

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

```
1 module and_gate_3_input (input A, input B, input C, output Y);
2     wire w1;
3     and g1(w1, A, B);
4     and g2(Y, w1, C);
5 endmodule
```

3-input OR Gate

```
1 module or_gate_3_input (input A, input B, input C, output Y);
2     wire w1;
3     or g1(w1, A, B);
4     or g2(Y, w1, C);
5 endmodule
```

Testbench

3-input AND Gate

```
1 module and_gate_3_input_tb;
2     reg A, B, C;
3     wire Y;
4     reg [8*24-1:0] name;
5
6     and_gate_3_input dut (.A(A), .B(B), .C(C), .Y(Y));
7
8     initial begin
```

```

9      name = "_Kunwar_Arpit_Singh_22185";
10     $dumpfile("assign1_problem1_and_gate_3_input.vcd");
11     $dumpvars(1, and_gate_3_input_tb);
12
13     $display("Kunwar_Arpit_Singh");
14     $display("A_B_C_Y");
15     $display("-----");
16     A = 0; B = 0; C = 0; #10;
17     $display("%b|_b|_b|_b", A, B, C, Y);
18     A = 0; B = 0; C = 1; #10;
19     $display("%b|_b|_b|_b", A, B, C, Y);
20     A = 0; B = 1; C = 0; #10;
21     $display("%b|_b|_b|_b", A, B, C, Y);
22     A = 0; B = 1; C = 1; #10;
23     $display("%b|_b|_b|_b", A, B, C, Y);
24     A = 1; B = 0; C = 0; #10;
25     $display("%b|_b|_b|_b", A, B, C, Y);
26     A = 1; B = 0; C = 1; #10;
27     $display("%b|_b|_b|_b", A, B, C, Y);
28     A = 1; B = 1; C = 0; #10;
29     $display("%b|_b|_b|_b", A, B, C, Y);
30     A = 1; B = 1; C = 1; #10;
31     $display("%b|_b|_b|_b", A, B, C, Y);
32
33     $finish;
34 end
35 endmodule

```

3-input OR Gate

```

1  module or_gate_3_input_tb;
2  reg A, B, C;
3  wire Y;
4  reg [8*24-1:0] name;
5  or_gate_3_input dut(.A(A), .B(B), .C(C), .Y(Y));
6
7  initial begin
8      name = "_Kunwar_Arpit_Singh_22185";
9      $display("Kunwar_Arpit_Singh");
10     $dumpfile("assign1_problem1_or_gate_3_input.vcd");
11     $dumpvars(1, or_gate_3_input_tb);
12
13     $display("Kunwar_Arpit_Singh");
14     $display("A_B_C_Y");
15     $display("-----");
16
17     A = 0; B = 0; C = 0; #10;
18     $display("%b|_b|_b|_b", A, B, C, Y);
19     A = 0; B = 0; C = 1; #10;
20     $display("%b|_b|_b|_b", A, B, C, Y);
21     A = 0; B = 1; C = 0; #10;
22     $display("%b|_b|_b|_b", A, B, C, Y);

```

```

23     A = 0; B = 1; C = 1; #10;
24     $display("%b|_%b|_%b|_%b", A, B, C, Y);
25     A = 1; B = 0; C = 0; #10;
26     $display("%b|_%b|_%b|_%b", A, B, C, Y);
27     A = 1; B = 0; C = 1; #10;
28     $display("%b|_%b|_%b|_%b", A, B, C, Y);
29     A = 1; B = 1; C = 0; #10;
30     $display("%b|_%b|_%b|_%b", A, B, C, Y);
31     A = 1; B = 1; C = 1; #10;
32     $display("%b|_%b|_%b|_%b", A, B, C, Y);
33
34     $finish;
35 end
36 endmodule

```

Output in Terminal

3-input AND Gate

```

1 VCD info: dumpfile assign1_problem1_and_gate_3_input.vcd opened for output.
2 Kunwar Arpit Singh
3 A | B | C | Y
4 -----
5 0 | 0 | 0 | 0
6 0 | 0 | 1 | 0
7 0 | 1 | 0 | 0
8 0 | 1 | 1 | 0
9 1 | 0 | 0 | 0
10 1 | 0 | 1 | 0
11 1 | 1 | 0 | 0
12 1 | 1 | 1 | 1

```

3-input OR Gate

```

1 VCD info: dumpfile assign1_problem1_and_gate_3_input.vcd opened for output.
2 Kunwar Arpit Singh
3 A | B | C | Y
4 -----
5 0 | 0 | 0 | 0
6 0 | 0 | 1 | 1
7 0 | 1 | 0 | 1
8 0 | 1 | 1 | 1
9 1 | 0 | 0 | 1
10 1 | 0 | 1 | 1
11 1 | 1 | 0 | 1
12 1 | 1 | 1 | 1

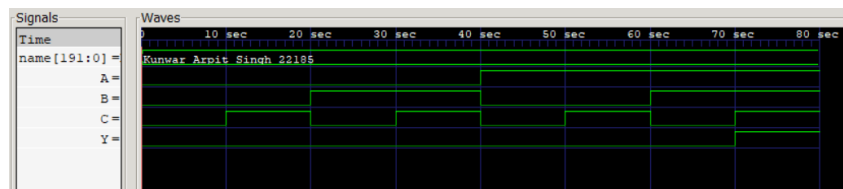
```

