

DDCO ASSIGNMENT 4

07-09-23

GAUTAM VIJAY HANCHINAL

PES1UG22CS215

SECTION D

ALU.v File:

module fa (input wire i0, i1, cin, output wire sum, cout);

   wire t0, t1, t2;

   xor3 \_i0 (i0, i1, cin, sum);

   and2 \_i1 (i0, i1, t0);

   and2 \_i2 (i1, cin, t1);

   and2 \_i3 (cin, i0, t2);

   or3 \_i4 (t0, t1, t2, cout);

endmodule

module addsub (input wire addsub, i0, i1, cin, output wire sumdiff, cout);

  wire t;

  fa \_i0 (i0, t, cin, sumdiff, cout);

  xor2 \_i1 (i1, addsub, t);

endmodule

module alu\_slice (input wire [1:0] op, input wire i0, i1, cin, output wire o, cout);

   wire t\_sumdiff, t\_and, t\_or, t\_andor;

   addsub \_i0 (op[0], i0, i1, cin, t\_sumdiff, cout);

   and2 \_i1 (i0, i1, t\_and);

   or2 \_i2 (i0, i1, t\_or);

   mux2 \_i3 (t\_and, t\_or, op[0], t\_andor);

   mux2 \_i4 (t\_sumdiff, t\_andor, op[1], o);

endmodule

module alu (input wire [1:0] op, input wire [15:0] i0, i1,

    output wire [15:0] o, output wire cout);

   wire  [14:0] c;

   alu\_slice \_i0 (op, i0[0], i1[0], op[0] , o[0], c[0]);

   alu\_slice \_i1 (op, i0[1], i1[1], c[0], o[1], c[1]);

   alu\_slice \_i2 (op, i0[2], i1[2], c[1], o[2], c[2]);

   alu\_slice \_i3 (op, i0[3], i1[3], c[2], o[3], c[3]);

   alu\_slice \_i4 (op, i0[4], i1[4], c[3], o[4], c[4]);

   alu\_slice \_i5 (op, i0[5], i1[5], c[4], o[5], c[5]);

   alu\_slice \_i6 (op, i0[6], i1[6], c[5], o[6], c[6]);

   alu\_slice \_i7 (op, i0[7], i1[7], c[6], o[7], c[7]);

   alu\_slice \_i8 (op, i0[8], i1[8], c[7], o[8], c[8]);

   alu\_slice \_i9 (op, i0[9], i1[9], c[8], o[9], c[9]);

   alu\_slice \_i10 (op, i0[10], i1[10], c[9] , o[10], c[10]);

   alu\_slice \_i11 (op, i0[11], i1[11], c[10], o[11], c[11]);

   alu\_slice \_i12 (op, i0[12], i1[12], c[11], o[12], c[12]);

   alu\_slice \_i13 (op, i0[13], i1[13], c[12], o[13], c[13]);

   alu\_slice \_i14 (op, i0[14], i1[14], c[13], o[14], c[14]);

   alu\_slice \_i15 (op, i0[15], i1[15], c[14], o[15], cout);

endmodule

Lib File:

module invert (input wire i, output wire o);

   assign o = !i;

endmodule

module and2 (input wire i0, i1, output wire o);

  assign o = i0 & i1;

endmodule

module or2 (input wire i0, i1, output wire o);

  assign o = i0 | i1;

endmodule

module xor2 (input wire i0, i1, output wire o);

  assign o = i0 ^ i1;

endmodule

module nand2 (input wire i0, i1, output wire o);

   wire t;

   and2 and2\_0 (i0, i1, t);

   invert invert\_0 (t, o);

endmodule

module nor2 (input wire i0, i1, output wire o);

   wire t;

   or2 or2\_0 (i0, i1, t);

   invert invert\_0 (t, o);

endmodule

module xnor2 (input wire i0, i1, output wire o);

   wire t;

   xor2 xor2\_0 (i0, i1, t);

   invert invert\_0 (t, o);

endmodule

module and3 (input wire i0, i1, i2, output wire o);

   wire t;

   and2 and2\_0 (i0, i1, t);

   and2 and2\_1 (i2, t, o);

endmodule

module or3 (input wire i0, i1, i2, output wire o);

   wire t;

   or2 or2\_0 (i0, i1, t);

   or2 or2\_1 (i2, t, o);

endmodule

module nor3 (input wire i0, i1, i2, output wire o);

   wire t;

   or2 or2\_0 (i0, i1, t);

   nor2 nor2\_0 (i2, t, o);

endmodule

module nand3 (input wire i0, i1, i2, output wire o);

   wire t;

   and2 and2\_0 (i0, i1, t);

