```
module alu (input [31:0] a, input [31:0] b, input [2:0] sel, output [31:0] out_data);
    reg [31:0] out;
    assign out_data = out;
    reg [32:0] carry;
    always @(*) begin
        case (sel)
            3'b000: out <= $signed(a) + $signed(b);
            3'b001: {carry, out} <= $unsigned(a) + $unsigned(b);
            3'b010: out <= $signed(a) - $signed(b);
            3'b011: out <= a - b;
            3'b100: out <= a & b;
            3'b101: out <= a | b;
            3'b110: out <= a >>> b;
            3'b111: out <= a >> b;
            default: out <= a + b;
        endcase
    end
endmodule
```

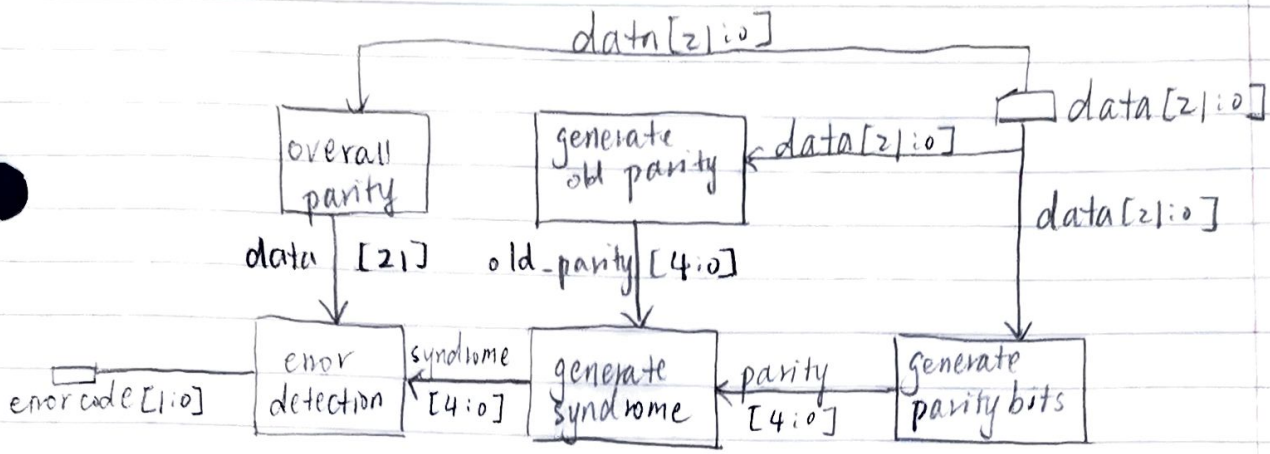
```
`timescale 1ns / 1ns

module alu_testbench();
    reg [31:0] a;
    reg [31:0] b;
    reg [2:0] sel;

    wire[31:0] out_data;
    reg fi = 1'b0;

    alu tu(
        .a(a),
        .b(b),
        .sel(sel),
        .out_data(out_data)
    );

    initial begin
        $dumpfile("alu_testbench.vcd");
        $dumpvars(0, alu_testbench);

        a = 32'd10;
        b = 32'd2;
        b = -b;
        sel = 3'b0;
        #1;
        $display("Signed Add: a = %0d, b = %0d, out = %0d", a, $signed(b), out_data);
        if (out_data != 32'd8) begin
            fi = 1'b1;
        end
        sel = 3'b1;
        a = 32'd1;
        #1;
        $display("Unsiged Add: a = %0d, b = %0d, out = %0d", a, $unsigned(b),
out_data);
        if (out_data != 32'd4294967295) begin
            fi = 1'b1;
        end
        sel = 3'b10;
        a = 32'd10;
        b = 32'd2;
        b = -b;
        #1;
        $display("Signed Subtract: a = %0d, b = %0d, out = %0d", a, $signed(b),
out_data);
        if (out_data != 32'd12) begin
            fi = 1'b1;
        end
        sel = 3'b11;
        a = b;
        b = 32'd1;
        #1;
        $display("Unsigned Subtract: a = %0d, b = %0d, out = %0d", a, $unsigned(b),
out_data);
        if (out_data != 32'd4294967293) begin
            fi = 1'b1;
        end
        sel = 3'b100;
        a = 32'd0;
        b = 32'd1;
        #1;
        $display("And: a = %0d, b = %0d, out = %0d", a, b, out_data);
```

```
        if (out_data != 32'd0) begin
            fi = 1'b1;
        end
        sel = 3'b101;
        #1;
        $display("Or: a = %0d, b = %0d, out = %0d", a, b, out_data);
        if (out_data != 32'd1) begin
            fi = 1'b1;
        end
        sel = 3'b110;
        a = 32'd4;
        a = -a;
        b = 32'd1;
        #1;
        $display("Shift Right Arithmetic: a = %0d, b = %0d, out = %0d", $signed(a),
b, out_data);
        if (out_data != 32'd2147483646) begin
            fi = 1'b1;
        end
        sel = 3'b111;
        a = 32'd4;
        b = 32'd1;
        #1;
        $display("Shift Right Logical: a = %0d, b = %0d, out = %0d", $signed(a), b,
out_data);
        if (out_data != 32'd2) begin
            fi = 1'b1;
        end

        if (fi == 1'b0) begin
            $display("Pass");
        end else begin
            $display("Fail");
        end
        $finish();
    end
endmodule
```

```
module hamming (input [21:0] data,  output [1:0] error_code);
    wire [4:0] parity;
    wire [4:0] syndrome;
    wire [4:0] old_parity;
    reg [1:0] ecode;
    assign error_code = ecode;
    assign parity[0] = data[20] ^ data[18] ^ data[16] ^ data[14] ^ data[12] ^
data[10] ^ data[8] ^ data[6] ^ data[4] ^ data[2];
    assign parity[1] = data[18] ^ data[17] ^ data[14] ^ data[13] ^ data[10] ^ data[9]
^ data[6] ^ data[5] ^ data[2];
    assign parity[2] = data[20] ^ data[19] ^ data[14] ^ data[13] ^ data[12] ^
data[11] ^ data[6] ^ data[5] ^ data[4];
    assign parity[3] = data[14] ^ data[13] ^ data[12] ^ data[11] ^ data[10] ^ data[9]
