MD TAHSEEN RAZA

B210562CS

1. 4 BIT BINARY ADDER

module mus(A,B,cin,S,C);

input [3:0] A,B;

input cin;

output [3:0]S;

output C;

wire d,e,f,g,p,q,r;

xor x1(d,B[0],cin);

FD A1(A[0],d,cin,S[0],p);

xor x2(e,B[1],cin);

FD A2(A[1],e,p,S[1],q);

xor x3(f,B[2],cin);

FD A3(A[2],f,q,S[2],r);

xor x4(g,B[3],cin);

FD A4(A[3],g,r,S[3],C);

endmodule

module FD(input a,b,cin, output S,C);

wire w1,w2,w3;

xor g1(w1,a,b);

xor g2(S,w1,cin);

and g3(w2,w1,cin);

and g4(w3,a,b);

or g5(C,w2,w3);

endmodule

module mus\_yz;

reg [3:0]X,Y;

wire [3:0]S;

wire C;

mus fb(X,Y,1'b0,S,C);

integer i;

initial begin

for(i=0;i<256;i=i+1)

begin

X=i;Y=i;

#10;

end

end

endmodule

//4 BIT BINARY SUBTRACTOR

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

wire f1,f2,f3;

xor(f3,a,b);

xor(s,f3,cin);

and(f1,a,b);

and(f2,cin,f3);

or(cout,f1,f2);

Endmodule

module ank(output [3:0] D,output cout,input [3:0] A,B);

wire c1,c2,c3;

fd d1(D[0],c1,A[0],~B[0],1'b1),

d2(D[1],c2,A[1],~B[1],c1),

d3(D[2],c3,A[2],~B[2],c2),

d4(D[3],cout,A[3],~B[3],c3);

endmodule

module ank\_mn;

reg [3:0] a,b;

wire Cout;

wire [3:0] d;

integer i;

ank Instance (d,Cout,a,b,1'b1);

initial begin

a <= 0;

b <= 0;

$monitor ("a=0x%0h b=0x%0h Cout=0x%0h sum=0x%0h", a, b,Cout,d);

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

#10 a <= $random;

b <= $random;

end

end

endmodule

2. 8:1 MULTIPLEXER

module Quad\_2to1\_ Multiplexer (output reg [3:0]out,input [3:0]in1,in2,input s,E);

always@(s,E,in1,in2)

begin

if(E)

begin

out=4'b0000;

case(s)

1'b0:out=in1;

1'b1:out=in2;

default:out=4'b0000;

endcase

end

else

out=4'b0000;

end

endmodule

Testbench:

module gk\_Quad\_2to1\_ Multiplexer ();

wire [3:0]out;

reg [3:0]in1,in2;

reg s,E;

Quad\_2to1\_ Multiplexer kg(out,in1,in2,s,E);

initial begin

repeat(25)

begin

in1=$random();

in2=$random();

s=$random();

E=$random();

#30;

end

end

endmodule

5. 4 BIT RIPPLE CARRY ADDER

module four bit\_ripple\_carry\_adder(m,n, S,C);

input [3:0] m,n;

output [3:0] S;

output C;

wire p1,p2,p3;

Full\_adder d1(m[0],n[0],1'b0,S[0],p1);

full\_adder d2(m[1],n[1],p1,S[1],p2);

Full\_adder d3(m[2],n[2],p2,S[2],p3);

full\_adder fd4(m[3],n[3],p3,S[3],C);

enmodule

module full\_adder(input r,q,cin, output S,C);

wire p1,p2,p3;

xor t1(p1,r,q);

xor t2(S,p1,cin);

and t3(p2,p1,cin);

and t4(p3,r,q);

or t5(C,p2,p3);

endmodule

module four bit\_ripple\_carry\_adder\_tb;

reg [3:0]m,n;

wire [3:0]S;

wire C;

four bit\_ripple\_carry\_adder arya(m,n,S,C);

integer i;

initial begin

for(i=0;i<256;i=i+1)

begin

m=i;n=i;

#10;

end

end

Endmodule