   nand2 nand2\_1 (i2, t, o);

endmodule

module xor3 (input wire i0, i1, i2, output wire o);

   wire t;

   xor2 xor2\_0 (i0, i1, t);

   xor2 xor2\_1 (i2, t, o);

endmodule

module xnor3 (input wire i0, i1, i2, output wire o);

   wire t;

   xor2 xor2\_0 (i0, i1, t);

   xnor2 xnor2\_0 (i2, t, o);

endmodule

module mux2 (input wire i0, i1, j, output wire o);

  assign o = (j==0)?i0:i1;

endmodule

module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);

  wire  t0, t1;

  mux2 mux2\_0 (i[0], i[1], j1, t0);

  mux2 mux2\_1 (i[2], i[3], j1, t1);

  mux2 mux2\_2 (t0, t1, j0, o);

endmodule

module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire o);

  wire  t0, t1;

  mux4 mux4\_0 (i[0:3], j2, j1, t0);

  mux4 mux4\_1 (i[4:7], j2, j1, t1);

  mux2 mux2\_0 (t0, t1, j0, o);

endmodule

Test Bench File:

`timescale 1 ns / 100 ps

`define TESTVECS 16

module tb;

  reg clk, reset;

  reg [1:0] op; reg [15:0] i0, i1;

  wire [15:0] o; wire cout;

  reg [33:0] test\_vecs [0:(`TESTVECS-1)];

  integer i;

  initial begin $dumpfile("tb\_alu.vcd"); $dumpvars(0,tb); end

  initial begin reset = 1'b1; #12.5 reset = 1'b0; end

  initial clk = 1'b0; always #5 clk = ~clk;

  initial begin

    test\_vecs[0][33:32] = 2'b00;

   test\_vecs[0][31:16] = 16'h0000;

  test\_vecs[0][15:0] = 16'h0000;

    test\_vecs[1][33:32] = 2'b00;

test\_vecs[1][31:16] = 16'haa55;

test\_vecs[1][15:0] = 16'h55aa;

    test\_vecs[2][33:32] = 2'b00;

test\_vecs[2][31:16] = 16'hffff;

test\_vecs[2][15:0] = 16'h0001;

    test\_vecs[3][33:32] = 2'b00;

 test\_vecs[3][31:16] = 16'h0001;

test\_vecs[3][15:0] = 16'h7fff;

    test\_vecs[4][33:32] = 2'b01; test\_vecs[4][31:16] = 16'h0000;test\_vecs[4][15:0] = 16'h0000;

    test\_vecs[5][33:32] = 2'b01; test\_vecs[5][31:16] = 16'haa55;test\_vecs[5][15:0] = 16'h55aa;

    test\_vecs[6][33:32] = 2'b01; test\_vecs[6][31:16] = 16'hffff;test\_vecs[6][15:0] = 16'h0001;

    test\_vecs[7][33:32] = 2'b01; test\_vecs[7][31:16] = 16'h0001;test\_vecs[7][15:0] = 16'h7fff;

    test\_vecs[8][33:32] = 2'b10; test\_vecs[8][31:16] = 16'h0000;test\_vecs[8][15:0] = 16'h0000;

    test\_vecs[9][33:32] = 2'b10; test\_vecs[9][31:16] = 16'haa55;test\_vecs[9][15:0] = 16'h55aa;

    test\_vecs[10][33:32] = 2'b10; test\_vecs[10][31:16] = 16'hffff;test\_vecs[10][15:0] = 16'h0001;

    test\_vecs[11][33:32] = 2'b10; test\_vecs[11][31:16] = 16'h0001;test\_vecs[11][15:0] = 16'h7fff;

    test\_vecs[12][33:32] = 2'b11; test\_vecs[12][31:16] = 16'h0000;test\_vecs[12][15:0] = 16'h0000;

    test\_vecs[13][33:32] = 2'b11; test\_vecs[13][31:16] = 16'haa55;test\_vecs[13][15:0] = 16'h55aa;

    test\_vecs[14][33:32] = 2'b11; test\_vecs[14][31:16] = 16'hffff;test\_vecs[14][15:0] = 16'h0001;

    test\_vecs[15][33:32] = 2'b11; test\_vecs[15][31:16] = 16'h0001;test\_vecs[15][15:0] = 16'h7fff;

  end

  initial {op, i0, i1} = 0;

  alu alu\_0 (op, i0, i1, o, cout);

  initial begin

    #6 for(i=0;i<`TESTVECS;i=i+1)

      begin #10 {op, i0, i1} = test\_vecs[i]; end

    #100 $finish;

  end

endmodule

cmd screenshot:



GTKWAVE output:



