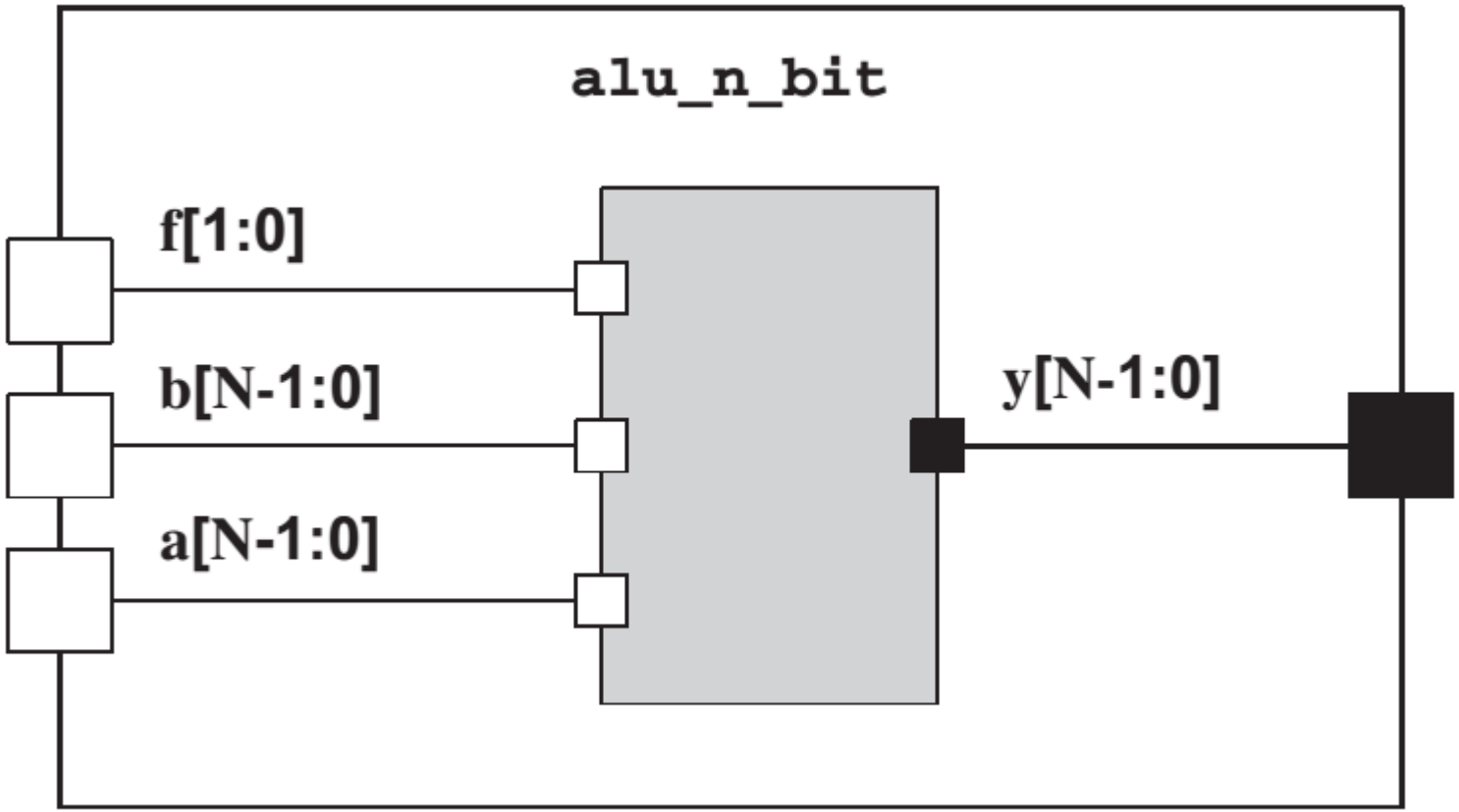
آزمایشگاه سیستم های دیجیتال2

آزمایش 6

واحد محاسبه و منطق (ALU)

Assign Statements

ALU N bit



Assign Statements

ALU N bit

```
module alu_n_bit (a, b, f, y );
    parameter N=4;
    input [N-1:0] a, b;
    input [1:0] f;
    output [N-1:0] y;
    reg [N-1:0] y;
    always @ (a, b, f)
    begin
        casez ( f )
            2'b00 : y = a + b;
            2'b01 : y = a - b;
            2'b10 : y = a & b;
            2'b11 : y = a ^ b;
            default: y = 0;
        endcase
    end

endmodule
```

