Digital Design and Computer Organisation Laboratory

3rd Semester, Academic Year 2025

Date:06-10-2025

Name: Prajwal M	SRN:PES2UG24CS910	Section:C

Week Number: 7 Program Number: 1

TOPIC :IMPLEMENTATION OF ALU

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 (i0, cin, t1);
and2 _i3 (i1, cin, 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
```

tb_alu.v

tb_alu.v

```
`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;
reg [31:0] op_name; // Operation name as string
initial
begin
$dumpfile("tb_alu.vcd");
$dumpvars(0,tb);
$display("Op | Input A  | Input B  | Output   | Carry | Operation");
$display("---|----------|----------|----------|-------|----------");
$monitor("%2b | %8h | %8h | %8h |   %1b", op, i0, i1, o, cout);
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
 for(i=0;i<`TESTVECS ; i=i+1)
  begin
 #10
  {op, i0, i1}=test_vecs[i];
  #1; // Small delay to let outputs settle
  case(op)
    2'b00: $display("Test %2d: ADD operation", i+1);
    2'b01: $display("Test %2d: SUB operation", i+1);
    2'b10: $display("Test %2d: AND operation", i+1);
    2'b11: $display("Test %2d: OR  operation", i+1);
  endcase
 end
 #10 $display("\n Completed!");
  #100 $finish;
 end
endmodule
```

Vvp output

```
D:\iverilog lab\week 8>vvp alu
VCD info: dumpfile tb_alu.vcd opened for output.
Op | Input A  | Input B  | Output   | Carry | Operation
---|----------|----------|----------|-------|----------
00 |     0000 |     0000 |     0000 |   0
Test  1: ADD operation
00 |     aa55 |     55aa |     ffff |   0
Test  2: ADD operation
00 |     ffff |     0001 |     0000 |   1
Test  3: ADD operation
00 |     0001 |     7fff |     8000 |   0
Test  4: ADD operation
01 |     0000 |     0000 |     0000 |   1
Test  5: SUB operation
01 |     aa55 |     55aa |     54ab |   1
Test  6: SUB operation
01 |     ffff |     0001 |     fffe |   1
Test  7: SUB operation
01 |     0001 |     7fff |     8002 |   0
Test  8: SUB operation
10 |     0000 |     0000 |     0000 |   0
Test  9: AND operation
10 |     aa55 |     55aa |     0000 |   0
Test 10: AND operation
10 |     ffff |     0001 |     0001 |   1
Test 11: AND operation
10 |     0001 |     7fff |     0001 |   0
Test 12: AND operation
11 |     0000 |     0000 |     0000 |   1
Test 13: OR  operation
11 |     aa55 |     55aa |     ffff |   1
Test 14: OR  operation
11 |     ffff |     0001 |     ffff |   1
Test 15: OR  operation
11 |     0001 |     7fff |     7fff |   0
Test 16: OR  operation

 Completed!

D:\iverilog lab\week 8>
D:\iverilog lab\week 8>gtkwave tb_alu.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[286000] end time.
```

GTKWave output