ECS 409/609
Assignment 1
Structural (Gate-Level) Assignment Solutions

Kunwar Arpit Singh

Submission Deadline: September 8, 2025

For code repository of this assignment: **Verilog Assignments**

---

1. Problem 1

Problem statement

Implement a structural Verilog model for a 3-input AND gate and a 3-input OR gate.

Verilog source (structural)

3-input AND Gate

```verilog
module and_gate_3_input (input A, input B, input C, output Y);
    wire w1;
    and g1(w1, A, B);
    and g2(Y, w1, C);
endmodule
```

3-input OR Gate

```verilog
module or_gate_3_input (input A, input B, input C, output Y);
    wire w1;
    or g1(w1, A, B);
    or g2(Y, w1, C);
endmodule
```

Testbench

3-input AND Gate

```verilog
module and_gate_3_input_tb;
    reg A, B, C;
    wire Y;
    reg [8*24-1:0] name;

    and_gate_3_input dut (.A(A), .B(B), .C(C), .Y(Y));

    initial begin
```

```
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $dumpfile("assign1_problem1_and_gate_3_input.vcd");
        $dumpvars(1, and_gate_3_input_tb);

        $display("Kunwar␣Arpit␣Singh");
        $display("A␣|␣B␣|␣C␣|␣Y");
        $display("-----------------");
        A = 0; B = 0; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 0; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 1; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 1; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 0; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 0; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 1; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 1; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);

        $finish;
    end
endmodule
```

3-input OR Gate

```
    module or_gate_3_input_tb;
    reg A, B, C;
    wire Y;
    reg [8*24-1:0] name;
    or_gate_3_input dut(.A(A), .B(B), .C(C), .Y(Y));

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem1_or_gate_3_input.vcd");
        $dumpvars(1, or_gate_3_input_tb);

        $display("Kunwar␣Arpit␣Singh");
        $display("A␣|␣B␣|␣C␣|␣Y");
        $display("-----------------");

        A = 0; B = 0; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 0; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 1; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 0; B = 1; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 0; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 0; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 1; C = 0; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);
        A = 1; B = 1; C = 1; #10;
        $display("%b␣|␣%b␣|␣%b␣|␣%b", A, B, C, Y);

        $finish;
    end
endmodule
```

Output in Terminal

3-input AND Gate

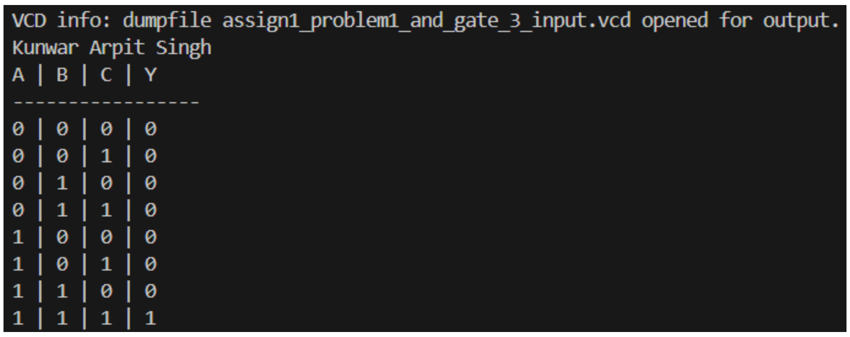
```
VCD info: dumpfile assign1_problem1_and_gate_3_input.vcd opened for output.
Kunwar Arpit Singh
A | B | C | Y
-----------------
0 | 0 | 0 | 0
0 | 0 | 1 | 0
0 | 1 | 0 | 0
0 | 1 | 1 | 0
1 | 0 | 0 | 0
1 | 0 | 1 | 0
1 | 1 | 0 | 0
1 | 1 | 1 | 1
```

3-input OR Gate

```
VCD info: dumpfile assign1_problem1_and_gate_3_input.vcd opened for output.
Kunwar Arpit Singh
A | B | C | Y
-----------------
0 | 0 | 0 | 0
0 | 0 | 1 | 1
0 | 1 | 0 | 1
0 | 1 | 1 | 1
1 | 0 | 0 | 1
1 | 0 | 1 | 1
1 | 1 | 0 | 1
1 | 1 | 1 | 1
```

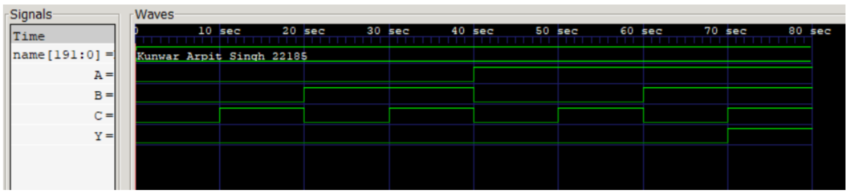
Output Screenshots

Terminal output (3-input AND Gate)

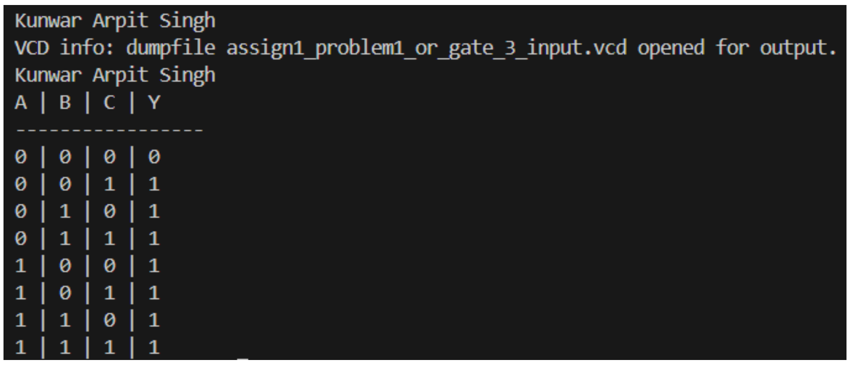


```
VCD info: dumpfile assign1_problem1_and_gate_3_input.vcd opened for output.
Kunwar Arpit Singh
A | B | C | Y
-----------------
0 | 0 | 0 | 0
0 | 0 | 1 | 0
0 | 1 | 0 | 0
0 | 1 | 1 | 0
1 | 0 | 0 | 0
1 | 0 | 1 | 0
1 | 1 | 0 | 0
1 | 1 | 1 | 1
```

GTKWave waveform(3-input AND Gate)

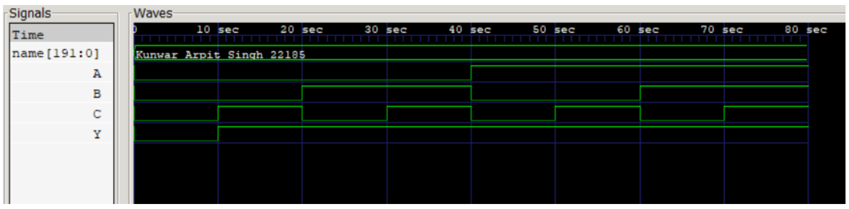


Terminal output (3-input OR Gate)



```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem1_or_gate_3_input.vcd opened for output.
Kunwar Arpit Singh
A | B | C | Y
-----------------
0 | 0 | 0 | 0
0 | 0 | 1 | 1
0 | 1 | 0 | 1
0 | 1 | 1 | 1
1 | 0 | 0 | 1
1 | 0 | 1 | 1
1 | 1 | 0 | 1
1 | 1 | 1 | 1
```

GTKWave waveform (3-input OR Gate)



2. Problem 2

Problem statement

Implement a structural Verilog model for a 4-bit magnitude comparator.