Assign Statements

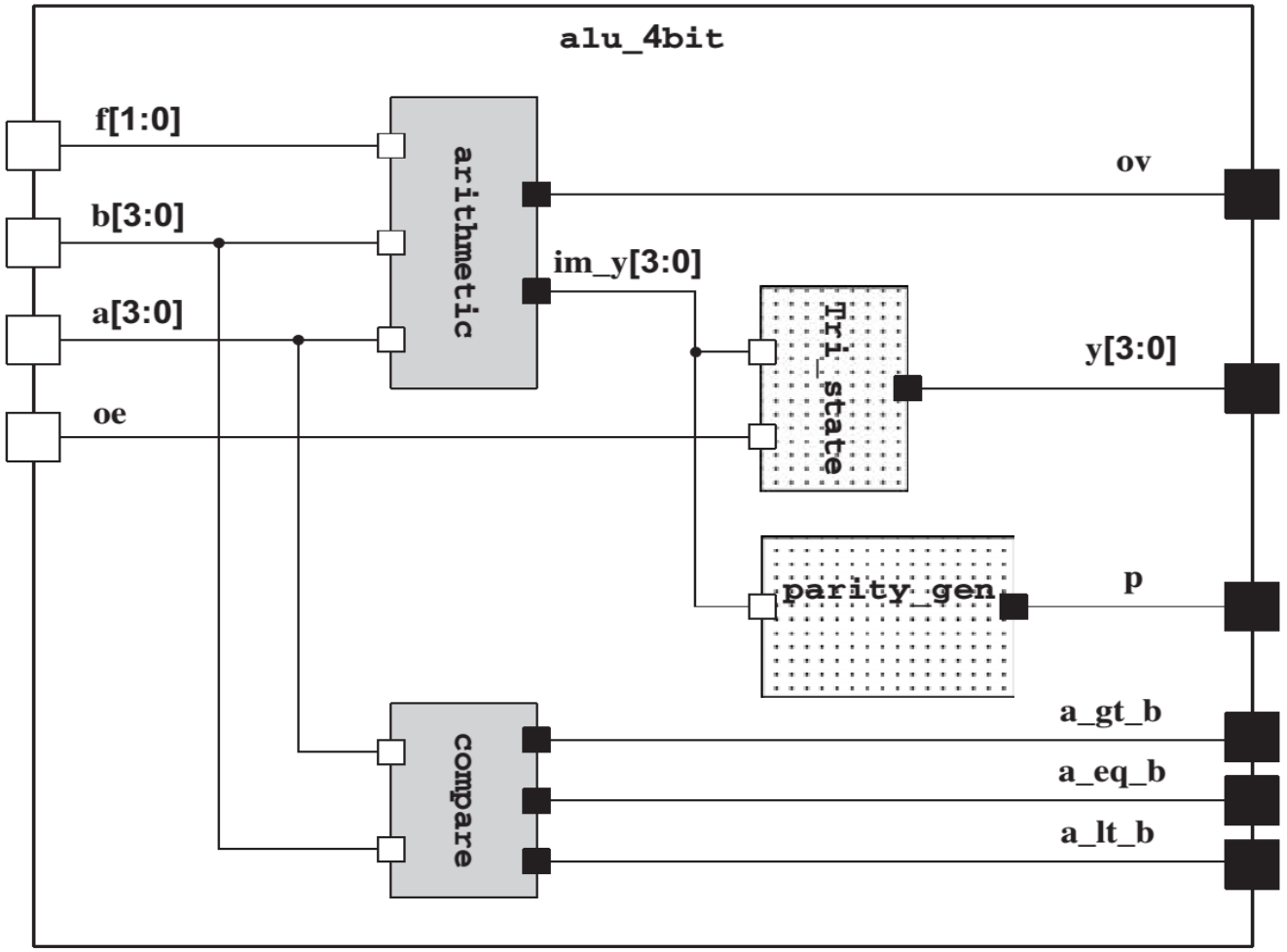
ALU N bit

```
module test_alu;

    // Inputs
    reg [3:0] a;
    reg [3:0] b;
    reg [1:0] f;

    // Outputs
    wire [3:0] y;

    // Instantiate the Unit Under Test
    alu_n_bit uut (
        .a(a),
        .b(b),
        .f(f),
        .y(y)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        f = 0;
        #50 a = 4'b0110; b = 4'b1001;
        #50 f = 1;
        #50 f = 2'b10;
        #50 f = 2'b11;
        #50 a = 4'bzzzz; b = 4'b1001;
        #50 f = 0;
        #50 f = 2'b10;
        #50 a = 4'bzzzz; b = 4'bzzzz;
        #50 f = 1;
        #50 f = 2'b00;
        #50 f = 2'b11;
        #50 a = 4'b1110; b = 4'b1001; f = 0;
    end
endmodule
```

