MD TAHSEEN RAZA

B210562CS

Q1

4-bit incrementer verilog code

{

module andd(input a,b , output s);

wire w1;

nand n1(w1,a,b);

nand n2(s,w1,w1);

endmodule

module xorr(input a,b , output s);

wire w1,w2,w3,w4;

nott m1(a,w1);

nott m2(b,w2);

andd a1(a,w2,w3);

andd a2(w1,b,w4);

orr o1(w3,w4,s);

endmodule

module halfadder(input a,b , output s,c);

xorr x1(a,b , s);

andd a3(a,b , c);

endmodule

module inc4bit(input [3:0]a, output c,output [3:0]s);

wire w1,w2,w3,w4,w5;

halfadder h1(a[0],1 , s[0],w1);

halfadder h2(a[1],w1, s[1],w2);

halfadder h3(a[2],w2, s[2],w3);

halfadder h4(a[3],w3, s[3],c);

endmodule

}

4-bit incrementer testbench

{

module inc4bit\_tb;

reg [3:0]a;

wire [3:0]s;

wire c;

inc4bit uut(a,c,s);

integer i;

initial begin

for(i=0;i<=15;i=i+1)

begin

{a[3],a[2],a[1],a[0]} = i;

#20;

end

end

endmodule

}

Q2

4-bit decrementer verilog code

{

module andd(input a,b , output s);

wire w1;

nand n1(w1,a,b);

nand n2(s,w1,w1);

endmodule

module orr(input a,b , output s);

wire w1,w2,w3;

nand n3(w1,a,a);

nand n4(w2,b,b);

nand n5(s,w1,w2);

endmodule

module nott(input a, output s);

nand n6(s,a,a);

endmodule

module xorr(input a,b , output s);

wire w1,w2,w3,w4;

nott m1(a,w1);

nott m2(b,w2);

andd a1(a,w2,w3);

andd a2(w1,b,w4);

orr o1(w3,w4,s);

endmodule

module halfsubtractor(input a,b , output s,c);

wire w1;

xorr x1(a,b , s);

nott n1(a,w1);

andd a3(w1,b, c);

endmodule

module dec4bit(input [3:0]a , output b,output [3:0]d);

wire w1,w2,w3,w4,w5,w6;

halfsubtractor h1(a[0],1 ,d[0],w1);

halfsubtractor h2(a[1],w1,d[1],w2);

halfsubtractor h3(a[2],w2,d[2],w3);

halfsubtractor h4(a[3],w3,d[3],b );

endmodule

}

4-bit decrementer testbench

{

module dec4bit\_tb;

reg [3:0]a;

wire [3:0]d;

wire b;

dec4bit uut(a,b,d);

integer i;

initial begin

for(i=0;i<=15;i=i+1)

begin

{a[3],a[2],a[1],a[0]} = i;

#20;

end

end

endmodule

}

Q3

16-bit incrementer verilog code

{

module andd(input a,b , output s);

wire w1;

nand n1(w1,a,b);

nand n2(s,w1,w1);

endmodule

module xorr(input a,b , output s);

wire w1,w2,w3,w4;

nott m1(a,w1);

nott m2(b,w2);

andd a1(a,w2,w3);

andd a2(w1,b,w4);

orr o1(w3,w4,s);

endmodule

module halfadder(input a,b , output s,c);

xorr x1(a,b , s);

andd a3(a,b , c);

endmodule

module inc16bit(input [15:0]a, output c,output [15:0]s);

wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15;

halfadder h1(a[0],1 , s[0],w1);

halfadder h2(a[1],w1, s[1],w2);

halfadder h3(a[2],w2, s[2],w3);

halfadder h4(a[3],w3, s[3],w4);

halfadder h5(a[4],w4, s[4],w5);

halfadder h6(a[5],w5, s[5],w6);

halfadder h7(a[6],w6, s[6],w7);

halfadder h8(a[7],w7, s[7],w8);

halfadder h9(a[8],w8, s[8],w9);

halfadder h10(a[9],w9, s[9],w10);

halfadder h11(a[10],w10, s[10],w11);

halfadder h12(a[11],w11, s[11],w12);

halfadder h13(a[12],w12, s[12],w13);

halfadder h14(a[13],w13, s[13],w14);

halfadder h15(a[14],w14, s[14],w15);

halfadder h16(a[15],w15, s[15],c);

endmodule

}

16-bit incrementer testbench

{

module inc16bit\_tb;

reg [15:0]a;

wire [15:0]s;

wire c;

inc16bit uut(a,c,s);

integer i;

initial begin

for(i=0;i<=65535;i=i+1)

begin

{a[15],a[14],a[13],a[12],a[11],a[10],a[9],a[8],

a[7],a[6],a[5],a[4],a[3],a[2],a[1],a[0]} = i;