Verilog source (structural)

```verilog
module comparator_1_bit(input A, input B, output eq, output gt, output lt);
    xnor(eq, A, B);
    and(gt, A, ~B);
    and(lt, ~A, B);
endmodule

module comparator_4_bit(input [3:0] A, input [3:0] B, output a_eq_b, output a_gt_b, output a_lt_b);
    wire [3:0] eq, gt, lt;

    comparator_1_bit comp0(A[0], B[0], eq[0], gt[0], lt[0]);
    comparator_1_bit comp1(A[1], B[1], eq[1], gt[1], lt[1]);
    comparator_1_bit comp2(A[2], B[2], eq[2], gt[2], lt[2]);
    comparator_1_bit comp3(A[3], B[3], eq[3], gt[3], lt[3]);

    assign a_eq_b = eq[0] & eq[1] & eq[2] & eq[3];
    assign a_gt_b = gt[0] | (eq[0] & gt[1]) | (eq[0] & eq[1] & gt[2]) | (eq[0] & eq[1] & eq[2] & gt[3]);
    assign a_lt_b = lt[0] | (eq[0] & lt[1]) | (eq[0] & eq[1] & lt[2]) | (eq[0] & eq[1] & eq[2] & lt[3]);
endmodule
```

Testbench

Testbench (All the 256 possible inputs): This testbench shows all the 256 combinations of inputs and only the GTKWave is shown in this report.

```verilog
module comparator_4_bit_tb;
    reg [3:0] A, B;
    wire a_eq_b;
    wire a_gt_b;
    wire a_lt_b;
    integer i, j;
    comparator_4_bit dut (.A(A), .B(B), .a_eq_b(a_eq_b), .a_gt_b(a_gt_b), .a_lt_b(a_lt_b));
    reg [8*24-1:0] name;

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem2_comparator_4_bit.vcd");
        $dumpvars(1, comparator_4_bit_tb);

        $display("Kunwar Arpit Singh");
        $display("A    | B     | A == B | A > B | A < B");
        $display("-----------------------");

        for (i = 0 ; i < 16 ; i = i+1) begin
            for (j = 0 ; j < 16 ; j = j+1) begin
                A = i;
                B = j;
                #10;
                $display("%b | %b |   %b |   %b |   %b", A, B, a_eq_b, a_gt_b,
                    a_lt_b);
            end
        end
    end
endmodule
```

Testbench (Specific outputs (For Clarity))

```
    module comparator_4_bit_tb;
    reg [3:0] A, B;
    wire a_eq_b;
    wire a_gt_b;
    wire a_lt_b;
    integer i, j;
    comparator_4_bit dut (.A(A), .B(B), .a_eq_b(a_eq_b), .a_gt_b(a_gt_b), .
        a_lt_b(a_lt_b));
    reg [8*24-1:0] name;

    initial begin
        name = " Kunwar Arpit Singh 22185";
        $display("Kunwar Arpit Singh");
        $dumpfile("assign1_problem2_comparator_4_bit_small.vcd");
        $dumpvars(1, comparator_4_bit_tb);

        $display("Kunwar Arpit Singh");
        $display("A    | B     | A == B | A > B | A < B");
        $display("-----------------------");

        A = 4'b1010; B = 4'b1010; #10;
        $display("%b | %b | %b | %b | %b", A, B, a_eq_b, a_gt_b, a_lt_b);
        A = 4'b1010; B = 4'b1011; #10;
        $display("%b | %b | %b | %b | %b", A, B, a_eq_b, a_gt_b, a_lt_b);
        A = 4'b0000; B = 4'b1111; #10;
        $display("%b | %b | %b | %b | %b", A, B, a_eq_b, a_gt_b, a_lt_b);
        A = 4'b1111; B = 4'b1111; #10;
        $display("%b | %b | %b | %b | %b", A, B, a_eq_b, a_gt_b, a_lt_b);
        $finish;
    end
endmodule
```

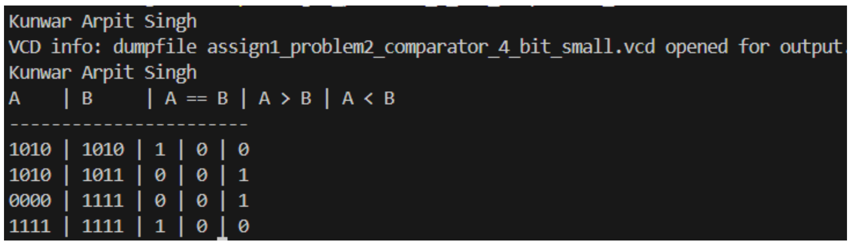
Output in Terminal

Specific outputs (For Clarity)

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem2_comparator_4_bit_small.vcd opened for
    output.
Kunwar Arpit Singh
A | B | A == B | A > B | A < B
-----------------------
1010 | 1010 | 1 | 0 | 0
1010 | 1011 | 0 | 0 | 1
0000 | 1111 | 0 | 0 | 1
1111 | 1111 | 1 | 0 | 0
```

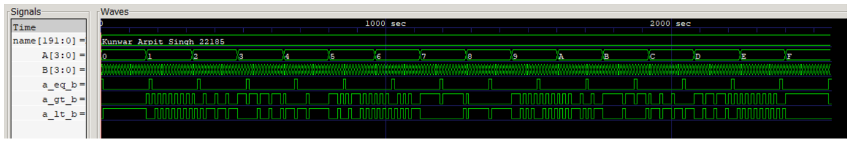
Output Screenshots

Terminal output(Specific outputs (For Clarity))

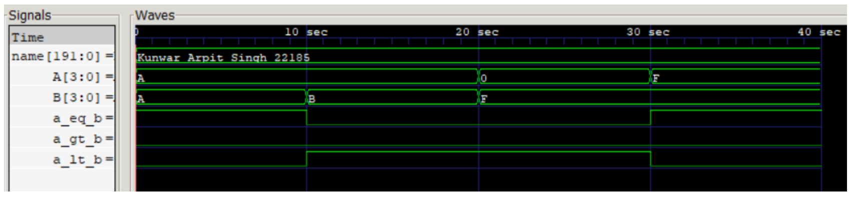


```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem2_comparator_4_bit_small.vcd opened for output
Kunwar Arpit Singh
A    | B    | A == B | A > B | A < B
-----------------------
1010 | 1010 | 1 | 0 | 0
1010 | 1011 | 0 | 0 | 1
0000 | 1111 | 0 | 0 | 1
1111 | 1111 | 1 | 0 | 0
```

GTKWave waveform (All the 256 possible inputs)



GTKWave waveform (Specific outputs (For Clarity))



3. Problem 3

Problem statement

Develop a structural Verilog model for a half adder and subtractor using basic gates and a full adder/subtractor using basic gates.

Verilog source (structural)

Half Adder

```verilog
module half_adder(input A, input B, output SUM, output CARRY);
    xor(SUM, A, B);
    and(CARRY, A, B);
endmodule
```

Full Adder

```verilog
`include "assign1_problem3_half_adder.v"
module full_adder(input A, input B, input CIN, output SUM, output COUT);
    wire SUM1, CARRY1, CARRY2;
    half_adder HA1(A, B, SUM1, CARRY1);
    half_adder HA2(SUM1, CIN, SUM, CARRY2);
    or g1(COUT, CARRY1, CARRY2);
endmodule
```

Half Subtractor

```verilog
module half_subtractor (input A, input B, output diff, output borrow_out);
    xor g1(diff, A, B);
    wire not_A;
    not g2(not_A, A);
    and g3(borrow_out, not_A, B);
endmodule
```

Full Subtractor

```verilog
`include "assign1_problem3_half_subtractor.v"
module full_subtractor (input A, input B, input borrow_in, output diff, output
     borrow_out);
    wire diff1, borrow1, borrow2;
    half_subtractor hs1 (.A(A), .B(B), .diff(diff1), .borrow_out(borrow1));
    half_subtractor hs2 (.A(diff1), .B(borrow_in), .diff(diff), .borrow_out(
        borrow2));
    or g1(borrow_out, borrow1, borrow2);
endmodule
```

Testbench

Half Adder

```verilog
module half_adder_tb;
    reg A, B;
    wire SUM, CARRY;
    reg [8*24-1:0] name;
    half_adder dut (A, B, SUM, CARRY);

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem3_half_adder.vcd");
        $dumpvars(1, half_adder_tb);
        $display("Kunwar␣Arpit␣Singh");
        $display("A␣|␣B␣|␣CARRY␣|␣SUM");
        $display("-------------------");

        A = 0 ; B = 0 ; #10;
        $display("%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣%b", A, B, CARRY, SUM);
        A = 0 ; B = 1 ; #10;
        $display("%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣%b", A, B, CARRY, SUM);
        A = 1 ; B = 0 ; #10;
        $display("%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣%b", A, B, CARRY, SUM);
        A = 1 ; B = 1 ; #10;
        $display("%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣%b", A, B, CARRY, SUM);
        $finish;
    end
endmodule
```

Full Adder