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
```

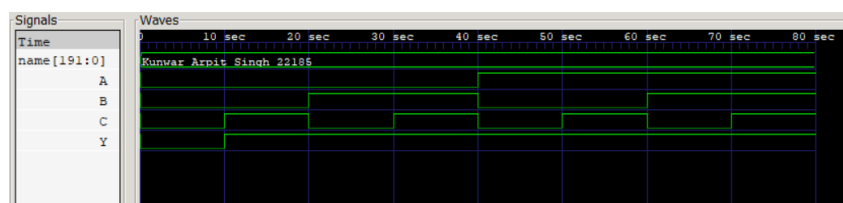
GTKWave waveform(3-input AND Gate)



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)

```
1 module comparator_1_bit(input A, input B, output eq, output gt, output lt);
2     xnor(eq, A, B);
3     and(gt, A, ~B);
4     and(lt, ~A, B);
5 endmodule
6
7 module comparator_4_bit(input [3:0] A, input [3:0] B, output a_eq_b, output
8     a_gt_b, output a_lt_b);
9     wire [3:0] eq, gt, lt;
10
11     comparator_1_bit comp0(A[0], B[0], eq[0], gt[0], lt[0]);
12     comparator_1_bit comp1(A[1], B[1], eq[1], gt[1], lt[1]);
13     comparator_1_bit comp2(A[2], B[2], eq[2], gt[2], lt[2]);
14     comparator_1_bit comp3(A[3], B[3], eq[3], gt[3], lt[3]);
15
16     assign a_eq_b = eq[0] & eq[1] & eq[2] & eq[3];
17     assign a_gt_b = gt[0] | (eq[0] & gt[1]) | (eq[0] & eq[1] & gt[2]) | (eq[0]
18         & eq[1] & eq[2] & gt[3]);
19     assign a_lt_b = lt[0] | (eq[0] & lt[1]) | (eq[0] & eq[1] & lt[2]) | (eq[0]
20         & eq[1] & eq[2] & lt[3]);
21 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.

```
1 module comparator_4_bit_tb;
2     reg [3:0] A, B;
3     wire a_eq_b;
4     wire a_gt_b;
5     wire a_lt_b;
6     integer i, j;
7     comparator_4_bit dut (.A(A), .B(B), .a_eq_b(a_eq_b), .a_gt_b(a_gt_b), .
8         a_lt_b(a_lt_b));
9     reg [8*24-1:0] name;
10
11     initial begin
12         name = "_Kunwar_Arpit_Singh_22185";
13         $display("Kunwar_Arpit_Singh");
14         $dumpfile("assign1_problem2_comparator_4_bit.vcd");
15         $dumpvars(1, comparator_4_bit_tb);
16     end
```

```

15
16     $display("Kunwar_Arpit_Singh");
17     $display("A_B|B_A|A==B|A>B|A<B");
18     $display("-----");
19
20     for (i = 0 ; i < 16 ; i = i+1) begin
21         for (j = 0 ; j < 16 ; j = j+1) begin
22             A = i;
23             B = j;
24             #10;
25             $display("%b| %b| %b| %b| %b", A, B, a_eq_b, a_gt_b,
26                     a_lt_b);
27         end
28     end
29 endmodule

```

Testbench (Specific outputs (For Clarity))

```

1     module comparator_4_bit_tb;
2     reg [3:0] A, B;
3     wire a_eq_b;
4     wire a_gt_b;
5     wire a_lt_b;
6     integer i, j;
7     comparator_4_bit dut (.A(A), .B(B), .a_eq_b(a_eq_b), .a_gt_b(a_gt_b), .
8         a_lt_b(a_lt_b));
9     reg [8*24-1:0] name;
10
11     initial begin
12         name = "Kunwar_Arpit_Singh_22185";
13         $display("Kunwar_Arpit_Singh");
14         $dumpfile("assign1_problem2_comparator_4_bit_small.vcd");
15         $dumpvars(1, comparator_4_bit_tb);
16
17         $display("Kunwar_Arpit_Singh");
18         $display("A_B|B_A|A==B|A>B|A<B");
19         $display("-----");
20
21         A = 4'b1010; B = 4'b1010; #10;
22         $display("%b| %b| %b| %b| %b", A, B, a_eq_b, a_gt_b, a_lt_b);
23         A = 4'b1010; B = 4'b1011; #10;
24         $display("%b| %b| %b| %b| %b", A, B, a_eq_b, a_gt_b, a_lt_b);
25         A = 4'b0000; B = 4'b1111; #10;
26         $display("%b| %b| %b| %b| %b", A, B, a_eq_b, a_gt_b, a_lt_b);
27         A = 4'b1111; B = 4'b1111; #10;
28         $display("%b| %b| %b| %b| %b", A, B, a_eq_b, a_gt_b, a_lt_b);
29         $finish;
30     end
31 endmodule

```

Output in Terminal

Specific outputs (For Clarity)

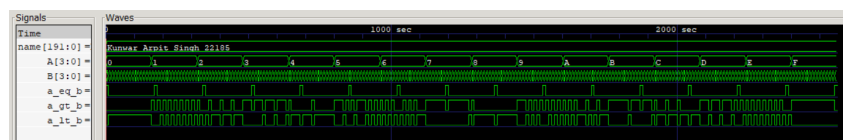
```
1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem2_comparator_4_bit_small.vcd opened for
  output.
3 Kunwar Arpit Singh
4 A | B | A == B | A > B | A < B
5 -----
6 1010 | 1010 | 1 | 0 | 0
7 1010 | 1011 | 0 | 0 | 1
8 0000 | 1111 | 0 | 0 | 1
9 1111 | 1111 | 1 | 0 | 0
```