#20;

end

end

endmodule

}

Q4

16-bit decrementer verilog code

{

module andd(input a,b , output s);

wire w1;

nand n1(w1,a,b);

nand n2(s,w1,w1);

endmodule

module orr(input a,b , output s);

wire w1,w2,w3;

nand n3(w1,a,a);

nand n4(w2,b,b);

nand n5(s,w1,w2);

endmodule

module nott(input a, output s);

nand n6(s,a,a);

endmodule

module xorr(input a,b , output s);

wire w1,w2,w3,w4;

nott m1(a,w1);

nott m2(b,w2);

andd a1(a,w2,w3);

andd a2(w1,b,w4);

orr o1(w3,w4,s);

endmodule

module halfsubtractor(input a,b , output s,c);

wire w1;

xorr x1(a,b , s);

nott n1(a,w1);

andd a3(w1,b, c);

endmodule

module dec16bit(input [15:0]a , output b,output [15:0]d);

wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16;

halfsubtractor h1(a[0],1 ,d[0],w1);

halfsubtractor h2(a[1],w1,d[1],w2);

halfsubtractor h3(a[2],w2,d[2],w3);

halfsubtractor h4(a[3],w3,d[3],w4);

halfsubtractor h5(a[4],w4,d[4],w5);

halfsubtractor h6(a[5],w5,d[5],w6);

halfsubtractor h7(a[6],w6,d[6],w7);

halfsubtractor h8(a[7],w7,d[7],w8);

halfsubtractor h9(a[8],w8,d[8],w9);

halfsubtractor h10(a[9],w9,d[9],w10);

halfsubtractor h11(a[10],w10,d[10],w11);

halfsubtractor h12(a[11],w11,d[11],w12);

halfsubtractor h13(a[12],w12,d[12],w13);

halfsubtractor h14(a[13],w13,d[13],w14);

halfsubtractor h15(a[14],w14,d[14],w15);

halfsubtractor h16(a[15],w15,d[15],b);

endmodule

}

16-bit decrementer testbench

{

module dec16bit\_tb;

reg [15:0]a;

wire [15:0]d;

wire b;

dec16bit uut(a,b,d);

integer i;

initial begin

for(i=0;i<=65535;i=i+1)

begin

{ a[15],a[14],a[13],a[12], a[11],a[10],a[9],a[8],

a[7],a[6],a[5],a[4],a[3],a[2],a[1],a[0] } = i;

#20;

end

end

endmodule

}

Q5

4-bit negator verilog code

{

module neg4bit(input [3:0]a , output [3:0]b);

nand n1(b[0],a[0],a[0]);

nand n2(b[1],a[1],a[1]);

nand n3(b[2],a[2],a[2]);

nand n4(b[3],a[3],a[3]);

endmodule

}

4-bit negator testbench

{

module neg4bit\_tb;

reg [3:0]a;

wire [3:0]b;

neg4bit uut(a,b);

integer i;

initial begin

for(i=0;i<=15;i=i+1)

begin

{a[3],a[2],a[1],a[0]} = i;

#20;

end

end

endmodule

}

Q6

16-bit negator verilog code

{

module neg16bit(input [15:0]a , output [15:0]b);

nand n1(b[0],a[0],a[0]);

nand n2(b[1],a[1],a[1]);

nand n3(b[2],a[2],a[2]);

nand n4(b[3],a[3],a[3]);

nand n5(b[4],a[4],a[4]);

nand n6(b[5],a[5],a[5]);

nand n7(b[6],a[6],a[6]);

nand n8(b[7],a[7],a[7]);

nand n9(b[8],a[8],a[8]);

nand n10(b[9],a[9],a[9]);

nand n11(b[10],a[10],a[10]);

nand n12(b[11],a[11],a[11]);

nand n13(b[12],a[12],a[12]);

nand n14(b[13],a[13],a[13]);

nand n15(b[14],a[14],a[14]);

nand n16(b[15],a[15],a[15]);

endmodule

}

16-bit negator testbench

{

module neg16bit\_tb;

reg [15:0]a;

wire [15:0]b;

neg16bit uut(a,b);

integer i;

initial begin

for(i=0;i<=65355;i=i+1)

begin

{ a[15],a[14],a[13],a[12],a[11],a[10],a[9],a[8],

a[7],a[6],a[5],a[4],a[3],a[2],a[1],a[0] } = i;