```
module full_adder_tb;
    reg A, B, CIN;
    wire SUM, COUT;
    integer i;
    reg [8*24-1:0] name;
    full_adder dut (A, B, CIN, SUM, COUT);
    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem3_full_adder.vcd");
        $dumpvars(1, full_adder_tb);
        $display("Kunwar␣Arpit␣Singh");
        $display("A␣B␣CIN␣|␣SUM␣COUT");
        $display("-------------------");
        for (i = 0 ; i < 8 ; i = i + 1) begin
            {A, B, CIN} = i;
            #10;
            $display("%b␣%b␣%b␣|␣␣␣%b␣␣␣%b", A, B, CIN, SUM, COUT);
        end
        $finish;
    end
endmodule
```

Half Subtractor

```
module half_subtractor_tb;
    reg A, B;
    wire diff, borrow_out;
    reg [8*24-1:0] name;
    half_subtractor dut(A, B, diff, borrow_out);

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem3_half_subtractor.vcd");
        $dumpvars(1, half_subtractor_tb);
        $display("A␣|␣B␣|␣diff␣|␣borrow_out");
        $display("--------------------");
        A = 0 ; B = 0 ; #10;
        $display("%b␣|␣%b␣|␣␣%b␣␣|␣␣␣␣␣%b", A, B, diff, borrow_out);
        A = 0 ; B = 1 ; #10;
        $display("%b␣|␣%b␣|␣␣%b␣␣|␣␣␣␣␣%b", A, B, diff, borrow_out);
        A = 1 ; B = 0 ; #10;
        $display("%b␣|␣%b␣|␣␣%b␣␣|␣␣␣␣␣%b", A, B, diff, borrow_out);
        A = 1 ; B = 1 ; #10;
        $display("%b␣|␣%b␣|␣␣%b␣␣|␣␣␣␣␣%b", A, B, diff, borrow_out);
        $finish;
    end
endmodule
```

Full Subtractor

```
module full_subtractor_tb;

    reg A, B, borrow_in;
    wire diff, borrow_out;
    integer i;
    reg [8*24-1:0] name;

    full_subtractor dut (.A(A), .B(B), .borrow_in(borrow_in), .diff(diff), .borrow_out(borrow_out));

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem3_full_subtractor.vcd");
        $dumpvars(1, full_subtractor_tb);

        $display("␣A␣|␣B␣|␣borrow_in␣|␣diff␣|␣borrow_out");
        $display("----------------------------");

        for ( i = 0 ; i < 8; i = i + 1) begin
            {A, B, borrow_in} = i;
            #10;
            $display("␣%b␣|␣%b␣|␣␣␣␣%b␣␣␣␣␣|␣␣%b␣␣|␣␣␣␣␣%b", A, B, borrow_in, diff, borrow_out);
        end
        $finish;
    end
```

```
endmodule
```

Output in Terminal

Half Adder

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_half_adder.vcd opened for output.
Kunwar Arpit Singh
A | B | CARRY | SUM
-------------------
0 | 0 | 0 | 0
0 | 1 | 0 | 1
1 | 0 | 0 | 1
1 | 1 | 1 | 0
```

Full Adder

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_full_adder.vcd opened for output.
Kunwar Arpit Singh
A B CIN | SUM COUT
-------------------
0 0 0 | 0 0
0 0 1 | 1 0
0 1 0 | 1 0
0 1 1 | 0 1
1 0 0 | 1 0
1 0 1 | 0 1
1 1 0 | 0 1
1 1 1 | 1 1
```

Half Subtractor

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_half_subtractor.vcd opened for output.
A | B | diff | borrow_out
-------------------
0 | 0 | 0 | 0
0 | 1 | 1 | 1
1 | 0 | 1 | 0
1 | 1 | 0 | 0
```

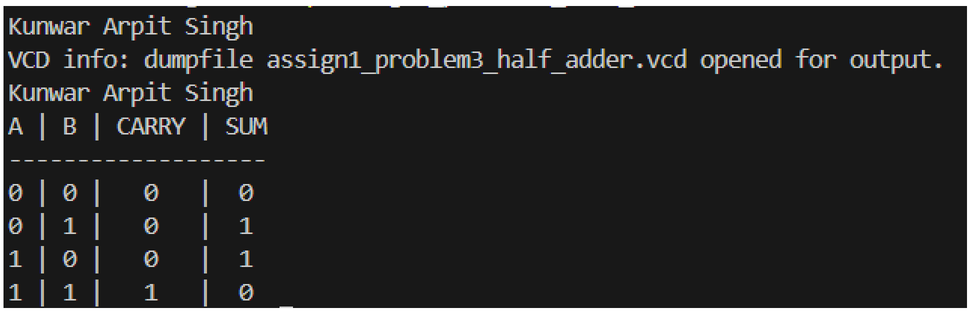
Full Subtractor

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_full_subtractor.vcd opened for output.
 A | B | borrow_in | diff | borrow_out
---------------------------
 0 | 0 | 0 | 0 | 0
 0 | 0 | 1 | 1 | 1
 0 | 1 | 0 | 1 | 1
 0 | 1 | 1 | 0 | 1
```

```
1 | 0 | 0 | 1 | 0
1 | 0 | 1 | 0 | 0
1 | 1 | 0 | 0 | 0
1 | 1 | 1 | 1 | 1
```

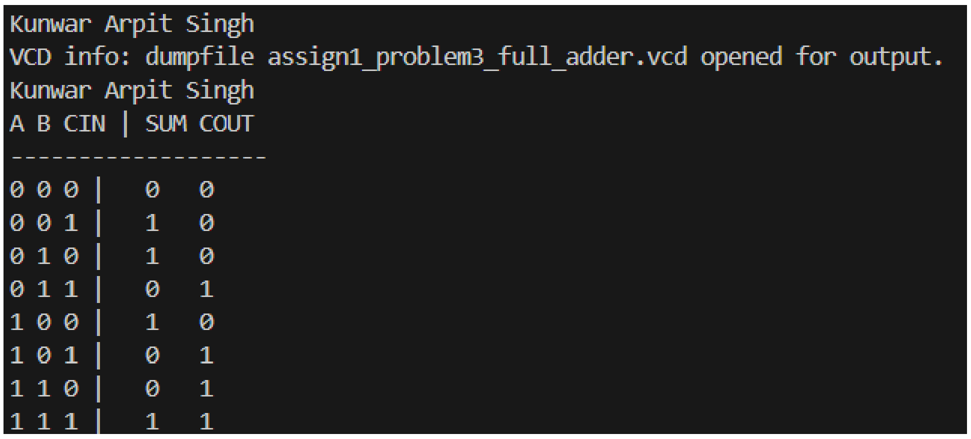
Output Screenshots

Terminal output (Half Adder)



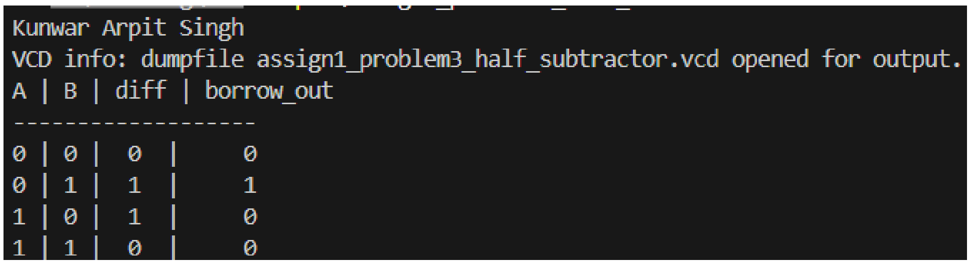
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_half_adder.vcd opened for output.
Kunwar Arpit Singh
A | B | CARRY | SUM
-------------------
0 | 0 |   0   |  0
0 | 1 |   0   |  1
1 | 0 |   0   |  1
1 | 1 |   1   |  0
```

Terminal output (Full Adder)