Output Screenshots

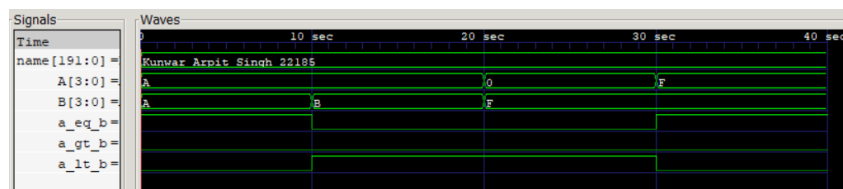
Terminal output(Specific outputs (For Clarity))

```
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

```
1 module half_adder(input A, input B, output SUM, output CARRY);
2     xor(SUM, A, B);
3     and(CARRY, A, B);
4 endmodule
```

Full Adder

```
1 'include "assign1_problem3_half_adder.v"
2 module full_adder(input A, input B, input CIN, output SUM, output COUT);
3     wire SUM1, CARRY1, CARRY2;
4     half_adder HA1(A, B, SUM1, CARRY1);
5     half_adder HA2(SUM1, CIN, SUM, CARRY2);
6     or g1(COUT, CARRY1, CARRY2);
7 endmodule
```

Half Subtractor

```
1 module half_subtractor (input A, input B, output diff, output borrow_out);
2     xor g1(diff, A, B);
3     wire not_A;
4     not g2(not_A, A);
5     and g3(borrow_out, not_A, B);
6 endmodule
```

Full Subtractor

```
1 'include "assign1_problem3_half_subtractor.v"
2 module full_subtractor (input A, input B, input borrow_in, output diff, output
3     borrow_out);
4     wire diff1, borrow1, borrow2;
5     half_subtractor hs1 (.A(A), .B(B), .diff(diff1), .borrow_out(borrow1));
6     half_subtractor hs2 (.A(diff1), .B(borrow_in), .diff(diff), .borrow_out(
7         borrow2));
8     or g1(borrow_out, borrow1, borrow2);
9 endmodule
```

Testbench

Half Adder

```
1 module half_adder_tb;
2     reg A, B;
3     wire SUM, CARRY;
```



```

4  reg [8*24-1:0] name;
5  half_adder dut (A, B, SUM, CARRY);
6
7  initial begin
8      name = "KunwarArpitSingh22185";
9      $display("KunwarArpitSingh");
10     $dumpfile("assign1_problem3_half_adder.vcd");
11     $dumpvars(1, half_adder_tb);
12     $display("KunwarArpitSingh");
13     $display("A|B|CARRY|SUM");
14     $display("-----");
15
16     A = 0 ; B = 0 ; #10;
17     $display("%b| %b| %b| %b", A, B, CARRY, SUM);
18     A = 0 ; B = 1 ; #10;
19     $display("%b| %b| %b| %b", A, B, CARRY, SUM);
20     A = 1 ; B = 0 ; #10;
21     $display("%b| %b| %b| %b", A, B, CARRY, SUM);
22     A = 1 ; B = 1 ; #10;
23     $display("%b| %b| %b| %b", A, B, CARRY, SUM);
24     $finish;
25 end
26 endmodule

```

Full Adder

```

1  module full_adder_tb;
2      reg A, B, CIN;
3      wire SUM, COUT;
4      integer i;
5      reg [8*24-1:0] name;
6      full_adder dut (A, B, CIN, SUM, COUT);
7      initial begin
8          name = "KunwarArpitSingh22185";
9          $display("KunwarArpitSingh");
10         $dumpfile("assign1_problem3_full_adder.vcd");
11         $dumpvars(1, full_adder_tb);
12         $display("KunwarArpitSingh");
13         $display("A|B|CIN|SUM|COUT");
14         $display("-----");
15         for (i = 0 ; i < 8 ; i = i + 1) begin
16             {A, B, CIN} = i;
17             #10;
18             $display("%b| %b| %b| %b| %b", A, B, CIN, SUM, COUT);
19         end
20         $finish;
21     end
22 endmodule

```

Half Subtractor

```

1  module half_subtractor_tb;
2      reg A, B;

```

```

3  wire diff, borrow_out;
4  reg [8*24-1:0] name;
5  half_subtractor dut(A, B, diff, borrow_out);
6
7  initial begin
8      name = "_Kunwar_Arpit_Singh_22185";
9      $display("Kunwar_Arpit_Singh");
10     $dumpfile("assign1_problem3_half_subtractor.vcd");
11     $dumpvars(1, half_subtractor_tb);
12     $display("A_|_B_|_diff_|_borrow_out");
13     $display("-----");
14     A = 0 ; B = 0 ; #10;
15     $display("%b_|_%b_|_%b_|_%b", A, B, diff, borrow_out);
16     A = 0 ; B = 1 ; #10;
17     $display("%b_|_%b_|_%b_|_%b", A, B, diff, borrow_out);
18     A = 1 ; B = 0 ; #10;
19     $display("%b_|_%b_|_%b_|_%b", A, B, diff, borrow_out);
20     A = 1 ; B = 1 ; #10;
21     $display("%b_|_%b_|_%b_|_%b", A, B, diff, borrow_out);
22     $finish;
23 end
24 endmodule

```

Full Subtractor

```

1  module full_subtractor_tb;
2
3      reg A, B, borrow_in;
4      wire diff, borrow_out;
5      integer i;
6      reg [8*24-1:0] name;
7
8      full_subtractor dut (.A(A), .B(B), .borrow_in(borrow_in), .diff(diff), .
9          borrow_out(borrow_out));
10
11     initial begin
12         name = "_Kunwar_Arpit_Singh_22185";
13         $display("Kunwar_Arpit_Singh");
14         $dumpfile("assign1_problem3_full_subtractor.vcd");
15         $dumpvars(1, full_subtractor_tb);
16
17         $display("A_|_B_|_borrow_in_|_diff_|_borrow_out");
18         $display("-----");
19
20         for ( i = 0 ; i < 8; i = i + 1) begin
21             {A, B, borrow_in} = i;
22             #10;
23             $display("_%b_|_%b_|_%b_|_%b_|_%b", A, B, borrow_in,
24                 diff, borrow_out);
25         end
26     end
27     $finish;
28 end

```

26 `endmodule`

Output in Terminal

Half Adder