#20;

end

end

endmodule

}

Q7

S-R latch verilog code

{

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

wire w1;

nand g1(w1,a,b);

nand g2(c,w1,w1);

endmodule

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

wire w1,w2;

nand g1(w1,a,a);

nand g2(w2,b,b);

nand g3(c,w1,w2);

endmodule

module nott(output b,input a);

nand g1(b,a,a);

endmodule

module sr(input s,r,output q,qn);

wire qt,qnt,w1,w2,w3,w4,w5,w8;

nand g1(qt,s,qnt);

nand g2(qnt,r,qt);

andd g5(w1,qt,qnt);

andd t6(w8,w1,1'bx);

nott g3(w3,qnt);

andd g6(w2,qt,w3);

orr g7(q,w8,w2);

nott g4(w4,qt);

andd g9(w5,w4,qnt);

orr g10(qn,w8,w5);

endmodule

}

S-R latch testbench

{

module sr\_tb;

reg s,r;

wire q,qn;

sr uut(s,r,q,qn);

integer i;

initial begin

$monitor("r=%b,s=%b,q=%b,qn=%b",r,s,q,qn);

s=1'b1;

r=1'b1;

#10;

for(i=0;i<10;i=i+1)

begin

s=$random;

r=$random;

#10;

end

s=1'b0;

r=1'b0;

#10;

end

endmodule

}

Q8

Gated S-R latch verilog code

{

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

wire w1;

nand g1(w1,a,b);

nand g2(c,w1,w1);

endmodule

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

wire w1,w2;

nand g1(w1,a,a);

nand g2(w2,b,b);

nand g3(c,w1,w2);

endmodule

module nott(output b,input a);

nand g1(b,a,a);

endmodule

module gsr(input s,r,c,output q,qn);

wire qt,qnt,w1,w2,w3,w4,w5,w6,w7;

nand t1(w6,s,c);

nand t2(w7,r,c);

nand g1(qt,w6,qnt);

nand g2(qnt,w7,qt);

andd g5(w1,qt,qnt);

andd t6(w8,w1,1'bx);

nott g3(w3,qnt);

andd g6(w2,qt,w3);

orr g7(q,w8,w2);

nott g4(w4,qt);

andd g9(w5,w4,qnt);

orr g10(qn,w8,w5);

endmodule

}

Gated S-R latch testbench

{

module gsr\_tb;

reg s,r,c;

wire q,qn;

gsr uut(s,r,c,q,qn);

integer i;

initial begin

$monitor("c=%b,r=%b,s=%b,q=%b,qn=%b",c,r,s,q,qn);

s=1'b1;

r=1'b1;

c=1'b1;

#10;

for(i=0;i<10;i=i+1)

begin

s=$random;

r=$random;

c=$random;

#10;

end

s=1'b0;

r=1'b0;

c=1'b1;

#10;

end

endmodule

}

Q9

ALU verilog code

{

module andd(input a,b , output s);

wire w1;

nand n1(w1,a,b);

nand n2(s,w1,w1);

endmodule

module orr(input a,b , output s);

wire w1,w2,w3;

nand n3(w1,a,a);

nand n4(w2,b,b);

nand n5(s,w1,w2);

endmodule

module nott(input a, output s);

nand n6(s,a,a);

endmodule

module xorr(input a,b , output s);

wire w1,w2,w3,w4;

nott m1(a,w1);

nott m2(b,w2);

andd a1(a,w2,w3);

andd a2(w1,b,w4);

orr o1(w3,w4,s);

endmodule

module halfsubtractor(input a,b , output s,c);

wire w1;

xorr x1(a,b , s);

nott n1(a,w1);

andd a3(w1,b, c);

endmodule

module fadd(input a,b,co , output s,c);