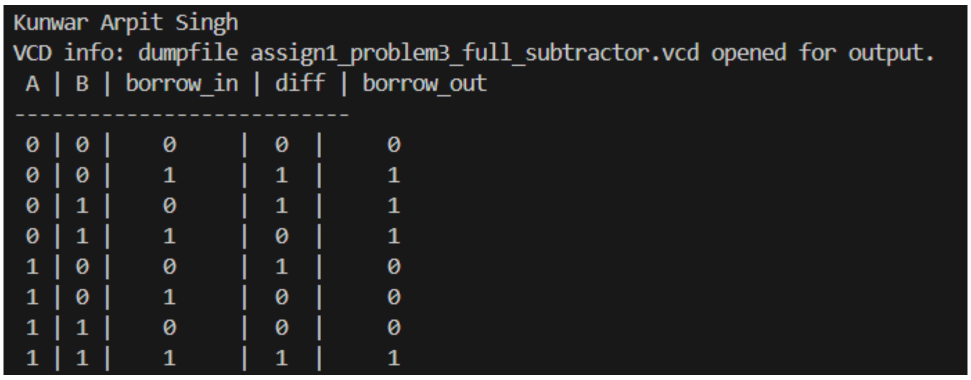
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_full_adder.vcd opened for output.
Kunwar Arpit Singh
A B CIN | SUM COUT
-------------------
0 0 0 |   0   0
0 0 1 |   1   0
0 1 0 |   1   0
0 1 1 |   0   1
1 0 0 |   1   0
1 0 1 |   0   1
1 1 0 |   0   1
1 1 1 |   1   1
```

Terminal output (Half Subtractor)



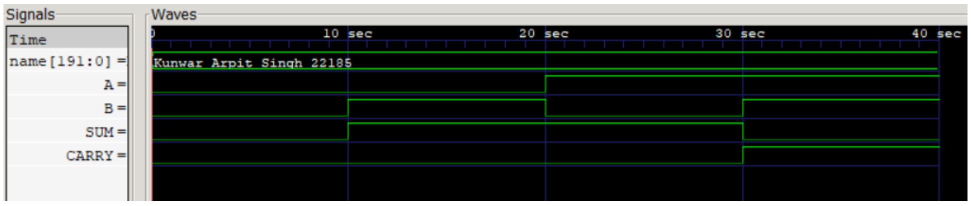
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_half_subtractor.vcd opened for output.
A | B | diff | borrow_out
-------------------
0 | 0 |  0  |     0
0 | 1 |  1  |     1
1 | 0 |  1  |     0
1 | 1 |  0  |     0
```

Terminal output (Full Subtractor)

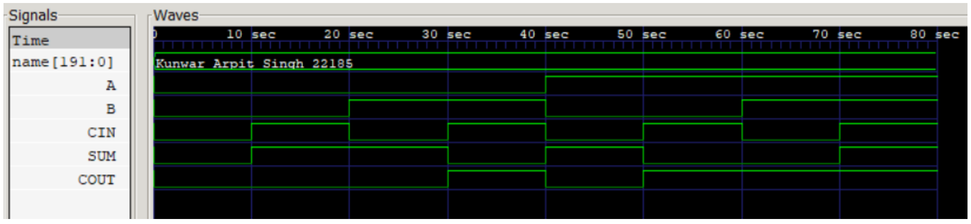


```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem3_full_subtractor.vcd opened for output.
 A | B | borrow_in | diff | borrow_out
---------------------------
 0 | 0 |    0     |  0  |     0
 0 | 0 |    1     |  1  |     1
 0 | 1 |    0     |  1  |     1
 0 | 1 |    1     |  0  |     1
 1 | 0 |    0     |  1  |     0
 1 | 0 |    1     |  0  |     0
 1 | 1 |    0     |  0  |     0
 1 | 1 |    1     |  1  |     1
```

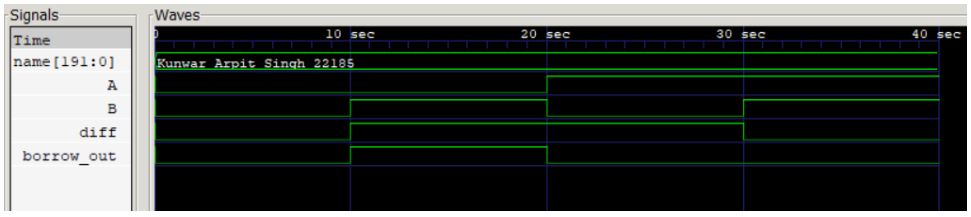
GTKWave waveform (Half Adder)



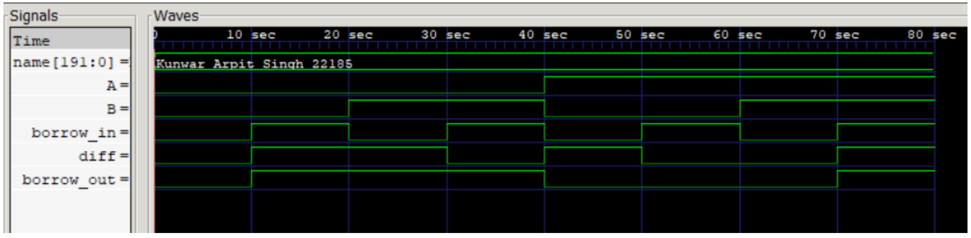
GTKWave waveform (Full Adder)



GTKWave waveform (Half Subtractor)



GTKWave waveform (Full Subtractor)



4. Problem 4

Problem statement

Implement a structural Verilog model for a 4-to-1 multiplexer and 3-to-8 decoder.

Verilog source (structural)

4-to-1 multiplexer

```
module mux_4_to_1(input [3:0] d_in, input[1:0] s, output y);
    wire s0_not, s1_not;
    wire w0, w1, w2, w3;

    not(s0_not, s[0]);
    not(s1_not, s[1]);

    and(w0, d_in[0], s0_not, s1_not);
    and(w1, d_in[1], s[0], s1_not);
    and(w2, d_in[2], s0_not, s[1]);
    and(w3, d_in[3], s[0], s[1]);

    or(y, w0, w1, w2, w3);
endmodule
```

3-to-8 decoder

```
module decoder_3_to_8(input [2:0] d, output [7:0] y);

    wire d0_not, d1_not, d2_not;

    not (d0_not, d[0]);
    not (d1_not, d[1]);
    not (d2_not, d[2]);

    and (y[0], d0_not, d1_not, d2_not);
    and (y[1], d[0], d1_not, d2_not);
    and (y[2], d0_not, d[1], d2_not);
    and (y[3], d[0], d[1], d2_not);
    and (y[4], d0_not, d1_not, d[2]);
    and (y[5], d[0], d1_not, d[2]);
    and (y[6], d0_not, d[1], d[2]);
    and (y[7], d[0], d[1], d[2]);

endmodule
```

Testbench

4-to-1 multiplexer

```
module mux_4_to_1_tb;
    reg [3:0] d_in;
    reg [1:0] s;
    wire y;
    reg [8*24-1:0] name;

    mux_4_to_1 dut(.d_in(d_in), .s(s), .y(y));

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem4_mux_4_to_1.vcd");
        $dumpvars(1, mux_4_to_1_tb);

        d_in = 4'b1010;
        $display("s1␣|␣s0␣|␣y␣(selected␣data)");
        $display("-----------------------");
        s = 2'b00; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d0)", s[1], s[0], y);
        s = 2'b01; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d1)", s[1], s[0], y);
        s = 2'b10; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d2)", s[1], s[0], y);
        s = 2'b11; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d3)", s[1], s[0], y);

        d_in = 4'b0101;
        $display("s1␣|␣s0␣|␣y␣(selected␣data)");
        $display("-----------------------");
        s = 2'b00; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d0)", s[1], s[0], y);
        s = 2'b01; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d1)", s[1], s[0], y);
        s = 2'b10; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d2)", s[1], s[0], y);
        s = 2'b11; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d3)", s[1], s[0], y);

        d_in = 4'b1111;
        $display("s1␣|␣s0␣|␣y␣(selected␣data)");
        $display("-----------------------");
        s = 2'b00; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d0)", s[1], s[0], y);
        s = 2'b01; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d1)", s[1], s[0], y);
        s = 2'b10; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d2)", s[1], s[0], y);
        s = 2'b11; #10;
        $display("␣%b␣|␣%b␣|␣␣%b␣(d3)", s[1], s[0], y);

        $finish;
    end
endmodule
```

3-to-8 decoder (Using Registers)

```
module decoder_3_to_8_tb;
    reg[2:0] d;
    wire[7:0] y;
    reg [8*24-1:0] name;

    decoder_3_to_8 dut (d, y);

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem4_3_to_8_decoder.vcd");
        $dumpvars(1, decoder_3_to_8_tb);

        $display("d2␣|␣d1␣|␣d0␣||␣y7␣|␣y6␣|␣y5␣|␣y4␣|␣y3␣|␣y2␣|␣y1␣|␣y0");
        $display("-------------------------------");
        d = 3’b000; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b001; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b010; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b011; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b100; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b101; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b110; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
        d = 3’b111; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d
            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
    end
endmodule
```

