module full\_adder\_1bit(S, Cout, A, B, Cin);

input A, B, Cin;

output Cout, S;

assign S = A ^ B ^ Cin;

assign Cout = (A & B) | (B & Cin) | (A & Cin);

endmodule

module addsub\_4bit(Sum, Ovfl, A, B, sub);

input[3:0] A, B;

input sub;

output [3:0] Sum;

output Ovfl;

wire [3:0] Cout;

wire [3:0] invB;

assign invB = sub? ~B + 1'b1: B;

assign Ovfl = Cout[3];

full\_adder\_1bit A1 (Sum[0], Cout[0], A[0], invB[0], 1'b0);

full\_adder\_1bit A2 (Sum[1], Cout[1], A[1], invB[1], Cout[0]);

full\_adder\_1bit A3 (Sum[2], Cout[2], A[2], invB[2], Cout[1]);

full\_adder\_1bit A4 (Sum[3], Cout[3], A[3], invB[3], Cout[2]);

endmodule

module addsub\_4bit\_tb();

reg [7:0] stim;

wire [3:0] Sum;

wire Ovfl;

reg sub;

reg [3:0] temp;

addsub\_4bit DUT(Sum, Ovfl, stim[3:0], stim[7:4], sub);

initial begin

assign sub = 1'b0;

stim = 8'b0;

repeat(100) begin

temp = stim[3:0] + stim[7:4];

#20 stim = $random;

assert(Sum == temp) $display("success");

else

begin

$display("sum is %d, it should be %d", Sum, stim[3:0] + stim[7:4]);

$stop;

end

end

#10;

assign sub = 1'b1;

stim = 8'b0;

repeat(100) begin

temp = stim[3:0] - stim[7:4];

#20 stim = $random;

assert(Sum == temp) $display("success");

else

begin

$display("sum is %d, it should be %d", Sum, stim[3:0] + stim[7:4]);

$stop;

end

end

#10;

$display("all tests passed");

$stop;

end

endmodule