wire w1,w2,w3,w4,w5;

xorr x1(a,b,w1);

xorr x2(w1,co,s);

andd a3(a,b,w2);

andd a4(b,co,w3);

andd a5(a,co,w4);

orr o6(w3,w4,w5);

orr o7(w2,w5,c);

endmodule

module fsub(input a,b,co,output s,c);

wire w1,w2,w3,w4,w5,w6,w7,w8;

nott nm1(a,w6);

xorr x1(a,b,w1);

xorr x2(w1,co,s);

andd a3(w6,b,w2);

andd a4(b,co,w3);

andd a5(w6,co,w4);

orr o6(w3,w4,w5);

orr o7(w2,w5,c);

endmodule

module neg16bit(input [15:0]a , output [15:0]b);

nand n1(b[0],a[0],a[0]);

nand n2(b[1],a[1],a[1]);

nand n3(b[2],a[2],a[2]);

nand n4(b[3],a[3],a[3]);

nand n5(b[4],a[4],a[4]);

nand n6(b[5],a[5],a[5]);

nand n7(b[6],a[6],a[6]);

nand n8(b[7],a[7],a[7]);

nand n9(b[8],a[8],a[8]);

nand n10(b[9],a[9],a[9]);

nand n11(b[10],a[10],a[10]);

nand n12(b[11],a[11],a[11]);

nand n13(b[12],a[12],a[12]);

nand n14(b[13],a[13],a[13]);

nand n15(b[14],a[14],a[14]);

nand n16(b[15],a[15],a[15]);

endmodule

module halfadder(input a,b , output s,c);

xorr x1(a,b , s);

andd a3(a,b , c);

endmodule

module inc16bit(input [15:0]a, output c,output [15:0]s);

wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15;

halfadder h1(a[0],1 , s[0],w1);

halfadder h2(a[1],w1, s[1],w2);

halfadder h3(a[2],w2, s[2],w3);

halfadder h4(a[3],w3, s[3],w4);

halfadder h5(a[4],w4, s[4],w5);

halfadder h6(a[5],w5, s[5],w6);

halfadder h7(a[6],w6, s[6],w7);

halfadder h8(a[7],w7, s[7],w8);

halfadder h9(a[8],w8, s[8],w9);

halfadder h10(a[9],w9, s[9],w10);

halfadder h11(a[10],w10, s[10],w11);

halfadder h12(a[11],w11, s[11],w12);

halfadder h13(a[12],w12, s[12],w13);

halfadder h14(a[13],w13, s[13],w14);

halfadder h15(a[14],w14, s[14],w15);

halfadder h16(a[15],w15, s[15],c);

endmodule

module add16(input [15:0]x,y,output [16:0]a);

wire [14:0]w;

fadd f1(x[0],0,y[0] , a[0],w[0]);

fadd f2(x[1],w[0],y[1] , a[1],w[1]);

fadd f3(x[2],w[1],y[2] , a[2],w[2]);

fadd f4(x[3],w[2],y[3] , a[3],w[3]);

fadd f5(x[4],w[3],y[4] , a[4],w[4]);

fadd f6(x[5],w[4],y[5] , a[5],w[5]);

fadd f7(x[6],w[5],y[6] , a[6],w[6]);

fadd f8(x[7],w[6],y[7] , a[7],w[7]);

fadd f9(x[8],w[7],y[8] , a[8],w[8]);

fadd f10(x[9],w[8],y[9] , a[9],w[9]);

fadd f11(x[10],w[9],y[10] , a[10],w[10]);

fadd f12(x[11],w[10],y[11] , a[11],w[11]);

fadd f13(x[12],w[11],y[12] , a[12],w[12]);

fadd f14(x[13],w[12],y[13] , a[13],w[13]);

fadd f15(x[14],w[13],y[14] , a[14],w[14]);

fadd f16(x[15],w[14],y[15] , a[15],a[16]);

endmodule

module dif16(input [15:0]x,y,output [16:0]a);

wire [14:0]w;

fsub f1(x[0],0,y[0] , a[0],w[0]);

fsub f2(x[1],w[0],y[1] , a[1],w[1]);

fsub f3(x[2],w[1],y[2] , a[2],w[2]);

fsub f4(x[3],w[2],y[3] , a[3],w[3]);

fsub f5(x[4],w[3],y[4] , a[4],w[4]);

