module Multiplier(a,b,q);

input[3:0]a,b;

wire[3:0]m0;

wire[4:0]m1;

wire[5:0]m2;

wire[6:0]m3;

wire[7:0]s1,s2,s3;

output reg[7:0]q;

assign m0={4{a[0]}}&b[3:0];

assign m1={4{a[1]}}&b[3:0];

assign m2={4{a[2]}}&b[3:0];

assign m3={4{a[3]}}&b[3:0];

assign s1=m0+(m1<<1);

assign s2=s1+(m2<<1);

assign s3=s2+(m3<<3);

assign s3=q;

endmodule

TEST BENCH CODE

module Multiplier\_test;

// Inputs

reg [3:0] a;

reg [3:0] b;

// Outputs

wire [7:0] q;

// Instantiate the Unit Under Test (UUT)

Multiplier uut (

.a(a),

.b(b),

.q(q)

);

initial begin

// Initialize Inputs

a = 4'b0010;

b = 4'b0100;

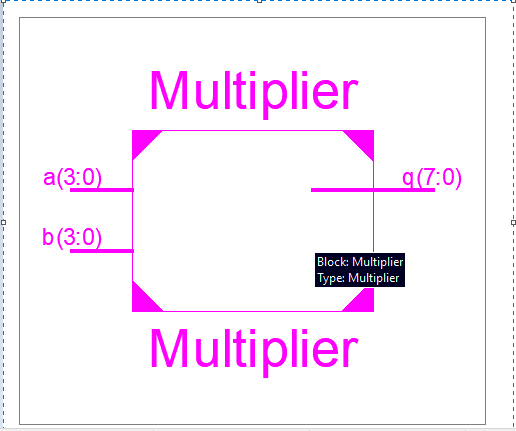
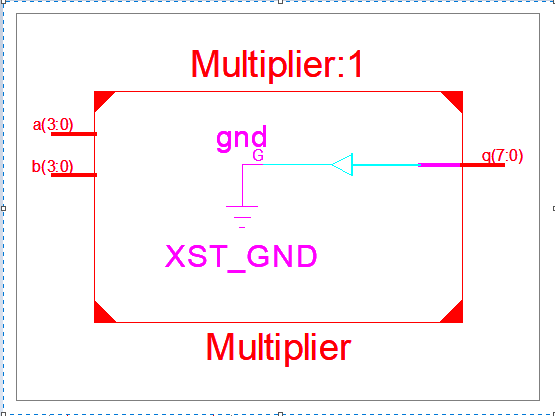
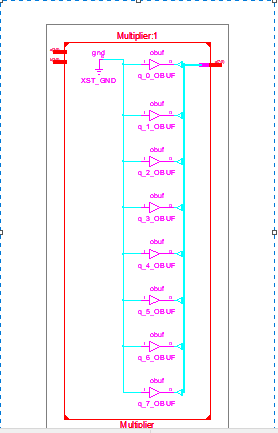
// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

end

endmodule

FULL ADDER CODE

module full\_adder\_1bit(a,b,Cin,sum,Cout);

input a;

input b;

input Cin;

output sum;

output Cout;

assign sum=(a^b^Cin);

assign Cout=((a&b)|(b&Cin)|(a&Cin));

endmodule

module fa;

// Inputs

reg a;

reg b;

reg Cin;

// Outputs

wire sum;

wire Cout;

// Instantiate the Unit Under Test (UUT)

full\_adder\_1bit uut (

.a(a),

.b(b),

.Cin(Cin),

.sum(sum),

.Cout(Cout)

);

initial begin

// Initialize Inputs

a = 0;b=0;Cin=0;#100;

a = 0;b=0;Cin=1;#100;

a = 0;b=1;Cin=0;#100;

a = 0;b=1;Cin=1;#100;

a = 1;b=0;Cin=0;#100;

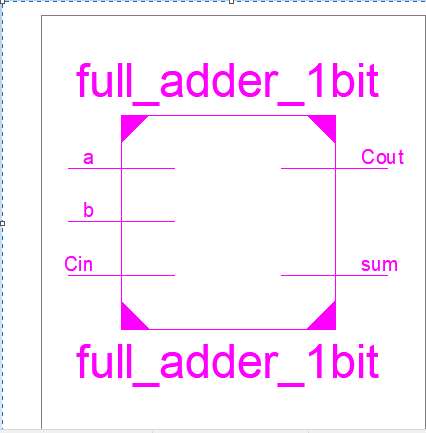
a = 1;b=0;Cin=1;#100;

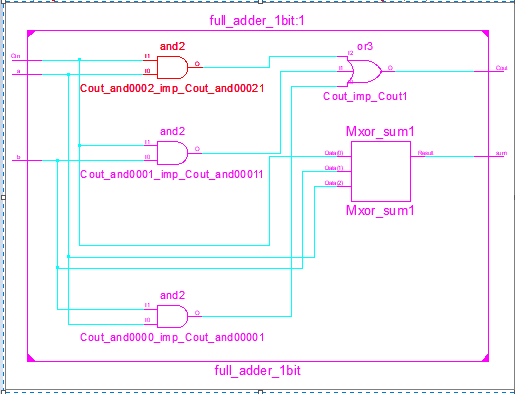
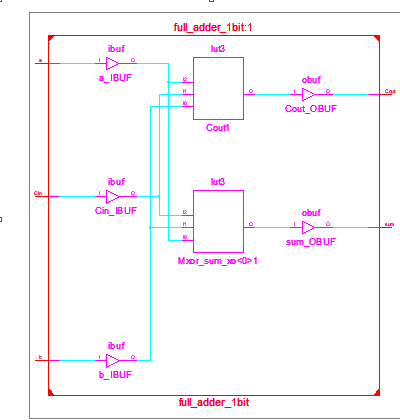
a = 1;b=1;Cin=0;#100;

a = 1;b=1;Cin=1;#100;

end

endmodule





entity logic\_gates is

Port ( a,b : in STD\_LOGIC;

c,d,e,f,g,h : out STD\_LOGIC);

end logic\_gates;

architecture dataflow of logic\_gates is

begin

c<=a and b;

d<=a or b;

e<=not a;

f<=a xor b;

g<=a nand b;

h<=a nor b;

end dataflow;