```
1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem3_half_adder.vcd opened for output.
3 Kunwar Arpit Singh
4 A | B | CARRY | SUM
5 -----
6 0 | 0 | 0 | 0
7 0 | 1 | 0 | 1
8 1 | 0 | 0 | 1
9 1 | 1 | 1 | 0
```

Full Adder

```
1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem3_full_adder.vcd opened for output.
3 Kunwar Arpit Singh
4 A B CIN | SUM COUT
5 -----
6 0 0 0 | 0 0
7 0 0 1 | 1 0
8 0 1 0 | 1 0
9 0 1 1 | 0 1
10 1 0 0 | 1 0
11 1 0 1 | 0 1
12 1 1 0 | 0 1
13 1 1 1 | 1 1
```

Half Subtractor

```
1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem3_half_subtractor.vcd opened for output.
3 A | B | diff | borrow_out
4 -----
5 0 | 0 | 0 | 0
6 0 | 1 | 1 | 1
7 1 | 0 | 1 | 0
8 1 | 1 | 0 | 0
```

Full Subtractor

```
1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem3_full_subtractor.vcd opened for output.
3 A | B | borrow_in | diff | borrow_out
4 -----
5 0 | 0 | 0 | 0 | 0
6 0 | 0 | 1 | 1 | 1
7 0 | 1 | 0 | 1 | 1
8 0 | 1 | 1 | 0 | 1
```

9	1	0	0	1	0
10	1	0	1	0	0
11	1	1	0	0	0
12	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
```

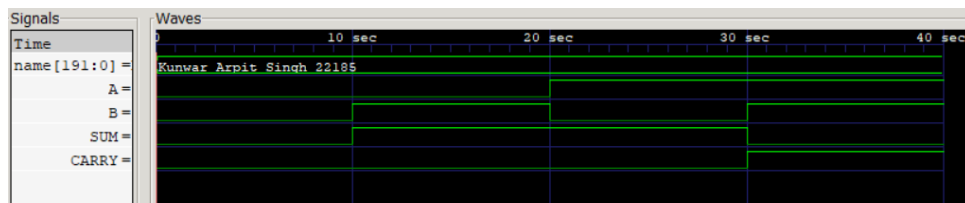
Terminal output (Half Subtractor)

```
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
```

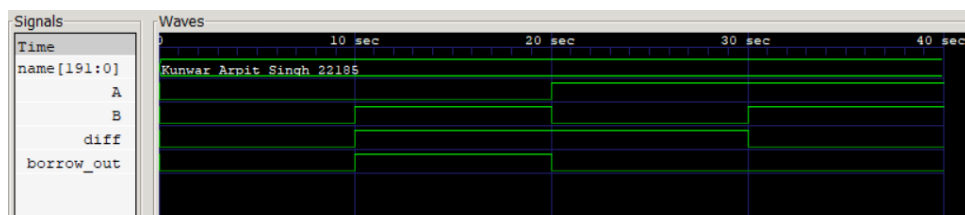
GTKWave waveform (Half Adder)



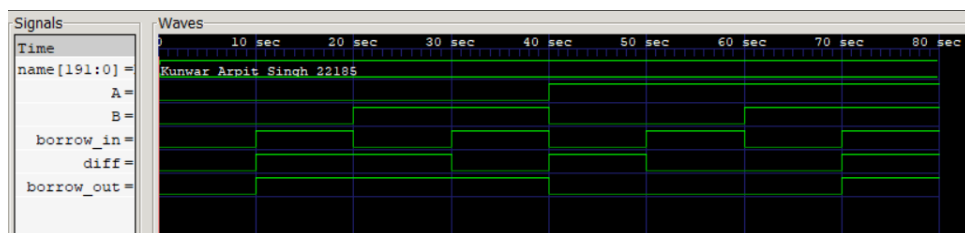
GTKWave waveform (Full Adder)



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

```
1 module mux_4_to_1(input [3:0] d_in, input[1:0] s, output y);
2     wire s0_not, s1_not;
3     wire w0, w1, w2, w3;
4
5     not(s0_not, s[0]);
6     not(s1_not, s[1]);
7
8     and(w0, d_in[0], s0_not, s1_not);
9     and(w1, d_in[1], s[0], s1_not);
10    and(w2, d_in[2], s0_not, s[1]);
11    and(w3, d_in[3], s[0], s[1]);
12
13    or(y, w0, w1, w2, w3);
14 endmodule
```

3-to-8 decoder

```
1 module decoder_3_to_8(input [2:0] d, output [7:0] y);
2
3     wire d0_not, d1_not, d2_not;
4
5     not (d0_not, d[0]);
6     not (d1_not, d[1]);
7     not (d2_not, d[2]);
8
9     and (y[0], d0_not, d1_not, d2_not);
10    and (y[1], d[0], d1_not, d2_not);
11    and (y[2], d0_not, d[1], d2_not);
12    and (y[3], d[0], d[1], d2_not);
13    and (y[4], d0_not, d1_not, d[2]);
14    and (y[5], d[0], d1_not, d[2]);
15    and (y[6], d0_not, d[1], d[2]);
16    and (y[7], d[0], d[1], d[2]);
17
18 endmodule
```

