Code:

`timescale 1ns / 1ps

module half\_adder(

input a,

input b,

output c,

output s

);

xor x1(s, a, b);

and a1(c, a, b);

endmodule

module full\_adder(

input a,

input b,

input cin,

output s,

output cout

);

wire carryGenerate, carryPropagate, sum1;

half\_adder

ha0(.a(a), .b(b), .c(carryGenerate), .s(sum1)),

ha1(.a(sum1), .b(cin), .c(carryPropagate), .s(s));

or o1(cout, carryGenerate, carryPropagate);

endmodule

module ripple\_carry4(

input cin,

input [3:0]a,

input [3:0]b,

output [3:0]s,

output cout

);

wire cout0, cout1, cout2;

full\_adder

fa0(a[0], b[0], cin, s[0], cout0),

fa1(a[1], b[1], cout0, s[1], cout1),

fa2(a[2], b[2], cout1, s[2], cout2),

fa3(a[3], b[3], cout2, s[3], cout);

endmodule

module adder8(

input cin,

input [7:0] A,

input[7:0] B,

output[7:0] S,

output Cout

);

wire cout3;

ripple\_carry4

rc0(cin, A[3:0], B[3:0], S[3:0], cout3),

rc1(cout3, A[7:4], B[7:4], S[7:4], Cout);

endmodule

module subtractor8 (

input Bin,

input [7:0] A,

input [7:0] B,

output [7:0] Diff,

output Bout

);

wire [7:0] e;

xor

x0 (e[0], B[0], Bin),

x1 (e[1], B[1], Bin),

x2 (e[2], B[2], Bin),

x3 (e[3], B[3], Bin),

x4 (e[4], B[4], Bin),

x5 (e[5], B[5], Bin),

x6 (e[6], B[6], Bin),

x7 (e[7], B[7], Bin);

adder8

add8 (Bin, A[7:0], e, Diff[7:0], Bout);

endmodule

module and8 (

input [7:0] a, b,

output [7:0] y

);

and

a0 (y[0], a[0], b[0]),

a1 (y[1], a[1], b[1]),

a2 (y[2], a[2], b[2]),

a3 (y[3], a[3], b[3]),

a4 (y[4], a[4], b[4]),

a5 (y[5], a[5], b[5]),

a6 (y[6], a[6], b[6]),

a7 (y[7], a[7], b[7]);

endmodule

module or8 (

input [7:0] a, b,

output [7:0] y

);

or

o0 (y[0], a[0], b[0]),

o1 (y[1], a[1], b[1]),

o2 (y[2], a[2], b[2]),

o3 (y[3], a[3], b[3]),

o4 (y[4], a[4], b[4]),

o5 (y[5], a[5], b[5]),

o6 (y[6], a[6], b[6]),

o7 (y[7], a[7], b[7]);

endmodule

module slt8 (

input [7:0] a, b,

output reg [7:0] y

);

always @ (\*)

begin

if (a < b)

y = 8'b0000\_0001;

else

y = 8'b0000\_0000;

end

endmodule

module mux8 (

input [7:0] d0, d1, d2, d3, d4, d5, d6, d7,

input [2:0] s,

output reg [7:0] y

);

always @ (\*)

begin

case(s)

3'b000: y = d0;

3'b001: y = d1;

3'b010: y = d2;

3'b011: y = d3;

3'b100: y = d4;

3'b101: y = d5;

3'b110: y = d6;

3'b111: y = d7;

default: y = 8'bx;

endcase

end

endmodule

module alu8 (

input [2:0] funSel,

input [7:0] a, b,

output zeroFlag,

output [7:0] result

);

wire [7:0] andab, orab, addab, subab, slthb;

and8 and8bit (a, b, andab);

or8 or8bit (a, b, orab);

adder8 rc8 (funSel[2:2], a, b, addab);

subtractor8 sb8 (funSel[2:2], a, b, subab);

