1, ctrl\_);

andg a2 (w2, in2, ctrl);

org o1 (ou, w1, w2);

endmodule

module sixteen\_mux(output o, input in1, in2, in3, in4, in5, in6, in7, in8, in9, in10, in11, in12, in13, in14, in15, in16, input [3:0] ctrl);

wire w11, w12, w13, w14, w15, w16, w17, w18;

wire w21, w22, w23, w24;

wire w31, w32;

mux m1(w11, in1, in2, ctrl[0]);

mux m2(w12, in3, in4, ctrl[0]);

mux m3(w13, in5, in6, ctrl[0]);

mux m4(w14, in7, in8, ctrl[0]);

mux m5(w15, in9, in10, ctrl[0]);

mux m6(w16, in11, in12, ctrl[0]);

mux m7(w17, in13, in14, ctrl[0]);

mux m8(w18, in15, in16, ctrl[0]);

mux m9(w21, w11, w12, ctrl[1]);

mux m10(w22, w13, w14, ctrl[1]);

mux m11(w23, w15, w16, ctrl[1]);

mux m12(w24, w17, w18, ctrl[1]);

mux m13(w31, w21, w22, ctrl[2]);

mux m14(w32, w23, w24, ctrl[2]);

mux m15(o, w31, w32, ctrl[3]);

endmodule

module multiplexer (output [16:0] result, input [16:0] result\_1, result\_2, result\_3, result\_4, result\_5, result\_6, result\_7, result\_8, result\_9, result\_10, result\_11, result\_12, result\_13, result\_14, result\_15, result\_16, input[3:0] ctrl);

sixteen\_mux m1(result[0], result\_1[0], result\_2[0], result\_3[0], result\_4[0], result\_5[0], result\_6[0], result\_7[0], result\_8[0],

result\_9[0], result\_10[0], result\_11[0], result\_12[0], result\_13[0], result\_14[0], result\_15[0], result\_16[0], ctrl);

sixteen\_mux m2(result[1], result\_1[1], result\_2[1], result\_3[1], result\_4[1], result\_5[1], result\_6[1], result\_7[1], result\_8[1], result\_9[1], result\_10[1], result\_11[1], result\_12[1], result\_13[1], result\_14[1], result\_15[1], result\_16[1], ctrl);

sixteen\_mux m3(result[2], result\_1[2], result\_2[2], result\_3[2], result\_4[2], result\_5[2], result\_6[2], result\_7[2], result\_8[2], result\_9[2], result\_10[2], result\_11[2], result\_12[2], result\_13[2], result\_14[2], result\_15[2], result\_16[2], ctrl);

sixteen\_mux m4(result[3], result\_1[3], result\_2[3], result\_3[3], result\_4[3], result\_5[3], result\_6[3], result\_7[3], result\_8[3], result\_9[3], result\_10[3], result\_11[3], result\_12[3], result\_13[3], result\_14[3], result\_15[3], result\_16[3], ctrl);

sixteen\_mux m5(result[4], result\_1[4], result\_2[4], result\_3[4], result\_4[4], result\_5[4], result\_6[4], result\_7[4], result\_8[4], result\_9[4], result\_10[4], result\_11[4], result\_12[4], result\_13[4], result\_14[4], result\_15[4], result\_16[4], ctrl);

sixteen\_mux m6(result[5], result\_1[5], result\_2[5], result\_3[5], result\_4[5], result\_5[5], result\_6[5], result\_7[5], result\_8[5], result\_9[5], result\_10[5], result\_11[5], result\_12[5], result\_13[5], result\_14[5], result\_15[5], result\_16[5], ctrl);

sixteen\_mux m7(result[6], result\_1[6], result\_2[6], result\_3[6], result\_4[6], result\_5[6], result\_6[6], result\_7[6], result\_8[6], result\_9[6], result\_10[6], result\_11[6], result\_12[6], result\_13[6], result\_14[6], result\_15[6], result\_16[6], ctrl);

sixteen\_mux m8(result[7], result\_1[7], result\_2[7], result\_3[7], result\_4[7], result\_5[7], result\_6[7], result\_7[7], result\_8[7], result\_9[7], result\_10[7], result\_11[7], result\_12[7], result\_13[7], result\_14[7], result\_15[7], result\_16[7], ctrl);

sixteen\_mux m9(result[8], result\_1[8], result\_2[8], result\_3[8], result\_4[8], result\_5[8], result\_6[8], result\_7[8], result\_8[8], result\_9[8], result\_10[8], result\_11[8], result\_12[8], result\_13[8], result\_14[8], result\_15[8], result\_16[8], ctrl);

sixteen\_mux m10(result[9], result\_1[9], result\_2[9], result\_3[9], result\_4[9], result\_5[9], result\_6[9], result\_7[9], result\_8[9], result\_9[9], result\_10[9], result\_11[9], result\_12[9], result\_13[9], result\_14[9], result\_15[9], result\_16[9], ctrl);

sixteen\_mux m11(result[10], result\_1[10], result\_2[10], result\_3[10], result\_4[10], result\_5[10], result\_6[10], result\_7[10], result\_8[10], result\_9[10], result\_10[10], result\_11[10], result\_12[10], result\_13[10], result\_14[10], result\_15[10], result\_16[10], ctrl);

sixteen\_mux m12(result[11], result\_1[11], result\_2[11], result\_3[11], result\_4[11], result\_5[11], result\_6[11], result\_7[11], result\_8[11], result\_9[11], result\_10[11], result\_11[11], result\_12[11], result\_13[11], result\_14[11], result\_15[11], result\_16[11], ctrl);

sixteen\_mux m13(result[12], result\_1[12], result\_2[12], result\_3[12], result\_4[12], result\_5[12], result\_6[12], result\_7[12], result\_8[12], result\_9[12], result\_10[12], result\_11[12], result\_12[12],

result\_13[12], result\_14[12], result\_15[12], result\_16[12], ctrl);

sixteen\_mux m14(result[13], result\_1[13], result\_2[13], result\_3[13], result\_4[13], result\_5[13], result\_6[13], result\_7[13], result\_8[13], result\_9[13], result\_10[13], result\_11[13], result\_12[13], result\_13[13], result\_14[13], result\_15[13], result\_16[13], ctrl);

sixteen\_mux m15(result[14], result\_1[14], result\_2[14], result\_3[14], result\_4[14], result\_5[14], result\_6[14], result\_7[14], result\_8[14], result\_9[14], result\_10[14], result\_11[14], result\_12[14], result\_13[14], result\_14[14], result\_15[14], result\_16[14], ctrl);

sixteen\_mux m16(result[15], result\_1[15], result\_2[15], result\_3[15], result\_4[15], result\_5[15], result\_6[15], result\_7[15], result\_8[15], result\_9[15], result\_10[15], result\_11[15], result\_12[15], result\_13[15], result\_14[15], result\_15[15], result\_16[15], ctrl);

sixteen\_mux m17(result[16], result\_1[16], result\_2[16], result\_3[16], result\_4[16], result\_5[16], result\_6[16], result\_7[16], result\_8[16], result\_9[16], result\_10[16], result\_11[16], result\_12[16], result\_13[16], result\_14[16], result\_16[16], result\_16[16], ctrl);

endmodule

**Testbench for ALU :-**

module alu\_tb;

reg [15:0] X, Y;

wire[16:0] result;

reg[3:0] ctrl;

alu a1 (X, Y, ctrl, result);

integer i;

initial

begin

assign X = $random;

assign Y = $random;

#10;

for(i = 0; i < 16; i = i + 1)

begin

ctrl = i;

#10;

end

end

endmodule