Testbench

4-to-1 multiplexer

```
1 module mux_4_to_1_tb;
2     reg [3:0] d_in;
```

```

3  reg [1:0] s;
4  wire y;
5  reg [8*24-1:0] name;
6
7  mux_4_to_1 dut(.d_in(d_in), .s(s), .y(y));
8
9  initial begin
10     name = "_Kunwar_Arpit_Singh_22185";
11     $display("Kunwar_Arpit_Singh");
12     $dumpfile("assign1_problem4_mux_4_to_1.vcd");
13     $dumpvars(1, mux_4_to_1_tb);
14
15     d_in = 4'b1010;
16     $display("s1|s0|y(selected_data)");
17     $display("-----");
18     s = 2'b00; #10;
19     $display("_%b|_%b|_%b(d0)", s[1], s[0], y);
20     s = 2'b01; #10;
21     $display("_%b|_%b|_%b(d1)", s[1], s[0], y);
22     s = 2'b10; #10;
23     $display("_%b|_%b|_%b(d2)", s[1], s[0], y);
24     s = 2'b11; #10;
25     $display("_%b|_%b|_%b(d3)", s[1], s[0], y);
26
27     d_in = 4'b0101;
28     $display("s1|s0|y(selected_data)");
29     $display("-----");
30     s = 2'b00; #10;
31     $display("_%b|_%b|_%b(d0)", s[1], s[0], y);
32     s = 2'b01; #10;
33     $display("_%b|_%b|_%b(d1)", s[1], s[0], y);
34     s = 2'b10; #10;
35     $display("_%b|_%b|_%b(d2)", s[1], s[0], y);
36     s = 2'b11; #10;
37     $display("_%b|_%b|_%b(d3)", s[1], s[0], y);
38
39     d_in = 4'b1111;
40     $display("s1|s0|y(selected_data)");
41     $display("-----");
42     s = 2'b00; #10;
43     $display("_%b|_%b|_%b(d0)", s[1], s[0], y);
44     s = 2'b01; #10;
45     $display("_%b|_%b|_%b(d1)", s[1], s[0], y);
46     s = 2'b10; #10;
47     $display("_%b|_%b|_%b(d2)", s[1], s[0], y);
48     s = 2'b11; #10;
49     $display("_%b|_%b|_%b(d3)", s[1], s[0], y);
50
51     $finish;
52 end
53 endmodule

```

3-to-8 decoder (Using Registers)

```

1 module decoder_3_to_8_tb;
2     reg[2:0] d;
3     wire[7:0] y;
4     reg [8*24-1:0] name;
5
6     decoder_3_to_8 dut (d, y);
7
8     initial begin
9         name = "_Kunwar_Arpit_Singh_22185";
10        $display("Kunwar_Arpit_Singh");
11        $dumpfile("assign1_problem4_3_to_8_decoder.vcd");
12        $dumpvars(1, decoder_3_to_8_tb);
13
14        $display("d2_d1_d0||y7_y6_y5_y4_y3_y2_y1_y0");
15        $display("-----");
16        d = 3'b000; #10;
17        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
18            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
19        d = 3'b001; #10;
20        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
21            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
22        d = 3'b010; #10;
23        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
24            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
25        d = 3'b011; #10;
26        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
27            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
28        d = 3'b100; #10;
29        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
30            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
31        d = 3'b101; #10;
32        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
33            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
34        d = 3'b110; #10;
35        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
36            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
37        d = 3'b111; #10;
38        $display("%b||%b||%b||%b||%b||%b||%b||%b", d
39            [2], d[1], d[0], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);
40    end
41 endmodule

```

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

```

1 module decoder_3_to_8_tb;
2     reg d0, d1, d2;
3     wire y0, y1, y2, y3, y4, y5, y6, y7;
4     reg [8*24-1:0] name;
5     decoder_3_to_8 dut (.d({d2, d1, d0}), .y({y7, y6, y5, y4, y3, y2, y1, y0}))
6     ;

```



```

7   initial begin
8       name = "Kunwar_Arpit_Singh_22185";
9       $display("Kunwar_Arpit_Singh");
10      $dumpfile("assign1_problem4_3_to_8_decoder.vcd");
11      $dumpvars(1, decoder_3_to_8_tb);
12
13      $display("d2_d1_d0_y7_y6_y5_y4_y3_y2_y1_y0");
14      $display("-----");
15      d0 = 1'b0; d1 = 1'b0; d2 = 1'b0; #10;
16      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
17      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
18      d0 = 1'b1; d1 = 1'b0; d2 = 1'b0; #10;
19      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
20      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
21      d0 = 1'b0; d1 = 1'b1; d2 = 1'b0; #10;
22      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
23      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
24      d0 = 1'b0; d1 = 1'b0; d2 = 1'b1; #10;
25      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
26      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
27      d0 = 1'b1; d1 = 1'b0; d2 = 1'b1; #10;
28      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
29      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
30      d0 = 1'b1; d1 = 1'b1; d2 = 1'b1; #10;
31      $display("%b_%b_%b_|_%b_%b_%b_|_%b_%b_%b_|_%b", d2
32      , d1, d0, y7, y6, y5, y4, y3, y2, y1, y0);
31  end
32  endmodule

```

Output in Terminal

4-to-1 multiplexer

```

1  Kunwar Arpit Singh
2  VCD info: dumpfile assign1_problem4_mux_4_to_1.vcd opened for output.
3  s1 | s0 | y (selected data)
4  -----
5  0 | 0 | 0 (d0)
6  0 | 1 | 1 (d1)
7  1 | 0 | 0 (d2)
8  1 | 1 | 1 (d3)
9  s1 | s0 | y (selected data)
10 -----
11 0 | 0 | 1 (d0)
12 0 | 1 | 0 (d1)