fsub f6(x[5],w[4],y[5] , a[5],w[5]);

fsub f7(x[6],w[5],y[6] , a[6],w[6]);

fsub f8(x[7],w[6],y[7] , a[7],w[7]);

fsub f9(x[8],w[7],y[8] , a[8],w[8]);

fsub f10(x[9],w[8],y[9] , a[9],w[9]);

fsub f11(x[10],w[9],y[10] , a[10],w[10]);

fsub f12(x[11],w[10],y[11] , a[11],w[11]);

fsub f13(x[12],w[11],y[12] , a[12],w[12]);

fsub f14(x[13],w[12],y[13] , a[13],w[13]);

fsub f15(x[14],w[13],y[14] , a[14],w[14]);

fsub f16(x[15],w[14],y[15] , a[15],a[16]);

endmodule

module alu(x,y,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p);

input [15:0]x,y;

output [16:0]a,b,c,f,k,l,m,n;

output [15:0]g,h,i,j,o,p;

output d,e;

wire [15:0]y1,x1,y2,x2,x3,y3,y4;

wire q1,q2,q3,q4,r1,r2,r3,r4,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15;

wire lp1,lp2,lp3,lo1,lo2,lk1,lk2,lk3,lk4,kl1,kl2,kl3,kl4,kl5,kl6,kl7,kl8,kl9,kl10,kl11,kl12,kl13,kl14,kl15,lk5,lk6,lk7,lk8,lk9,lk10,lk11,lk12,lk13,lk14,lk15;

add16 a1(x,y,a);

dif16 d1(x,y,b);

dif16 d2(y,x,c);

orr o1(0,0,d);

orr k1(0,1,e);

orr l1(0,1,f[0]);

orr l2(0,1,f[1]);

orr l3(0,1,f[2]);

orr l4(0,1,f[3]);

orr l5(0,1,f[4]);

orr l6(0,1,f[5]);

orr l7(0,1,f[6]);

orr l8(0,1,f[7]);

orr l9(0,1,f[8]);

orr l10(0,1,f[9]);

orr l11(0,1,f[10]);

orr l12(0,1,f[11]);

orr l13(0,1,f[12]);

orr l14(0,1,f[13]);

orr l15(0,1,f[14]);

orr l16(0,1,f[15]);

orr l17(0,1,f[16]);

neg16bit n3(x,x3);

inc16bit s3(x3,q3,g);

neg16bit n4(y,y4);

inc16bit s4(y4,q4,h);

neg16bit n5(x,i);

neg16bit n6(y,j);

halfadder h1(x[0],1 , k[0],w1);

halfadder h2(x[1],w1, k[1],w2);

halfadder h3(x[2],w2, k[2],w3);

halfadder h4(x[3],w3, k[3],w4);

halfadder h5(x[4],w4, k[4],w5);

halfadder h6(x[5],w5, k[5],w6);

halfadder h7(x[6],w6, k[6],w7);

halfadder h8(x[7],w7, k[7],w8);

halfadder h9(x[8],w8, k[8],w9);

halfadder h10(x[9],w9, k[9],w10);

halfadder h11(x[10],w10, k[10],w11);

halfadder h12(x[11],w11, k[11],w12);

halfadder h13(x[12],w12, k[12],w13);

halfadder h14(x[13],w13, k[13],w14);

halfadder h15(x[14],w14, k[14],w15);

halfadder h16(x[15],w15, k[15],k[16]);

halfadder z1(y[0],1 , l[0],r1);

halfadder z2(y[1],r1, l[1],r2);

halfadder z3(y[2],r2, l[2],r3);

halfadder z4(y[3],r3, l[3],r4);

halfadder z5(y[4],r4, l[4],r5);

halfadder z6(y[5],r5, l[5],r6);

halfadder z7(y[6],r6, l[6],r7);

halfadder z8(y[7],r7, l[7],r8);

halfadder z9(y[8],r8, l[8],r9);

halfadder z10(y[9],r9, l[9],r10);

halfadder z11(y[10],r10, l[10],r11);

halfadder z12(y[11],r11, l[11],r12);

halfadder z13(y[12],r12, l[12],r13);

halfadder z14(y[13],r13, l[13],r14);

halfadder z15(y[14],r14, l[14],r15);