Combinational Synthesis

Multi-Function

ALU Block



Combinational Synthesis

Multi-Function

ALU Verilog code 1

```
`timescale 1ns/100ps

module alu_4bit (a, b, f, oe, y, p, ov, a_gt_b,
                                a_eq_b, a_lt_b);
    input [3:0] a, b;
    input [1:0] f;
    input oe;
    output [3:0] y;
    output p, ov, a_gt_b, a_eq_b, a_lt_b;
    reg ov, a_gt_b, a_eq_b, a_lt_b;

    reg [4:0] im_y;
```

Combinational Synthesis

Multi-Function

ALU Verilog code 2

```
always @( a or b or f ) begin : arithmethic
   ov = 1'b0;
   im_y = 0;
   case ( f )
      2'b00 :
         begin
            im_y = a + b;
            if ( im_y>5'b01111 ) ov = 1'b1;
         end
      2'b01 :
         begin
            im_y = a - b;
            if ( im_y>5'b01111 ) ov = 1'b1;
         end
      2'b10 : im_y[3:0] = a & b;
      2'b11 : im_y[3:0] = a ^ b;
      default: im_y[3:0] = 4'b0000;
   endcase
end
```

Combinational Synthesis

Multi-Function

ALU Verilog code 3

```
    always @( a or b ) begin : compare
       if ( a > b ) { a_gt_b, a_eq_b, a_lt_b } = 3'b100;
       else if ( a < b ) { a_gt_b, a_eq_b, a_lt_b } = 3'b001;
       else { a_gt_b, a_eq_b, a_lt_b } = 3'b010;
    end

    assign p = ^ im_y[3:0];

    assign y = oe ? im_y[3:0] : 4'bz;

endmodule
```