3-to-8 decoder (Using 1-bit at a time)

```
module decoder_3_to_8_tb;
    reg d0, d1, d2;
    wire y0, y1, y2, y3, y4, y5, y6, y7;
    reg [8*24-1:0] name;
    decoder_3_to_8 dut (.d({d2, d1, d0}), .y({y7, y6, y5, y4, y3, y2, y1, y0}))
        ;

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem4_3_to_8_decoder.vcd");
        $dumpvars(1, decoder_3_to_8_tb);

        $display("d2␣|␣d1␣|␣d0␣||␣y7␣|␣y6␣|␣y5␣|␣y4␣|␣y3␣|␣y2␣|␣y1␣|␣y0");
        $display("-------------------------------");
        d0 = 1'b0; d1 = 1'b0; d2 = 1'b0; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b1; d1 = 1'b0; d2 = 1'b0; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b0; d1 = 1'b1; d2 = 1'b0; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b1; d1 = 1'b1; d2 = 1'b0; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b0; d1 = 1'b0; d2 = 1'b1; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b1; d1 = 1'b0; d2 = 1'b1; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b0; d1 = 1'b1; d2 = 1'b1; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
        d0 = 1'b1; d1 = 1'b1; d2 = 1'b1; #10;
        $display("%b␣␣|␣%b␣␣|␣%b␣␣||␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b␣|␣%b", d2
            , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
    end
endmodule
```

Output in Terminal

4-to-1 multiplexer

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_mux_4_to_1.vcd opened for output.
s1 | s0 | y (selected data)
-----------------------
 0 | 0 | 0 (d0)
 0 | 1 | 1 (d1)
 1 | 0 | 0 (d2)
 1 | 1 | 1 (d3)
s1 | s0 | y (selected data)
-----------------------
 0 | 0 | 1 (d0)
 0 | 1 | 0 (d1)
 1 | 0 | 1 (d2)
 1 | 1 | 0 (d3)
s1 | s0 | y (selected data)
-----------------------
 0 | 0 | 1 (d0)
 0 | 1 | 1 (d1)
 1 | 0 | 1 (d2)
 1 | 1 | 1 (d3)
```

3-to-8 decoder (Using Registers)

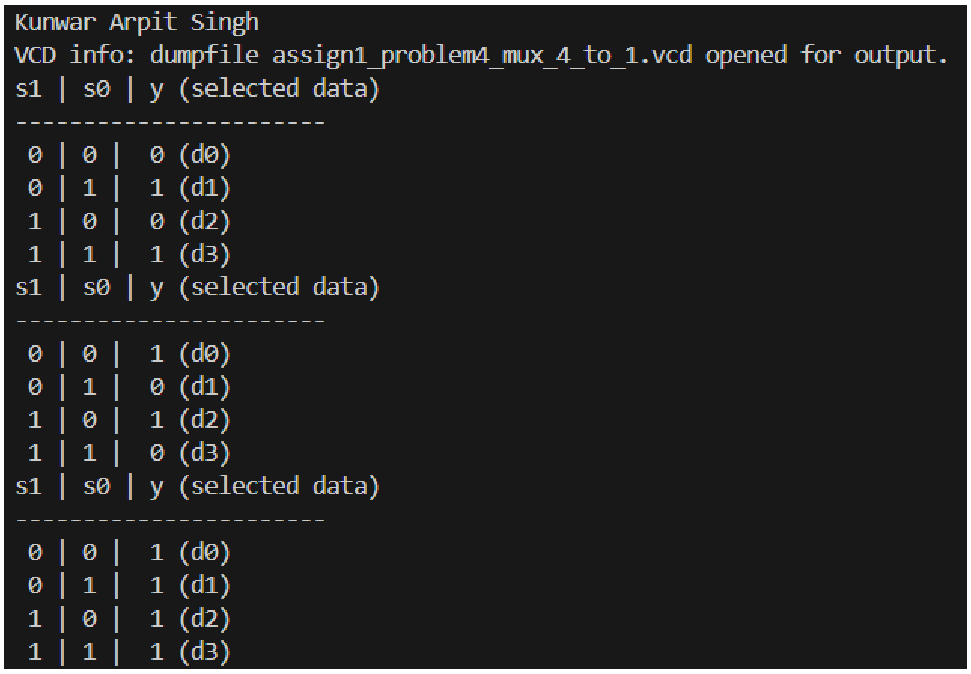
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for output.
d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
-------------------------------
0 | 0 | 0 || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
0 | 0 | 1 || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
0 | 1 | 0 || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
0 | 1 | 1 || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
1 | 0 | 0 || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
1 | 0 | 1 || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
1 | 1 | 0 || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
1 | 1 | 1 || 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0
```

3-to-8 decoder (Using 1-bit at a time)

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for output.
d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
-------------------------------
0 | 0 | 0 || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
0 | 0 | 1 || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
0 | 1 | 0 || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
0 | 1 | 1 || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
1 | 0 | 0 || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
1 | 0 | 1 || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
1 | 1 | 0 || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
1 | 1 | 1 || 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0
```

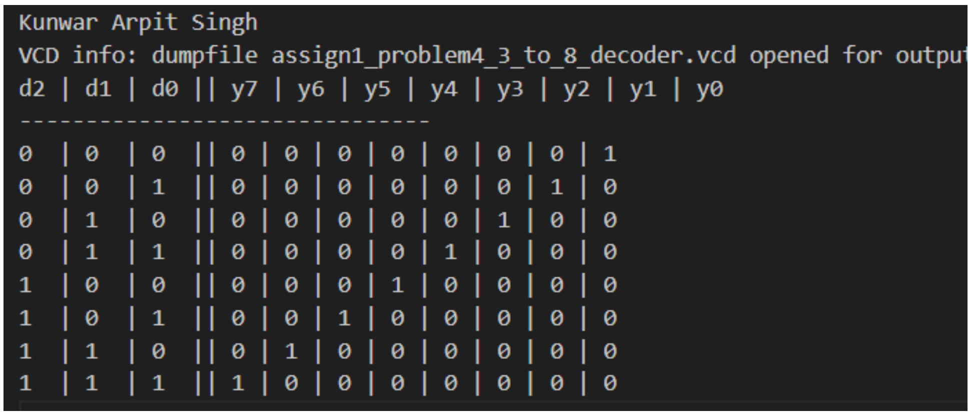
Output Screenshots

Terminal output(4-to-1 multiplexer)



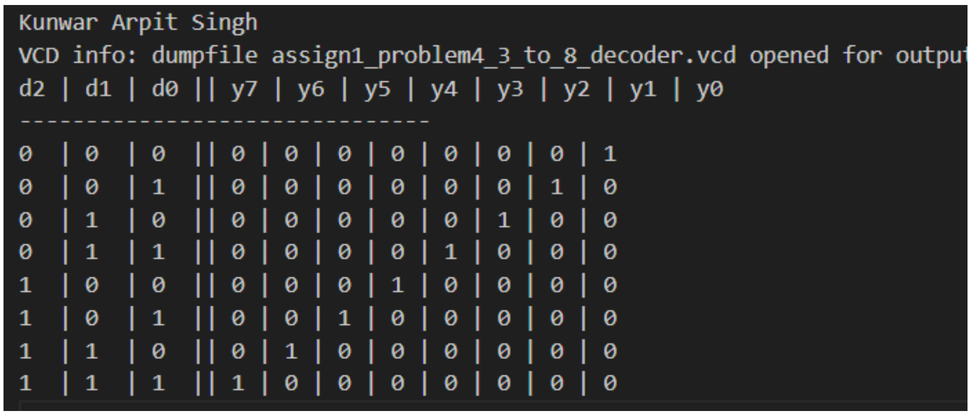
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_mux_4_to_1.vcd opened for output.
s1 | s0 | y (selected data)
----------------------
 0 | 0 |  0 (d0)
 0 | 1 |  1 (d1)
 1 | 0 |  0 (d2)
 1 | 1 |  1 (d3)
s1 | s0 | y (selected data)
----------------------
 0 | 0 |  1 (d0)
 0 | 1 |  0 (d1)
 1 | 0 |  1 (d2)
 1 | 1 |  0 (d3)
s1 | s0 | y (selected data)
----------------------
 0 | 0 |  1 (d0)
 0 | 1 |  1 (d1)
 1 | 0 |  1 (d2)
 1 | 1 |  1 (d3)
```

