Digital Design and Computer Organisation Laboratory

UE23CS252A

3rd Semester, Academic Year 2024-25

Date: 10/09/2024

Name: KEERTHAN P.V	SRN: PES2UG23CS272	Section 3E

Week:5 Program Number:1

TITLE:

WRITE A VERILOG PROGRAM TO MODEL A 16 BIT ALU THAT CAN PERFORM FOUR OPERATIONS-ADDITION, SUBTRACTION, AND along with OR OPERATION. ALL THESE OPERATIONS GENERATE A SIXTEEN BIT RESULT.SHOW THE VVP OUTPUT.DISPLAY THE SIMULATION WAVEFORM USING GTKWAVE. VERIFY THE OUTPUT AND WAVEFORM WITH THE TRUTH TABLE

I. Verilog Code Screenshot

1.libb.v

```
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;

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
```

```
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
```

2.alutb.v

```
`timescale 1 ns / 100 ps
`define TESTVECS 8

module tb;
  reg clk, reset, wr, sel;
  reg [1:0] op;
  reg [2:0] rd_addr_a, rd_addr_b, wr_addr; reg [15:0] d_in;
  wire [15:0] d_out_a, d_out_b;
  reg [28:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_reg_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][28] = 1'b0; test_vecs[0][27] = 1'b1; test_vecs[0][26:25] = 2'bxx;
    test_vecs[0][24:22] = 3'ox; test_vecs[0][21:19] = 3'ox;
    test_vecs[0][18:16] = 3'h3; test_vecs[0][15:0] = 16'hcdef;

    test_vecs[1][28] = 1'b0; test_vecs[1][27] = 1'b1; test_vecs[1][26:25] = 2'bxx;
    test_vecs[1][24:22] = 3'ox; test_vecs[1][21:19] = 3'ox;
    test_vecs[1][18:16] = 3'o7; test_vecs[1][15:0] = 16'h3210;

    test_vecs[2][28] = 1'b0; test_vecs[2][27] = 1'b1; test_vecs[2][26:25] = 2'bxx;
    test_vecs[2][24:22] = 3'o3; test_vecs[2][21:19] = 3'o7;
    test_vecs[2][18:16] = 3'o5; test_vecs[2][15:0] = 16'h4567;

    test_vecs[3][28] = 1'b0; test_vecs[3][27] = 1'b1; test_vecs[3][26:25] = 2'bxx;
    test_vecs[3][24:22] = 3'o1; test_vecs[3][21:19] = 3'o5;
    test_vecs[3][18:16] = 3'o1; test_vecs[3][15:0] = 16'hba98;

    test_vecs[4][28] = 1'b0; test_vecs[4][27] = 1'b0; test_vecs[4][26:25] = 2'bxx;
    test_vecs[4][24:22] = 3'o1; test_vecs[4][21:19] = 3'o5;
    test_vecs[4][18:16] = 3'o1; test_vecs[4][15:0] = 16'hxxxx;

    test_vecs[5][28] = 1'b1; test_vecs[5][27] = 1'b1; test_vecs[5][26:25] = 2'b00;
    test_vecs[5][24:22] = 3'o1; test_vecs[5][21:19] = 3'o5;
    test_vecs[5][18:16] = 3'o2; test_vecs[5][15:0] = 16'hxxxx;

    test_vecs[6][28] = 1'b1; test_vecs[6][27] = 1'b1; test_vecs[6][26:25] = 2'b01;
    test_vecs[6][24:22] = 3'o2; test_vecs[6][21:19] = 3'o7;
    test_vecs[6][18:16] = 3'o4; test_vecs[6][15:0] = 16'hxxxx;

    test_vecs[7][28] = 1'b1; test_vecs[7][27] = 1'b0; test_vecs[7][26:25] = 2'b01;
    test_vecs[7][24:22] = 3'o4; test_vecs[7][21:19] = 3'o4;
    test_vecs[7][18:16] = 3'ox; test_vecs[7][15:0] = 16'hxxxx;
  end
  initial {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;
  reg_alu reg_alu_0 (clk, reset, sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in,
  d_out_a, d_out_b, cout);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {sel, wr, op, rd_addr_a, rd_addr_b, wr_addr, d_in}=test_vecs[i];
    end
    #100 $finish;
  end


endmodule
```

3.alu.v

```
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
    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
);
```