```

```

13 1 | 0 | 1 (d2)
14 1 | 1 | 0 (d3)
15 s1 | s0 | y (selected data)
16 -----
17 0 | 0 | 1 (d0)
18 0 | 1 | 1 (d1)
19 1 | 0 | 1 (d2)
20 1 | 1 | 1 (d3)

```

3-to-8 decoder (Using Registers)

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for output.
3 d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
4 -----
5 0 | 0 | 0 || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
6 0 | 0 | 1 || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
7 0 | 1 | 0 || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
8 0 | 1 | 1 || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
9 1 | 0 | 0 || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
10 1 | 0 | 1 || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
11 1 | 1 | 0 || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
12 1 | 1 | 1 || 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0

```

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

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for output.
3 d2 | d1 | d0 || y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0
4 -----
5 0 | 0 | 0 || 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1
6 0 | 0 | 1 || 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0
7 0 | 1 | 0 || 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0
8 0 | 1 | 1 || 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0
9 1 | 0 | 0 || 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0
10 1 | 0 | 1 || 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0
11 1 | 1 | 0 || 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0
12 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)
```

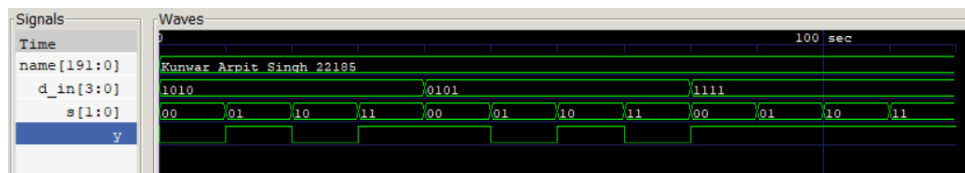
Terminal output(3-to-8 decoder (Using Registers))

```
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
```

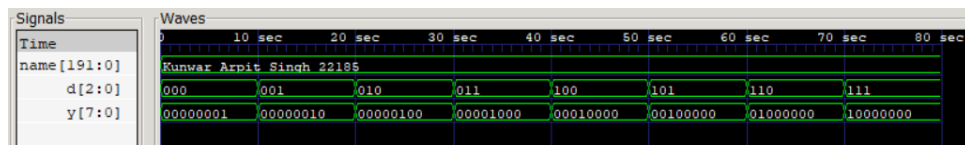
Terminal output(3-to-8 decoder (Using 1-bit at a time))

```
Kunwar Arpit Singh
VCD info: dumpfile assign1_problem4_3_to_8_decoder.vcd opened for 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
```

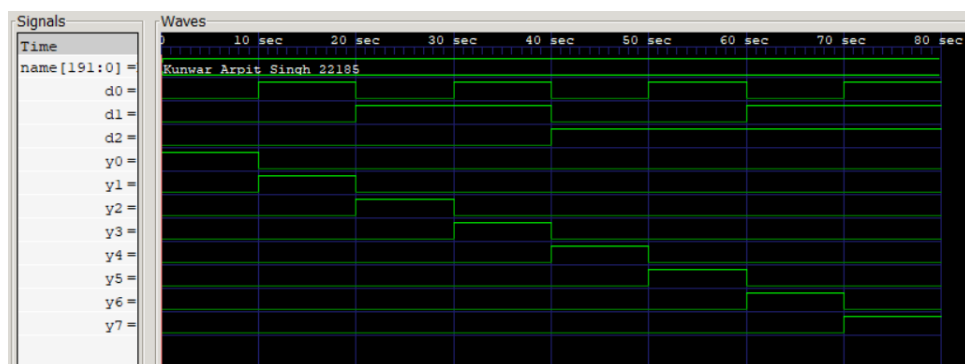
Terminal output(4-to-1 multiplexer)



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)

```
1 module barrel_shifter_4bit (input [3:0] A, input [1:0] sel, input dir, output
  [3:0] out);
2   wire [3:0] left1, left2;
3   wire [3:0] right1, right2;
4   wire [3:0] sel_left, sel_right;
5
6   assign left1 = {A[2:0], 1'b0};
7   assign left2 = {A[1:0], 2'b00};
8
9   assign sel_left = (sel == 2'b00) ? A : (sel == 2'b01) ? left1 : left2;
10
11  assign right1 = {1'b0, A[3:1]};
12  assign right2 = {2'b00, A[3:2]};
13
14  assign sel_right = (sel == 2'b00) ? A : (sel == 2'b01) ? right1 : right2;
15
16  assign out = (dir == 0) ? sel_left : sel_right;
17 endmodule
```

Testbench

```
1 module barrel_shifter_4bit_tb;
2   reg [3:0] A;
3   reg [1:0] sel;
4   reg dir;
5   wire [3:0] out;
6   reg [8*24-1:0] name;
7
8   barrel_shifter_4bit dut (.A(A), .sel(sel), .dir(dir), .out(out));
9
10  initial begin
11      name = "_Kunwar_Arpit_Singh_22185";
12
13      $display("Kunwar_Arpit_Singh");
14      $dumpfile("assign1_problem5_barrel_shifter_tb.vcd");
15      $dumpvars(1, barrel_shifter_4bit_tb);
16
17      $display("A_|_dir_|_Shift_by_|_out");
18      $display("-----");
19
20      A = 4'b1011; dir = 0; sel = 2'b00; #10;
```

```

21     $display("%b|_left_|_%0d_|_%b", A, sel, out);
22     A = 4'b1011; dir = 0; sel = 2'b01; #10;
23     $display("%b|_left_|_%0d_|_%b", A, sel, out);
24     A = 4'b1011; dir = 0; sel = 2'b10; #10;
25     $display("%b|_left_|_%0d_|_%b", A, sel, out);
26     A = 4'b1011; dir = 1; sel = 2'b00; #10;
27     $display("%b|_right_|_%0d_|_%b", A, sel, out);
28     A = 4'b1011; dir = 1; sel = 2'b01; #10;
29     $display("%b|_right_|_%0d_|_%b", A, sel, out);
30     A = 4'b1011; dir = 1; sel = 2'b10; #10;
31     $display("%b|_right_|_%0d_|_%b", A, sel, out);
32     $finish;
33 end
34 endmodule