Terminal output(3-to-8 decoder (Using Registers))



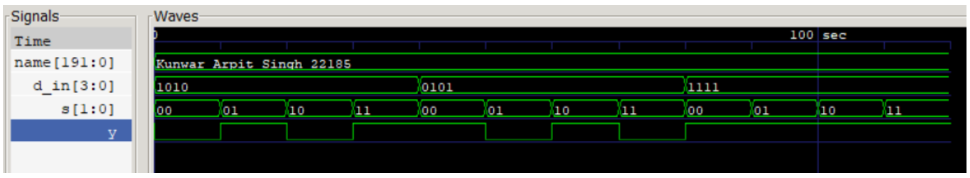
```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for output
d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
------------------------------
0  | 0  | 0  || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
0  | 0  | 1  || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
0  | 1  | 0  || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
0  | 1  | 1  || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
1  | 0  | 0  || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
1  | 0  | 1  || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
1  | 1  | 0  || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
1  | 1  | 1  || 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0
```

Terminal output(3-to-8 decoder (Using 1-bit at a time))

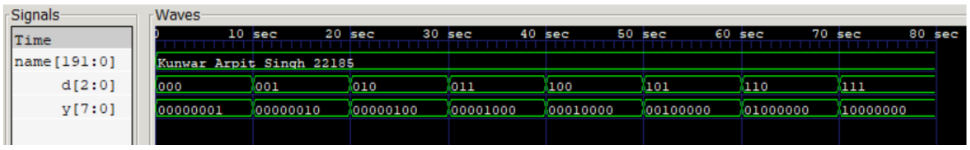


```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for outpu
d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
------------------------------
0  | 0  | 0  || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
0  | 0  | 1  || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
0  | 1  | 0  || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
0  | 1  | 1  || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
1  | 0  | 0  || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
1  | 0  | 1  || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
1  | 1  | 0  || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
1  | 1  | 1  || 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0
```

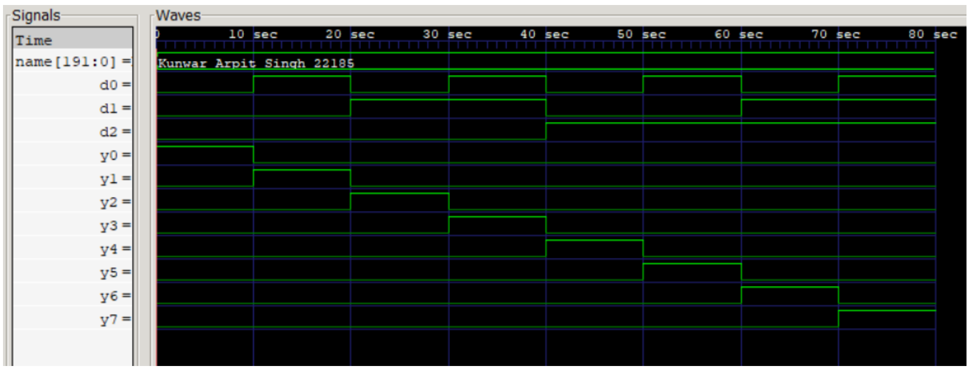
Terminal output(4-to-1 multiplexer)



GTKWave waveform (3-to-8 decoder (Using Registers))



GTKWave waveform (3-to-8 decoder (Using 1-bit at a time))



5. Problem 5

Problem statement

Create a structural Verilog model for a 4-bit barrel shifter with (1-bit and 2-bit) left and right shift.

Verilog source (structural)

```verilog
module barrel_shifter_4bit (input [3:0] A, input [1:0] sel, input dir, output
    [3:0] out);
    wire [3:0] left1, left2;
    wire [3:0] right1, right2;
    wire [3:0] sel_left, sel_right;

    assign left1 = {A[2:0], 1'b0};
    assign left2 = {A[1:0], 2'b00};

    assign sel_left = (sel == 2'b00) ? A :(sel == 2'b01) ? left1 : left2;

    assign right1 = {1'b0, A[3:1]};
    assign right2 = {2'b00, A[3:2]};

    assign sel_right = (sel == 2'b00) ? A :(sel == 2'b01) ? right1 : right2;

    assign out = (dir == 0) ? sel_left : sel_right;
endmodule
```

Testbench

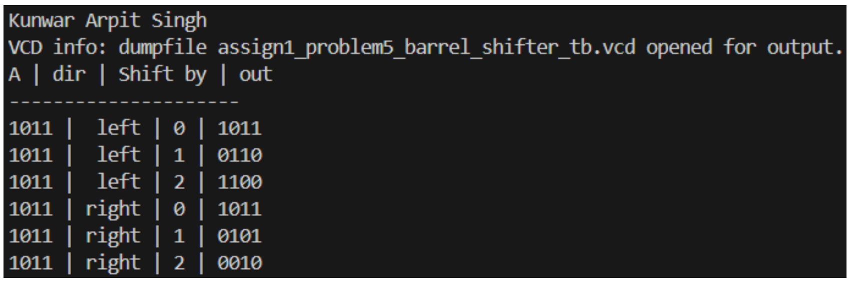
```verilog
module barrel_shifter_4bit_tb;
    reg [3:0] A;
    reg [1:0] sel;
    reg dir;
    wire [3:0] out;
    reg [8*24-1:0] name;

    barrel_shifter_4bit dut (.A(A), .sel(sel), .dir(dir), .out(out));

    initial begin
        name = " Kunwar Arpit Singh 22185";

        $display("Kunwar Arpit Singh");
        $dumpfile("assign1_problem5_barrel_shifter_tb.vcd");
        $dumpvars(1, barrel_shifter_4bit_tb);

        $display("A | dir | Shift by | out");
        $display("---------------------");

        A = 4'b1011; dir = 0; sel = 2'b00; #10;
        $display("%b␣|␣left␣␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        A = 4'b1011; dir = 0; sel = 2'b01; #10;
        $display("%b␣|␣left␣␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        A = 4'b1011; dir = 0; sel = 2'b10; #10;
        $display("%b␣|␣left␣␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        A = 4'b1011; dir = 1; sel = 2'b00; #10;
        $display("%b␣|␣right␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        A = 4'b1011; dir = 1; sel = 2'b01; #10;
        $display("%b␣|␣right␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        A = 4'b1011; dir = 1; sel = 2'b10; #10;
        $display("%b␣|␣right␣|␣%0d␣␣␣␣|␣%b", A, sel, out);
        $finish;
    end
endmodule
```

Output in Terminal

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem5_barrel_shifter_tb.vcd opened for output.
A | dir | Shift by | out
----------------------
1011 | left | 0 | 1011
1011 | left | 1 | 0110
1011 | left | 2 | 1100
1011 | right | 0 | 1011
1011 | right | 1 | 0101
1011 | right | 2 | 0010
```

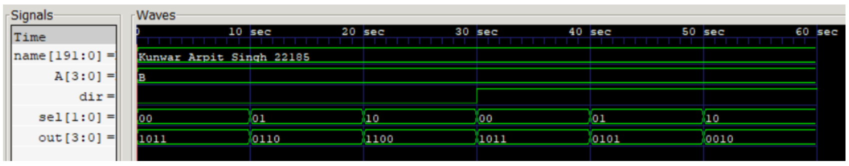
Output Screenshots

Terminal output (Barrel Shifter)



```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem5_barrel_shifter_tb.vcd opened for output.
A | dir | Shift by | out
----------------------
1011 |  left | 0 | 1011
1011 |  left | 1 | 0110
1011 |  left | 2 | 1100
1011 | right | 0 | 1011
1011 | right | 1 | 0101
1011 | right | 2 | 0010
```

GTKWave waveform (Barrel Shifter)



6. Problem 6

Problem statement

Create a structural Verilog model for a simple-bit ALU that supports addition, subtraction, multiplication, logical AND, logical OR, and logical XOR. Use Op-Code to specify the operation.

Verilog source (structural)