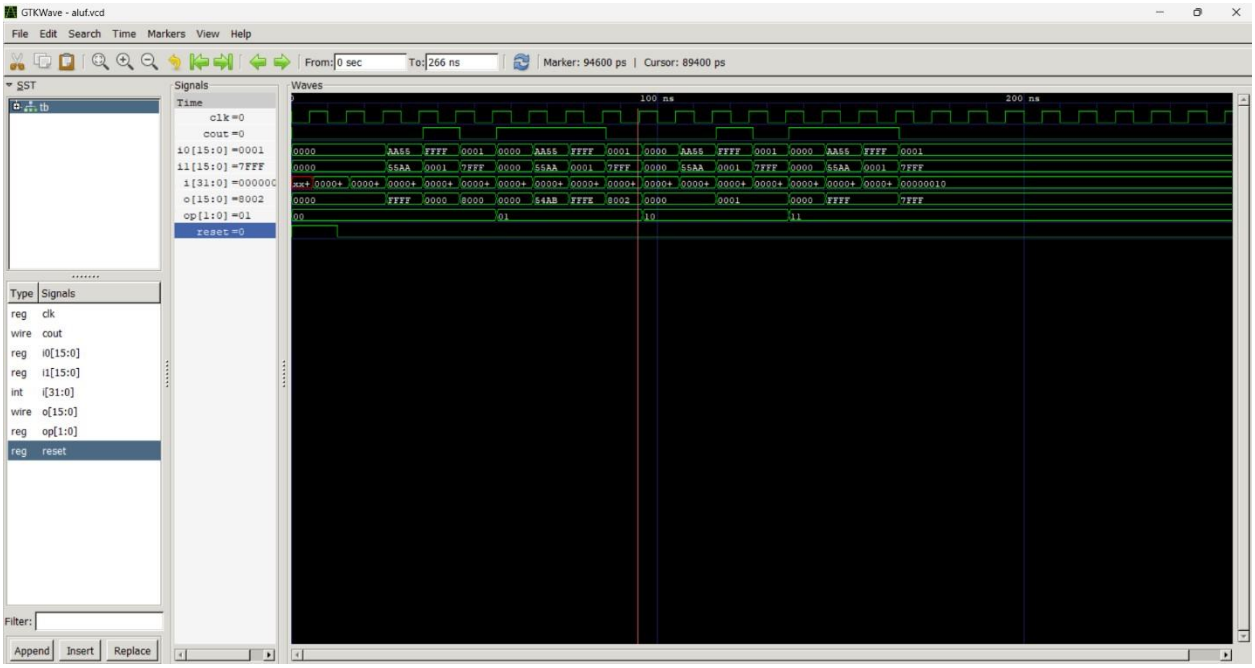
```
alu_slice _i14 (
    op, i0[14], i1[14], c[13], o[14], c[14]
);

alu_slice _i15 (
    op, i0[15], i1[15], c[14], o[15], cout
);

endmodule
```

```
);

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]
);
```

II. Verilog VVP Output Screen Shot

```
C:\iverilog\bin>vvp test
VCD info: dumpfile aluf.vcd opened for output.
Time: 0 | op: 00 | i0: 16'h0000 | i1: 16'h0000 | Output: o = 16'h0000, cout = 0
Time: 260 | op: 00 | i0: 16'haa55 | i1: 16'h55aa | Output: o = 16'hffff, cout = 0
Time: 360 | op: 00 | i0: 16'hffff | i1: 16'h0001 | Output: o = 16'h0000, cout = 1
Time: 460 | op: 00 | i0: 16'h0001 | i1: 16'h7fff | Output: o = 16'h8000, cout = 0
Time: 560 | op: 01 | i0: 16'h0000 | i1: 16'h0000 | Output: o = 16'h0000, cout = 1
Time: 660 | op: 01 | i0: 16'haa55 | i1: 16'h55aa | Output: o = 16'h54ab, cout = 1
Time: 760 | op: 01 | i0: 16'hffff | i1: 16'h0001 | Output: o = 16'hfffe, cout = 1
Time: 860 | op: 01 | i0: 16'h0001 | i1: 16'h7fff | Output: o = 16'h8002, cout = 0
Time: 960 | op: 10 | i0: 16'h0000 | i1: 16'h0000 | Output: o = 16'h0000, cout = 0
Time: 1060 | op: 10 | i0: 16'haa55 | i1: 16'h55aa | Output: o = 16'h0000, cout = 0
Time: 1160 | op: 10 | i0: 16'hffff | i1: 16'h0001 | Output: o = 16'h0001, cout = 1
Time: 1260 | op: 10 | i0: 16'h0001 | i1: 16'h7fff | Output: o = 16'h0001, cout = 0
Time: 1360 | op: 11 | i0: 16'h0000 | i1: 16'h0000 | Output: o = 16'h0000, cout = 1
Time: 1460 | op: 11 | i0: 16'haa55 | i1: 16'h55aa | Output: o = 16'hffff, cout = 1
Time: 1560 | op: 11 | i0: 16'hffff | i1: 16'h0001 | Output: o = 16'hffff, cout = 1
Time: 1660 | op: 11 | i0: 16'h0001 | i1: 16'h7fff | Output: o = 16'h7fff, cout = 0
```

III. GTKWAVE Screenshot



IV. Output Table to be completed and included

	op[1:0]	i0[15:0]	i1[15:0]	Output
TESTVECTOR0	2’b00	16’h0000	16’h0000	16’h0000
TESTVECTOR1	2’b00	16’haa55	16’h55aa	16’hffff
TESTVECTOR2	2’b00	16'hffff	16'h0001	16’h0000
TESTVECTOR3	2’b00	16'h0001	16'h7fff	16’h8000
TESTVECTOR4	2'b01	16'h0000	16'h0000	16’h0000
TESTVECTOR5	2'b01	16’haa55	16’h55aa	16’h54ab
TESTVECTOR6	2'b01	16'hffff	16'h0001	16’hfffe
TESTVECTOR7	2'b01	16'h0001	16'h7fff	16’h8002
TESTVECTOR8	2’b10	16’h0000	16’h0000	16’h0000
TESTVECTOR9	2’b10	16’haa55	16’h55aa	16’h0000
TESTVECTOR10	2’b10	16'hffff	16'h0001	16’h0001
TESTVECTOR11	2’b10	16'h0001	16'h7fff	16’h0001
TESTVECTOR12	2’b11	16’h0000	16’h0000	16’h0000
TESTVECTOR13	2’b11	16’haa55	16’h55aa	16’hffff
TESTVECTOR14	2’b11	16'hffff	16'h0001	16’hffff
TESTVECTOR15	2’b11	16'h0001	16'h7fff	16’h7fff

Disclaimer:

- The programs and output submitted is duly written, verified and executed by me.
- I have not copied from any of my peers nor from the external resource such as internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature: KEERTHAN P.V
Name: KEERTHAN P.V
SRN: PES2UG23CS272
Section:3E

Date:
10/09/2024