```

Output in Terminal

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign1_problem5_barrel_shifter_tb.vcd opened for output.
3 A | dir | Shift by | out
4 -----
5 1011 | left | 0 | 1011
6 1011 | left | 1 | 0110
7 1011 | left | 2 | 1100
8 1011 | right | 0 | 1011
9 1011 | right | 1 | 0101
10 1011 | right | 2 | 0010

```

Output Screenshots

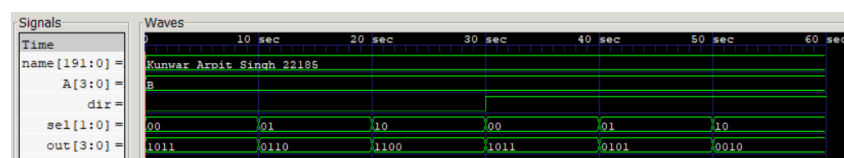
Terminal output (Barrel Shifter)

```

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)

```
1 module half_adder(input A, input B, output SUM, output CARRY);
2
3     xor(SUM, A, B);
4     and(CARRY, A, B);
5 endmodule
6
7 module full_adder(input A, input B, input CIN, output SUM, output COUT);
8     wire SUM1, CARRY1, CARRY2;
9
10    half_adder HA1(A, B, SUM1, CARRY1);
11
12    half_adder HA2(SUM1, CIN, SUM, CARRY2);
13
14    or g1(COUT, CARRY1, CARRY2);
15 endmodule
16
17 module mux_8_to_1_bit (input d0, d1, d2, d3, d4, d5, d6, d7, input [2:0] sel,
18     output out);
19     wire s0not, s1not, s2not;
20     wire w0, w1, w2, w3, w4, w5, w6, w7;
21     not (s0not, sel[0]);
22     not (s1not, sel[1]);
23     not (s2not, sel[2]);
24
25     and(w0, d0, s2not, s1not, s0not);
26     and(w1, d1, s2not, s1not, sel[0]);
27     and(w2, d2, s2not, sel[1], s0not);
28     and(w3, d3, s2not, sel[1], sel[0]);
29     and(w4, d4, sel[2], s1not, s0not);
30     and(w5, d5, sel[2], s1not, sel[0]);
31     and(w6, d6, sel[2], sel[1], s0not);
32     and(w7, d7, sel[2], sel[1], sel[0]);
33
34     or(out, w0, w1, w2, w3, w4, w5, w6, w7);
35 endmodule
36
37 module four_bit_adder(input [3:0] A, input [3:0] B, input CARRYIN, output
38     [3:0] SUM, output CARRYOUT);
39
40     wire C1, C2, C3;
```

```

40
41     full_adder FA0(A[0], B[0], CARRYIN, SUM[0], C1);
42     full_adder FA1(A[1], B[1], C1, SUM[1], C2);
43     full_adder FA2(A[2], B[2], C2, SUM[2], C3);
44     full_adder FA3(A[3], B[3], C3, SUM[3], CARRYOUT);
45 endmodule
46
47 module four_bit_subtractor (input [3:0] A, input [3:0] B, output [3:0]
    DIFFERENCE);
48     wire [3:0] B_COMPLEMENT;
49     wire CARRYOUT;
50     not (B_COMPLEMENT[0], B[0]);
51     not (B_COMPLEMENT[1], B[1]);
52     not (B_COMPLEMENT[2], B[2]);
53     not (B_COMPLEMENT[3], B[3]);
54
55     four_bit_adder SUBTRACTOR_ADDER (A, B_COMPLEMENT, 1'b1, DIFFERENCE,
        CARRYOUT);
56 endmodule
57
58 module four_by_four_multiplier (
59     input [3:0] A,
60     input [3:0] B,
61     output [7:0] PRODUCT
62 );
63     wire [3:0] PP0, PP1, PP2, PP3;
64
65     and (PP0[0], A[0], B[0]);
66     and (PP0[1], A[1], B[0]);
67     and (PP0[2], A[2], B[0]);
68     and (PP0[3], A[3], B[0]);
69
70     and (PP1[0], A[0], B[1]);
71     and (PP1[1], A[1], B[1]);
72     and (PP1[2], A[2], B[1]);
73     and (PP1[3], A[3], B[1]);
74
75     and (PP2[0], A[0], B[2]);
76     and (PP2[1], A[1], B[2]);
77     and (PP2[2], A[2], B[2]);
78     and (PP2[3], A[3], B[2]);
79
80     and (PP3[0], A[0], B[3]);
81     and (PP3[1], A[1], B[3]);
82     and (PP3[2], A[2], B[3]);
83     and (PP3[3], A[3], B[3]);
84
85     wire c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11;
86     wire s1, s2, s3, s4, s5, s6;
87
88     assign PRODUCT[0] = PP0[0];