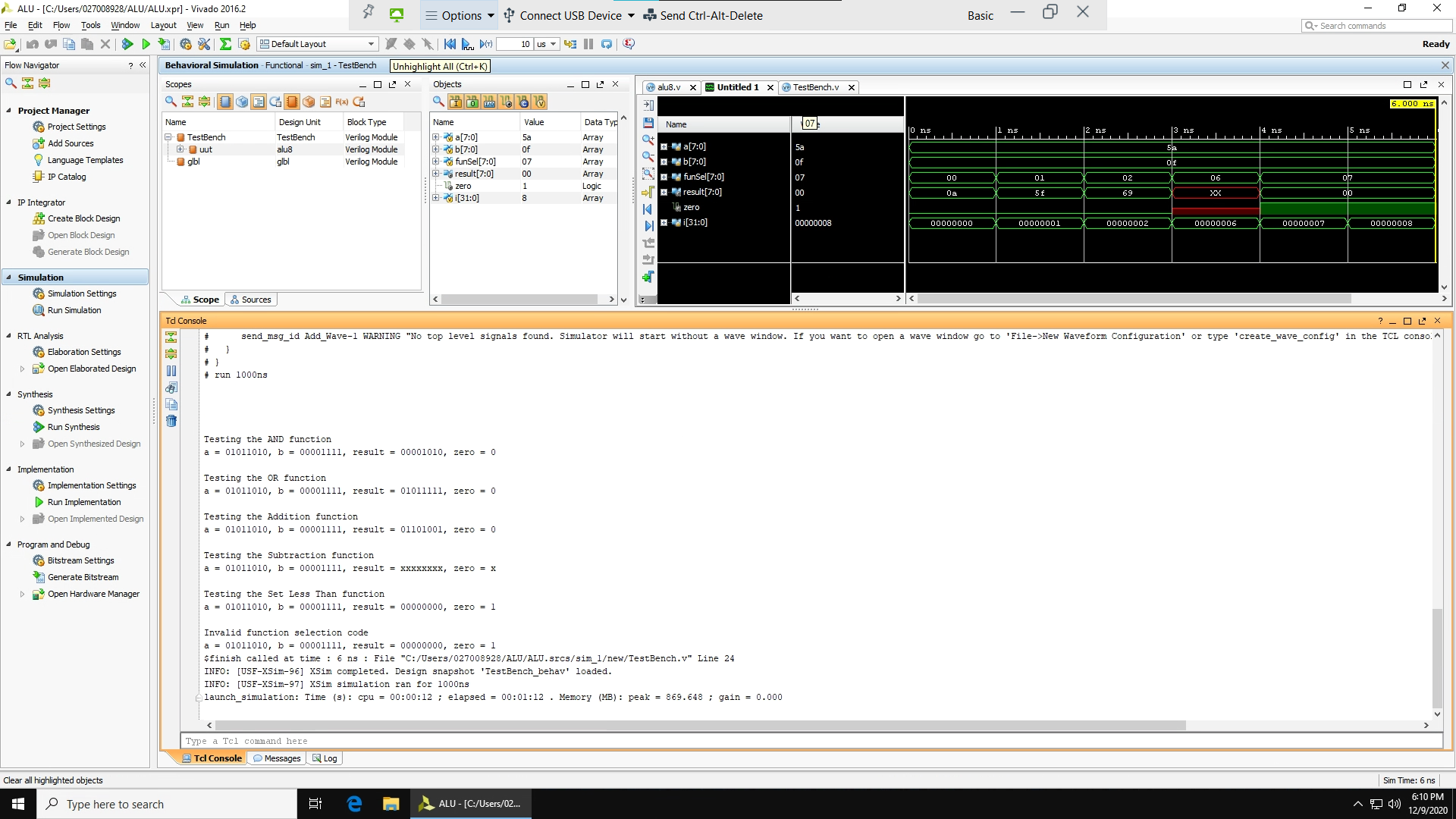
slt8 sbit8 (a, b, slthb);

mux8 m8 (andab, orab, addab, subab, 8'bx, 8'bx, 8'bx, slthb, funSel, result);

nor nor8 (zeroFlag, result[0], result[1], result[2], result[3], result[4], result[5], result[6], result[7]);

endmodule

Screenshot of timing diagram:



Youtube video:

<https://youtu.be/vwZFOF3hXm4>