^ data[8];
    assign parity[4] = data[20] ^ data[19] ^ data[18] ^ data[17] ^ data[16];

    assign old_parity[0] = data[0];
    assign old_parity[1] = data[1];
    assign old_parity[2] = data[3];
    assign old_parity[3] = data[7];
    assign old_parity[4] = data[15];

    assign syndrome = old_parity ^ parity;


    /*
    Error detection table
    +----------+-------------+--------------
+-----------------------------------------------+
    | syndrome |   overall   |  error type  |
notes                      |
    |          | Parity (P5) |
|                                               |
    +----------+-------------+--------------
+-----------------------------------------------+
    |    0     |      0      |   no error
|                                               |
    +----------+-------------+--------------
+-----------------------------------------------+
    |   /=0    |      1      | single error | correctable.syndrome holds incorrect
bit      |
    |          |             |              |
position.                                       |
    +----------+-------------+--------------
+-----------------------------------------------+
    |   /=0    |      0      | double error | not
correctable.                                |
    +----------+-------------+--------------
+-----------------------------------------------+
    |    0     |      1      | parity error | overall parity. P5 is in error and can
be     |
    |          |             |              |
corrected.                                      |
    +----------+-------------+--------------
+-----------------------------------------------+
    */

    always @(*) begin
        if (syndrome == 5'b0) begin
            if (data[21] == 1'b0) begin
                ecode <= 2'b00; // no error
            end else begin
```

```
                ecode <= 2'b01; // parity error
            end
        end else begin
            if (data[21] == 1'b0) begin
                ecode <= 2'b10; // double error
            end else begin
                ecode <= 2'b11; // single error
            end
        end
    end
endmodule
```

00 → No Error
11 → Single Error
10 → Double Error
01 → Parity Error


datn[2:0]
data[2:0]
overall parity
generate old parity
data[2:0]
data[2:0]
data [2]
old_parity [4:0]
error code [1:0]
error detection
syndrome [4:0]
generate syndrome
parity [4:0]
generate parity bits

```
`timescale 1ns / 1ns

module hamming_testbench();
    reg [21:0] data;
    wire [1:0] error;
    reg fi = 1'b0;

    hamming tu(
        .data(data),
        .error_code(error)
    );

    /*
    Error detection table
    +----------+-------------+--------------
+-----------------------------------------------+
    | syndrome |   overall   |  error type  |
notes                       |
    |          | Parity (P5) |
|                                               |
    +----------+-------------+--------------
+-----------------------------------------------+
    |    0     |      0      |   no error
|                                               |
    +----------+-------------+--------------
+-----------------------------------------------+
    |   /=0    |      1      | single error | correctable.syndrome holds incorrect
bit      |
    |          |             |              |
position.                                       |
    +----------+-------------+--------------
+-----------------------------------------------+
    |   /=0    |      0      | double error | not
correctable.                                  |
    +----------+-------------+--------------
+-----------------------------------------------+