halfadder z16(y[15],r15, l[15],l[16]);

halfsubtractor hs1(x[0],1 ,m[0],lk1);

halfsubtractor hs2(x[1],lk1,m[1],lk2);

halfsubtractor hs3(x[2],lk2,m[2],lk3);

halfsubtractor hs4(x[3],lk3,m[3],lk4);

halfsubtractor hs5(x[4],lk4,m[4],lk5);

halfsubtractor hs6(x[5],lk5,m[5],lk6);

halfsubtractor hs7(x[6],lk6,m[6],lk7);

halfsubtractor hs8(x[7],lk7,m[7],lk8);

halfsubtractor hs9(x[8],lk8,m[8],lk9);

halfsubtractor hs10(x[9],lk9,m[9],lk10);

halfsubtractor hs11(x[10],lk10,m[10],lk11);

halfsubtractor hs12(x[11],lk11,m[11],lk12);

halfsubtractor hs13(x[12],lk12,m[12],lk13);

halfsubtractor hs14(x[13],lk13,m[13],lk14);

halfsubtractor hs15(x[14],lk14,m[14],lk15);

halfsubtractor hs16(x[15],lk15,m[15],m[16]);

halfsubtractor hm1(y[0],1 ,n[0],kl1);

halfsubtractor hm2(y[1],kl1,n[1],kl2);

halfsubtractor hm3(y[2],kl2,n[2],kl3);

halfsubtractor hm4(y[3],kl3,n[3],kl4);

halfsubtractor hm5(y[4],kl4,n[4],kl5);

halfsubtractor hm6(y[5],kl5,n[5],kl6);

halfsubtractor hm7(y[6],kl6,n[6],kl7);

halfsubtractor hm8(y[7],kl7,n[7],kl8);

halfsubtractor hm9(y[8],kl8,n[8],kl9);

halfsubtractor hm10(y[9],kl9,n[9],kl10);

halfsubtractor hm11(y[10],kl10,n[10],kl11);

halfsubtractor hm12(y[11],kl11,n[11],kl12);

halfsubtractor hm13(y[12],kl12,n[12],kl13);

halfsubtractor hm14(y[13],kl13,n[13],kl14);

halfsubtractor hm15(y[14],kl14,n[14],kl15);

halfsubtractor hm16(y[15],kl15,n[15],n[16]);

andd ak1(x[0],y[0],o[0]);

andd ak2(x[1],y[1],o[1]);

andd ak3(x[2],y[2],o[2]);

andd ak4(x[3],y[3],o[3]);

andd ak5(x[4],y[4],o[4]);

andd ak6(x[5],y[5],o[5]);

andd ak7(x[6],y[6],o[6]);

andd ak8(x[7],y[7],o[7]);

andd ak9(x[8],y[8],o[8]);

andd ak10(x[9],y[9],o[9]);

andd ak11(x[10],y[10],o[10]);

andd ak12(x[11],y[11],o[11]);

andd ak13(x[12],y[12],o[12]);

andd ak14(x[13],y[13],o[13]);

andd ak15(x[14],y[14],o[14]);

andd ak16(x[15],y[15],o[15]);

orr pk1(x[0],y[0],p[0]);

orr pk2(x[1],y[1],p[1]);

orr pk3(x[2],y[2],p[2]);

orr pk4(x[3],y[3],p[3]);

orr pk5(x[4],y[4],p[4]);

orr pk6(x[5],y[5],p[5]);

orr pk7(x[6],y[6],p[6]);

orr pk8(x[7],y[7],p[7]);

orr pk9(x[8],y[8],p[8]);

orr pk10(x[9],y[9],p[9]);

orr pk11(x[10],y[10],p[10]);

orr pk12(x[11],y[11],p[11]);

orr pk13(x[12],y[12],p[12]);

orr pk14(x[13],y[13],p[13]);

orr pk15(x[14],y[14],p[14]);

orr pk16(x[15],y[15],p[15]);

endmodule

}

ALU testbench

{

module alu\_tb;

reg [15:0]x,y;

wire [16:0]a,b,c,f,k,l,m,n;

wire [15:0]g,h,i,j,o,p;

wire d,e;

alu uut(x,y,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p);

integer op;

initial begin

for(op=0;op<=1000;op=op+1)

begin

x=$random;

y=$random;

#20;

end

end

endmodule

}