```verilog
module half_adder(input A, input B, output SUM, output CARRY);

    xor(SUM, A, B);
    and(CARRY, A, B);
endmodule

module full_adder(input A, input B, input CIN, output SUM, output COUT);
    wire SUM1, CARRY1, CARRY2;

    half_adder HA1(A, B, SUM1, CARRY1);

    half_adder HA2(SUM1, CIN, SUM, CARRY2);

    or g1(COUT, CARRY1, CARRY2);
endmodule

module mux_8_to_1_bit (input d0, d1, d2, d3, d4, d5, d6, d7, input [2:0] sel,
    output out);
    wire s0not, s1not, s2not;
    wire w0, w1, w2, w3, w4, w5, w6, w7;
    not (s0not, sel[0]);
    not (s1not, sel[1]);
    not (s2not, sel[2]);

    and(w0, d0, s2not, s1not, s0not);
    and(w1, d1, s2not, s1not, sel[0]);
    and(w2, d2, s2not, sel[1], s0not);
    and(w3, d3, s2not, sel[1], sel[0]);
    and(w4, d4, sel[2], s1not, s0not);
    and(w5, d5, sel[2], s1not, sel[0]);
    and(w6, d6, sel[2], sel[1], s0not);
    and(w7, d7, sel[2], sel[1], sel[0]);

    or(out, w0, w1, w2, w3, w4, w5, w6, w7);

endmodule

module four_bit_adder(input [3:0] A, input [3:0] B, input CARRYIN, output
    [3:0] SUM, output CARRYOUT);

    wire C1, C2, C3;

    full_adder FA0(A[0], B[0], CARRYIN, SUM[0], C1);
    full_adder FA1(A[1], B[1], C1, SUM[1], C2);
    full_adder FA2(A[2], B[2], C2, SUM[2], C3);
    full_adder FA3(A[3], B[3], C3, SUM[3], CARRYOUT);
endmodule

module four_bit_subtractor (input [3:0] A, input [3:0] B, output [3:0]
    DIFFERENCE);
    wire [3:0] B_COMPLEMENT;
    wire CARRYOUT;
    not (B_COMPLEMENT[0], B[0]);
    not (B_COMPLEMENT[1], B[1]);
    not (B_COMPLEMENT[2], B[2]);
    not (B_COMPLEMENT[3], B[3]);

    four_bit_adder SUBTRACTOR_ADDER (A, B_COMPLEMENT, 1'b1, DIFFERENCE,
        CARRYOUT);
endmodule

module four_by_four_multiplier (
    input [3:0] A,
    input [3:0] B,
    output [7:0] PRODUCT
);
    wire [3:0] PP0, PP1, PP2, PP3;

    and (PP0[0], A[0], B[0]);
    and (PP0[1], A[1], B[0]);
    and (PP0[2], A[2], B[0]);
    and (PP0[3], A[3], B[0]);

    and (PP1[0], A[0], B[1]);
    and (PP1[1], A[1], B[1]);
    and (PP1[2], A[2], B[1]);
    and (PP1[3], A[3], B[1]);

    and (PP2[0], A[0], B[2]);
    and (PP2[1], A[1], B[2]);
    and (PP2[2], A[2], B[2]);
    and (PP2[3], A[3], B[2]);

    and (PP3[0], A[0], B[3]);
    and (PP3[1], A[1], B[3]);
    and (PP3[2], A[2], B[3]);
    and (PP3[3], A[3], B[3]);

    wire c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11;
    wire s1, s2, s3, s4, s5, s6;

    assign PRODUCT[0] = PP0[0];

    half_adder HA1 (PP0[1], PP1[0], PRODUCT[1], c1);

    full_adder FA1 (PP0[2], PP1[1], c1, s1, c2);
    half_adder HA2 (s1, PP2[0], PRODUCT[2], c3);

    full_adder FA2 (PP0[3], PP1[2], c2, s2, c4);
    full_adder FA3 (s2, PP2[1], c3, PRODUCT[3], c5);

    full_adder FA4 (PP1[3], PP2[2], c4, s3, c6);
    full_adder FA5 (s3, PP3[0], c5, PRODUCT[4], c7);

    full_adder FA6 (PP2[3], PP3[1], c6, s4, c8);
    full_adder FA7 (s4, c7, 1'b0, PRODUCT[5], c9);

    full_adder FA8 (PP3[2], c8, c9, PRODUCT[6], c10);

    full_adder FA9 (PP3[3], c10, 1'b0, PRODUCT[7], c11);

endmodule


module four_bit_and (input [3:0] A, input [3:0] B, output [3:0] out);
    and(out[0], A[0], B[0]);
    and(out[1], A[1], B[1]);
    and(out[2], A[2], B[2]);
    and(out[3], A[3], B[3]);
endmodule

module four_bit_or(input [3:0] A, input [3:0] B, output [3:0] out);
    or(out[0], A[0], B[0]);
    or(out[1], A[1], B[1]);
    or(out[2], A[2], B[2]);
    or(out[3], A[3], B[3]);
endmodule

module four_bit_xor(input [3:0] A, input [3:0] B, output [3:0] out);
    xor(out[0], A[0], B[0]);
    xor(out[1], A[1], B[1]);
    xor(out[2], A[2], B[2]);
    xor(out[3], A[3], B[3]);
endmodule

module simple_alu_4_bit (input [3:0] A, input [3:0] B, input [2:0] OpCode,
    output [7:0] out);
    wire [7:0] ADDER_OUT;
    wire [7:0] SUBTRACTOR_OUT;
    wire [7:0] MULTIPLIER_OUT;
    wire [7:0] AND_OUT;
    wire [7:0] OR_OUT;
    wire [7:0] XOR_OUT;
    wire CARRYOUT;

    four_bit_adder adder_unit(A, B, 1'b0, ADDER_OUT[3:0], CARRYOUT);
    four_bit_subtractor subtractor_unit(A, B, SUBTRACTOR_OUT[3:0]);
    four_by_four_multiplier multiplier_unit(A, B, MULTIPLIER_OUT);
    four_bit_and and_unit(A, B, AND_OUT[3:0]);
    four_bit_or or_unit(A, B, OR_OUT[3:0]);
    four_bit_xor xor_unit(A, B, XOR_OUT[3:0]);

    assign ADDER_OUT[7:4] = {3'b000, CARRYOUT};
    assign SUBTRACTOR_OUT[7:4] = 4'b0;
    assign AND_OUT[7:4] = 4'b0;
    assign OR_OUT[7:4] = 4'b0;
    assign XOR_OUT[7:4] = 4'b0;