    |    0     |      1      | parity error | overall parity. P5 is in error and can
be     |
    |          |             |              |
corrected.                                      |
    +----------+-------------+--------------
+-----------------------------------------------+
    */

    initial begin
        $dumpfile("hamming_testbench.vcd");
        $dumpvars(0, hamming_testbench);
        data = 22'b1111111111111111111111;
        #1;
        $display("Error code: %b", error);
        if (error != 2'b11) begin
            fi = 1'b1;
        end
        data = 22'b0100010000001100001110;
        #1;
        $display("Error code: %b", error);
        if (error != 2'b10) begin
            fi = 1'b1;
        end
        data = 22'b0100010000001100001100;
        #1;
        $display("Error code: %b", error);
```

```
        if (error != 2'b00) begin
            fi = 1'b1;
        end
        data = 22'b0110011001100110011001;
        #1;
        $display("Error code: %b", error);
        if (error != 2'b00) begin
            fi = 1'b1;
        end
        data = 22'b1100010000001100001100;
        #1;
        $display("Error code: %b", error);
        if (error != 2'b01) begin
            fi = 1'b1;
        end
        if (fi == 1'b0) begin
            $display("Pass");
        end else begin
            $display("Fail");
        end
        $finish();
    end
endmodule
```

```
module count (input clk, input en, input up, input reset, output [15:0] out_data);
    reg [15:0] out;
    initial out = 16'b0;
    assign out_data = out;
    always @(posedge clk) begin
        if (reset) begin
            out <= 16'b0;
        end else begin
            if (en) begin
                if (up) begin
                    out <= out + 16'b1;
                end else begin
                    out <= out - 16'b1;
                end
            end
        end
    end
endmodule
```

```
`timescale 1ns/1ns

module count_testbench();
    reg clk;
    reg en;
    reg up;
    reg reset;
    wire [15:0] out;
    initial clk = 0;
    initial en = 1'b0;
    initial up = 1'b0;
    initial reset = 1'b0;
    always #(10) clk <= ~clk;

    count tu(
        .clk(clk),
        .en(en),
        .up(up),
        .reset(reset),
        .out_data(out)
    );

    initial begin
        $dumpfile("count_testbench.vcd");
        $dumpvars(0, count_testbench);
        @(posedge clk); #1;
        reset = 1;
        @(posedge clk); #1;
        en = 1;
        up = 1;
        reset = 0;
        repeat (50) @(posedge clk); #1;
        $display("Count: %0d", out);
        @(posedge clk); #1;
        up = 0;
        repeat (26) @(posedge clk); #1;
        $display("Count: %0d", out);
        if (out == 25) begin
            $display("Pass");
        end else begin
            $display("Fail");
        end
        $finish();
    end
endmodule
```

Outputs

```
~/Documents/eecs151/hw/2 vvp alu
VCD info: dumpfile alu_testbench.vcd opened for output.
Signed Add: a = 10, b = -2, out = 8
Unsiged Add: a = 1, b = 4294967294, out = 4294967295
Signed Subtract: a = 10, b = -2, out = 12
Unsigned Subtract: a = 4294967294, b = 1, out = 4294967293
And: a = 0, b = 1, out = 0
Or: a = 0, b = 1, out = 1
Shift Right Arithmetic: a = -4, b = 1, out = 2147483646
Shift Right Logical: a = 4, b = 1, out = 2
Pass
```

```
~/Documents/eecs151/hw/2 vvp hamming
VCD info: dumpfile hamming_testbench.vcd opened for output.
Error code: 11
Error code: 10
Error code: 00
Error code: 00
Error code: 01
Pass
```

```
~/Documents/eecs151/hw/2 vvp count
VCD info: dumpfile count_testbench.vcd opened for output.
Count: 50
Count: 25
Pass
```