MD TASHEEN RAZA

B210562CS

//Q2:

//8 bit ripple carry adder using full adders

module Assg21(input [7:0]a,b,output[7:0]s,output cout);

wire c1,c2,c3,c4,c5,c6,c7;

fulladder FA00(a[0],b[0],1'b0,s[0],c1);

fulladder FA01(a[1],b[1],c1, s[1],c2);

fulladder FA02(a[2],b[2],c2, s[2],c3);

fulladder FA03(a[3],b[3],c3, s[3],c4);

fulladder FA04(a[4],b[4],c4, s[4],c5);

fulladder FA05(a[5],b[5],c5, s[5],c6);

fulladder FA06(a[6],b[6],c6, s[6],c7);

fulladder FA07(a[7],b[7],c7, s[7],cout);

endmodule

module fulladder(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

wire c1,c2,s1;

half g1(a,b,c1,s1);

half g2(s1,cin,c2,s);

or g3(cout,c1,c2);

endmodule

module half(a,b,c,s);

input a,b;

output c,s;

xor g0(s,a,b);

and g1(c,a,b);

endmodule

module test;

reg [7:0]a,b;

wire [7:0]s;

wire cout;

Assg21 uut(.a(a),.b(b),.s(s),.cout(cout));

integer i=16'b0;

initial begin

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

{a,b}=i;

#20;

end

end

endmodule

//Q3:

module Assg21 (

input [3:0] a, // 4-bit BCD input A

input [3:0] b, // 4-bit BCD input B

output [3:0] sum, // 4-bit BCD output Sum

output Cout // Carry-Out

);

wire [3:0] s;//4 bit sum for first full adder

wire c1s,c2s,c3s,couts;// wires for first adder

wire x1,x2;//intermediates to find cout

wire c1ss,c2ss,c3ss,coutss;//wires for second adder

fulladder FA00(a[0],b[0],1'b0,s[0],c1s);

fulladder FA01(a[1],b[1],c1s, s[1],c2s);

fulladder FA02(a[2],b[2],c2s, s[2],c3s);

fulladder FA03(a[3],b[3],c3s, s[3],couts);

and a1(x1,s[3],s[2]);

and a2(x2,s[3],s[1]);

or outputcarry( Cout,x1,x2,couts);

fulladder FA00s(0, s[0],1'b0, sum[0],c1ss);

fulladder FA01s(Cout,s[1],c1ss, sum[1],c2ss);

fulladder FA02s(Cout,s[2],c2ss, sum[2],c3ss);

fulladder FA03s(0, s[3],c3ss, sum[3],coutss);

endmodule

module fulladder(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

wire c1,c2,s1;

half g1(a,b,c1,s1);

half g2(s1,cin,c2,s);

or g3(cout,c1,c2);

endmodule

module half(a,b,c,s);

input a,b;

output c,s;

xor g0(s,a,b);

and g1(c,a,b);

endmodule

module test;

reg [3:0] A,B;

wire [3:0] Sum;

wire Cout;

Assg21 uut(.a(A),.b(B),.sum(Sum),.Cout(Cout));

integer i,j;

initial begin

for(i=0;i<9;i=i+1) begin

A=i;

#20;

for(j=0;j<9;j=j+1) begin

B=j;

#20;

end

end

end

endmodule

//Q5:

// half subtractor

module Assg21(

input a,b,

output diff,borrow

);

wire na,nb;

not f0(na,a);

xor f1(diff,a,b);

and f2(borrow,na,b);

endmodule

module test;

reg a,b;

wire diff,borrow;

Assg21 uut(.a(a),.b(b),.diff(diff),.borrow(borrow));

integer i=2'b0;

initial begin

for(i=0;i<4;i=i+1)begin

{a,b}=i;

#20;

end

end

endmodule

//Q6:

//full subtractor

module Assg21(

input a,b,bin,output d,bout);

wire w1,w2,w3;

half\_sub f0(a,b,w1,w2);

half\_sub f1(w1,bin,d,w3);

or f2(bout,w2,w3);

endmodule

module half\_sub(

input a,b,

output diff,borrow );

wire na,nb;

not f0(na,a);

xor f1(diff,a,b);

and f2(borrow,na,b);

endmodule

// test bench for full subtractor

module test;

reg a,b,bin;

wire d,bout;

Assg21 uut(.a(a),.b(b),.bin(bin),.d(d),.bout(bout));

integer i=3'b0;

initial begin

for(i=0;i<8;i=i+1)begin

{a,b,bin}=i;

#20;

end

end

endmodule