    mux_8_to_1_bit MUX0 (ADDER_OUT[0], SUBTRACTOR_OUT[0], MULTIPLIER_OUT[0],
        AND_OUT[0], OR_OUT[0], XOR_OUT[0], 1'b0, 1'b0, OpCode, out[0]);
    mux_8_to_1_bit MUX1 (ADDER_OUT[1], SUBTRACTOR_OUT[1], MULTIPLIER_OUT[1],
        AND_OUT[1], OR_OUT[1], XOR_OUT[1], 1'b0, 1'b0, OpCode, out[1]);
    mux_8_to_1_bit MUX2 (ADDER_OUT[2], SUBTRACTOR_OUT[2], MULTIPLIER_OUT[2],
        AND_OUT[2], OR_OUT[2], XOR_OUT[2], 1'b0, 1'b0, OpCode, out[2]);
    mux_8_to_1_bit MUX3 (ADDER_OUT[3], SUBTRACTOR_OUT[3], MULTIPLIER_OUT[3],
        AND_OUT[3], OR_OUT[3], XOR_OUT[3], 1'b0, 1'b0, OpCode, out[3]);
    mux_8_to_1_bit MUX4 (ADDER_OUT[4], SUBTRACTOR_OUT[4], MULTIPLIER_OUT[4],
        AND_OUT[4], OR_OUT[4], XOR_OUT[4], 1'b0, 1'b0, OpCode, out[4]);
    mux_8_to_1_bit MUX5 (ADDER_OUT[5], SUBTRACTOR_OUT[5], MULTIPLIER_OUT[5],
        AND_OUT[5], OR_OUT[5], XOR_OUT[5], 1'b0, 1'b0, OpCode, out[5]);
    mux_8_to_1_bit MUX6 (ADDER_OUT[6], SUBTRACTOR_OUT[6], MULTIPLIER_OUT[6],
        AND_OUT[6], OR_OUT[6], XOR_OUT[6], 1'b0, 1'b0, OpCode, out[6]);
    mux_8_to_1_bit MUX7 (ADDER_OUT[7], SUBTRACTOR_OUT[7], MULTIPLIER_OUT[7],
        AND_OUT[7], OR_OUT[7], XOR_OUT[7], 1'b0, 1'b0, OpCode, out[7]);
endmodule
```

Testbench

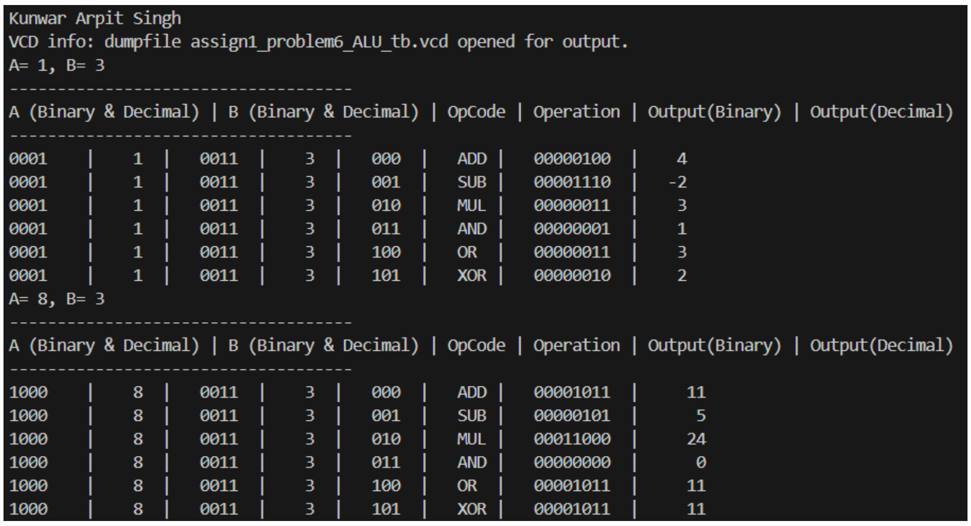
```
module simple_alu_4_bit_tb;
    reg [3:0] A;
    reg [3:0] B;
    reg [2:0] OpCode;
    wire signed [7:0] out;
    reg [8*24-1:0] name;

    simple_alu_4_bit dut (.A(A), .B(B), .OpCode(OpCode), .out(out));

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign1_problem6_ALU_tb.vcd");
        $dumpvars(1, simple_alu_4_bit_tb);

        A = 4'b0001;
        B = 4'b0011;

        $display("A=%d,␣B=%d", A, B);
        $display("------------------------------------");
        $display("A␣(Binary␣&␣Decimal)␣|␣B␣(Binary␣&␣Decimal)␣|␣OpCode␣|␣
            Operation␣|␣Output(Binary)␣|␣Output(Decimal)");
        $display("------------------------------------");

        OpCode = 3'b000; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣ADD␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));
        OpCode = 3'b001; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣SUB␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));
        OpCode = 3'b010; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣MUL␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));
        OpCode = 3'b011; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣AND␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));
        OpCode = 3'b100; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣OR␣␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));
        OpCode = 3'b101; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣XOR␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out[3:0]));


        A = 4'b1000;
        B = 4'b0011;

        $display("A=%d,␣B=%d", A, B);
        $display("------------------------------------");
        $display("A␣(Binary␣&␣Decimal)␣|␣B␣(Binary␣&␣Decimal)␣|␣OpCode␣|␣
            Operation␣|␣Output(Binary)␣|␣Output(Decimal)");
        $display("------------------------------------");

        OpCode = 3'b000; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣ADD␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        OpCode = 3'b001; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣SUB␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        OpCode = 3'b010; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣MUL␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        OpCode = 3'b011; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣AND␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        OpCode = 3'b100; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣OR␣␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        OpCode = 3'b101; #10;
        $display("%b␣␣␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣%d␣␣|␣␣␣%b␣␣|␣␣␣XOR␣|␣␣␣%b␣␣|␣␣␣%d"
            , A, A, B, B, OpCode, out, $signed(out));
        $finish;
    end
endmodule
```

Output in Terminal

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem6_ALU_tb.vcd opened for output.
A= 1, B= 3
------------------------------------
A (Binary & Decimal) | B (Binary & Decimal) | OpCode | Operation | Output(
    Binary) | Output(Decimal)
------------------------------------
0001 | 1 | 0011 | 3 | 000 | ADD | 00000100 | 4
0001 | 1 | 0011 | 3 | 001 | SUB | 00001110 | -2
0001 | 1 | 0011 | 3 | 010 | MUL | 00000011 | 3
0001 | 1 | 0011 | 3 | 011 | AND | 00000001 | 1
0001 | 1 | 0011 | 3 | 100 | OR | 00000011 | 3
0001 | 1 | 0011 | 3 | 101 | XOR | 00000010 | 2
A= 8, B= 3
------------------------------------
A (Binary & Decimal) | B (Binary & Decimal) | OpCode | Operation | Output(
    Binary) | Output(Decimal)
------------------------------------
1000 | 8 | 0011 | 3 | 000 | ADD | 00001011 | 11
1000 | 8 | 0011 | 3 | 001 | SUB | 00000101 | 5
1000 | 8 | 0011 | 3 | 010 | MUL | 00011000 | 24
1000 | 8 | 0011 | 3 | 011 | AND | 00000000 | 0
1000 | 8 | 0011 | 3 | 100 | OR | 00001011 | 11
1000 | 8 | 0011 | 3 | 101 | XOR | 00001011 | 11
```

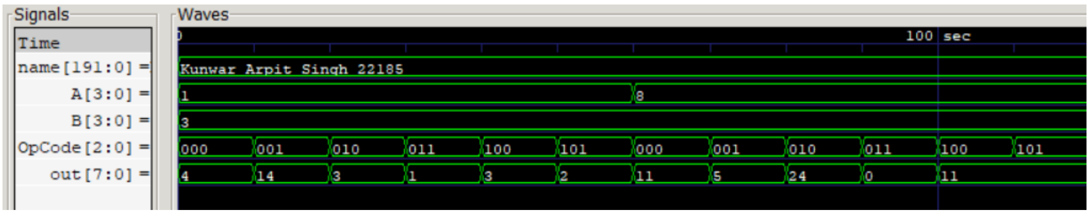
Output Screenshots

Terminal output (Simple-bit ALU)



```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem6_ALU_tb.vcd opened for output.
A= 1, B= 3
------------------------------------
A (Binary & Decimal) | B (Binary & Decimal) | OpCode | Operation | Output(Binary) | Output(Decimal)
------------------------------------
0001    |     1  |   0011   |     3  |   000   |   ADD |   00000100   |    4
0001    |     1  |   0011   |     3  |   001   |   SUB |   00001110   |   -2
0001    |     1  |   0011   |     3  |   010   |   MUL |   00000011   |    3
0001    |     1  |   0011   |     3  |   011   |   AND |   00000001   |    1
0001    |     1  |   0011   |     3  |   100   |   OR  |   00000011   |    3
0001    |     1  |   0011   |     3  |   101   |   XOR |   00000010   |    2
A= 8, B= 3
------------------------------------
A (Binary & Decimal) | B (Binary & Decimal) | OpCode | Operation | Output(Binary) | Output(Decimal)
------------------------------------
1000    |     8  |   0011   |     3  |   000   |   ADD |   00001011   |     11
1000    |     8  |   0011   |     3  |   001   |   SUB |   00000101   |      5
1000    |     8  |   0011   |     3  |   010   |   MUL |   00011000   |     24
1000    |     8  |   0011   |     3  |   011   |   AND |   00000000   |      0
1000    |     8  |   0011   |     3  |   100   |   OR  |   00001011   |     11
1000    |     8  |   0011   |     3  |   101   |   XOR |   00001011   |     11
```

GTKWave waveform (Simple-bit ALU)


Signals
Waves
Time
0
100 sec
name[191:0] =
A[3:0] =
B[3:0] =
OpCode[2:0] =
out[7:0] =
Kunwar Arpit Singh 22185
1
8
3
000 001 010 011 100 101 000 001 010 011 100 101
4 14 3 1 3 2 11 5 24 0 11