Combinational Synthesis

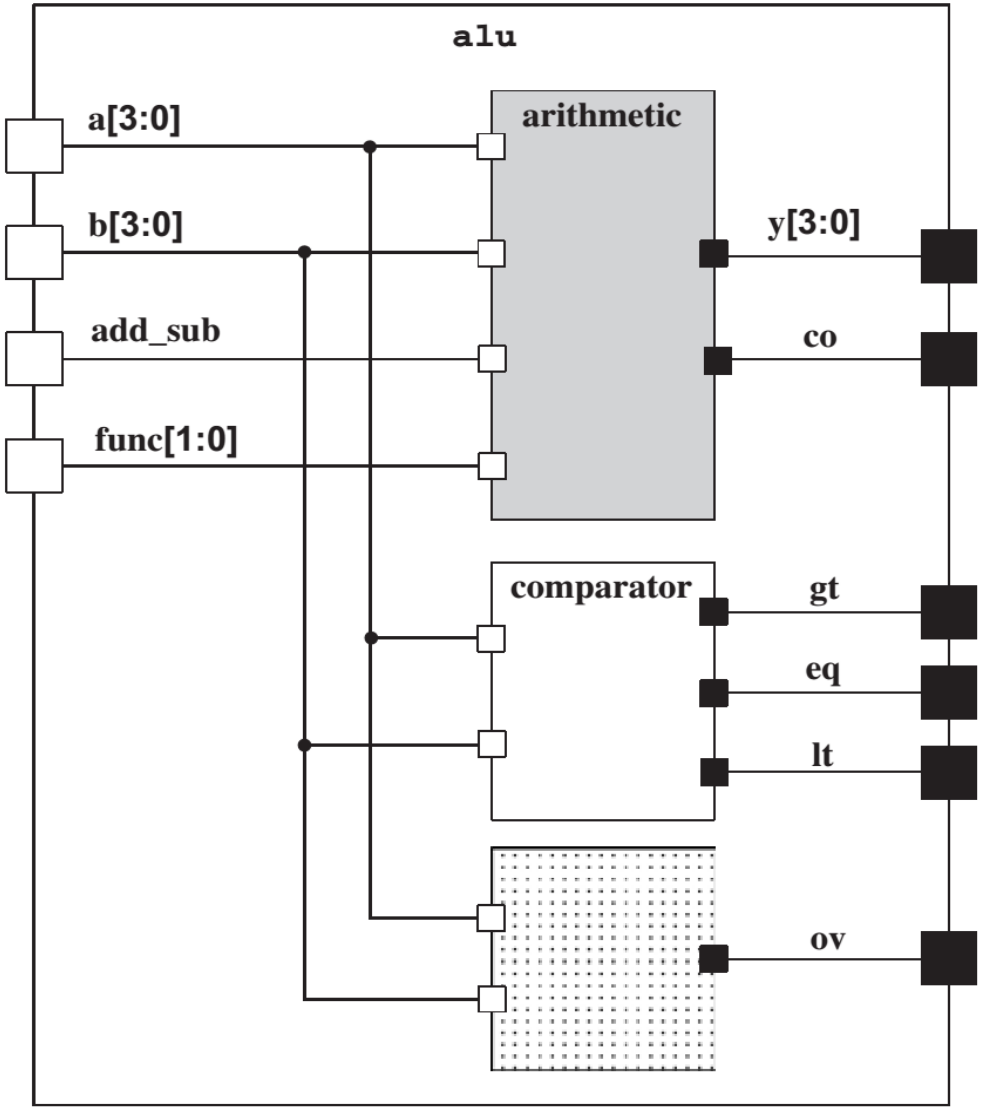
Multi-Function ALU

Testbench

```
module tb_alu;
    // Inputs
    reg [3:0] a;
    reg [3:0] b;
    reg [1:0] f;
    reg oe;

    // Outputs
    wire [3:0] y;
    wire p;
    wire ov;
    wire a_gt_b;
    wire a_eq_b;
    wire a_lt_b;
    // Instantiate the Unit Under Test
    alu_4bit uut (
        .a(a),
        .b(b),
        .f(f),
        .oe(oe),
        .y(y),
        .p(p),
        .ov(ov),
        .a_gt_b(a_gt_b),
        .a_eq_b(a_eq_b),
        .a_lt_b(a_lt_b));
    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        f = 0;
        oe = 0;
        #50 a = 4'b1010 ; b = 4'b0101;
        #50 oe = 1;
        #50 f = 1;
        #50 f = 2;
        #50 f = 3;
        #50 oe = 0;
        #50 a = 4'b0010 ; b = 4'b0100;f = 0;
        #50 oe = 1;
        #50 f = 1;
        #50 f = 2;
        #50 f = 3;
        #50 oe = 0;
        #50 a = 4'b1110 ; b = 4'b1110;f = 0;
        #50 oe = 1;
        #50 f = 1;
        #50 f = 2;
        #50 f = 3;
        #50 oe = 0;
    end
endmodule
```

Combinational Synthesis

ALU Block Diagram



Combinational Synthesis

ALU Block Diagram

ALU verilog code 1

```
module compartor ( a, b, gt, eq, lt );
    input [3:0] a, b;
    output gt, eq, lt;
    assign gt = (a>b) ? 1'b1 : 1'b0;
    assign eq = (a==b) ? 1'b1 : 1'b0;
    assign lt = (a<b) ? 1'b1 : 1'b0;

endmodule
```

Combinational Synthesis

ALU Block Diagram

ALU verilog code 2

```
module alu ( a, b, add_sub, func, y, co, gt, eq, lt, ov );
    input [3:0] a, b;
    input add_sub;
    input [1:0] func;
    output [3:0] y;
    reg [3:0] y;
    output co, gt, eq, lt, ov;
    reg co;
```

Combinational Synthesis

ALU Block Diagram

ALU verilog code 3

```
always @( a or b or add_sub or func ) : arithmetic
   case (func)
      2'b00 :
         if (add_sub) { co, y } = a - b;
         else { co, y } = a + b;
      2'b01 : { co, y } = { 1'b0, a };
      2'b10 : { co, y } = { 1'b0, a & b };
      2'b11 : { co, y } = { 1'b0, ~a };
      default: { co, y } = , 5'b00000 ;
   endcase

compartor cmp ( a, b, gt, eq, lt );
```

Combinational Synthesis

ALU Block Diagram

ALU verilog code 4

```
    assign ov = (func==2'b00)
         ? ((a[3] & b[3] & ~y[3]) | (~a[3] & ~b[3] & y[3]))
         : 1'b0;
endmodule
```

Combinational Synthesis

ALU Block Diagram

ALU Test 1

```
module tb_alu;

    // Inputs
    reg [3:0] a;
    reg [3:0] b;
    reg add_sub;
    reg [1:0] func;

    // Outputs
    wire [3:0] y;
    wire co;
    wire gt;
    wire eq;
    wire lt;
    wire ov;
```

Combinational Synthesis

ALU Block Diagram

ALU Test 2

```
// Instantiate the Unit Under Test (UUT)
alu uut (
    .a(a),
    .b(b),
    .add_sub(add_sub),
    .func(func),
    .y(y),
    .co(co),
    .gt(gt),
    .eq(eq),
    .lt(lt),
    .ov(ov)
);
```

Combinational Synthesis

ALU Block Diagram

ALU Test 3

```
    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        add_sub = 0;
        func = 0;
        #50 a = 4'b1010; b = 4'b0101;
        #50 add_sub = 1;
        #50 func = 1;
        #50 func = 2;
        #50 func = 3;
        #50 a = 4'b1110; b = 4'b1110; add_sub = 0; func = 0;
        #50 add_sub = 1;
        #50 func = 1;
        #50 func = 2;
        #50 func = 3;
        #50 a = 4'b0110; b = 4'b1010; add_sub = 0; func = 0;
        #50 add_sub = 1;
        #50 func = 1;
        #50 func = 2;
        #50 func = 3;
    end
endmodule
```