```



```

89     half_adder HA1 (PP0[1], PP1[0], PRODUCT[1], c1);
90
91     full_adder FA1 (PP0[2], PP1[1], c1, s1, c2);
92     half_adder HA2 (s1, PP2[0], PRODUCT[2], c3);
93
94     full_adder FA2 (PP0[3], PP1[2], c2, s2, c4);
95     full_adder FA3 (s2, PP2[1], c3, PRODUCT[3], c5);
96
97     full_adder FA4 (PP1[3], PP2[2], c4, s3, c6);
98     full_adder FA5 (s3, PP3[0], c5, PRODUCT[4], c7);
99
100    full_adder FA6 (PP2[3], PP3[1], c6, s4, c8);
101    full_adder FA7 (s4, c7, 1'b0, PRODUCT[5], c9);
102
103    full_adder FA8 (PP3[2], c8, c9, PRODUCT[6], c10);
104
105    full_adder FA9 (PP3[3], c10, 1'b0, PRODUCT[7], c11);
106
107    endmodule
108
109
110
111    module four_bit_and (input [3:0] A, input [3:0] B, output [3:0] out);
112        and(out[0], A[0], B[0]);
113        and(out[1], A[1], B[1]);
114        and(out[2], A[2], B[2]);
115        and(out[3], A[3], B[3]);
116    endmodule
117
118    module four_bit_or(input [3:0] A, input [3:0] B, output [3:0] out);
119        or(out[0], A[0], B[0]);
120        or(out[1], A[1], B[1]);
121        or(out[2], A[2], B[2]);
122        or(out[3], A[3], B[3]);
123    endmodule
124
125    module four_bit_xor(input [3:0] A, input [3:0] B, output [3:0] out);
126        xor(out[0], A[0], B[0]);
127        xor(out[1], A[1], B[1]);
128        xor(out[2], A[2], B[2]);
129        xor(out[3], A[3], B[3]);
130    endmodule
131
132    module simple_alu_4_bit (input [3:0] A, input [3:0] B, input [2:0] OpCode,
133        output [7:0] out);
134        wire [7:0] ADDER_OUT;
135        wire [7:0] SUBTRACTOR_OUT;
136        wire [7:0] MULTIPLIER_OUT;
137        wire [7:0] AND_OUT;
138        wire [7:0] OR_OUT;
139        wire [7:0] XOR_OUT;

```

```

139 wire CARRYOUT;
140
141 four_bit_adder adder_unit(A, B, 1'b0, ADDER_OUT[3:0], CARRYOUT);
142 four_bit_subtractor subtractor_unit(A, B, SUBTRACTOR_OUT[3:0]);
143 four_by_four_multiplier multiplier_unit(A, B, MULTIPLIER_OUT);
144 four_bit_and and_unit(A, B, AND_OUT[3:0]);
145 four_bit_or or_unit(A, B, OR_OUT[3:0]);
146 four_bit_xor xor_unit(A, B, XOR_OUT[3:0]);
147
148 assign ADDER_OUT[7:4] = {3'b000, CARRYOUT};
149 assign SUBTRACTOR_OUT[7:4] = 4'b0;
150 assign AND_OUT[7:4] = 4'b0;
151 assign OR_OUT[7:4] = 4'b0;
152 assign XOR_OUT[7:4] = 4'b0;
153
154 mux_8_to_1_bit MUX0 (ADDER_OUT[0], SUBTRACTOR_OUT[0], MULTIPLIER_OUT[0],
155     AND_OUT[0], OR_OUT[0], XOR_OUT[0], 1'b0, 1'b0, OpCode, out[0]);
156 mux_8_to_1_bit MUX1 (ADDER_OUT[1], SUBTRACTOR_OUT[1], MULTIPLIER_OUT[1],
157     AND_OUT[1], OR_OUT[1], XOR_OUT[1], 1'b0, 1'b0, OpCode, out[1]);
158 mux_8_to_1_bit MUX2 (ADDER_OUT[2], SUBTRACTOR_OUT[2], MULTIPLIER_OUT[2],
159     AND_OUT[2], OR_OUT[2], XOR_OUT[2], 1'b0, 1'b0, OpCode, out[2]);
160 mux_8_to_1_bit MUX3 (ADDER_OUT[3], SUBTRACTOR_OUT[3], MULTIPLIER_OUT[3],
161     AND_OUT[3], OR_OUT[3], XOR_OUT[3], 1'b0, 1'b0, OpCode, out[3]);
162 mux_8_to_1_bit MUX4 (ADDER_OUT[4], SUBTRACTOR_OUT[4], MULTIPLIER_OUT[4],
163     AND_OUT[4], OR_OUT[4], XOR_OUT[4], 1'b0, 1'b0, OpCode, out[4]);
164 mux_8_to_1_bit MUX5 (ADDER_OUT[5], SUBTRACTOR_OUT[5], MULTIPLIER_OUT[5],
165     AND_OUT[5], OR_OUT[5], XOR_OUT[5], 1'b0, 1'b0, OpCode, out[5]);
166 mux_8_to_1_bit MUX6 (ADDER_OUT[6], SUBTRACTOR_OUT[6], MULTIPLIER_OUT[6],
167     AND_OUT[6], OR_OUT[6], XOR_OUT[6], 1'b0, 1'b0, OpCode, out[6]);
168 mux_8_to_1_bit MUX7 (ADDER_OUT[7], SUBTRACTOR_OUT[7], MULTIPLIER_OUT[7],
169     AND_OUT[7], OR_OUT[7], XOR_OUT[7], 1'b0, 1'b0, OpCode, out[7]);
170 endmodule

```

Testbench

```

1 module simple_alu_4_bit_tb;
2     reg [3:0] A;
3     reg [3:0] B;
4     reg [2:0] OpCode;
5     wire signed [7:0] out;
6     reg [8*24-1:0] name;
7
8     simple_alu_4_bit dut (.A(A), .B(B), .OpCode(OpCode), .out(out));
9
10    initial begin
11        name = "_Kunwar_Arpit_Singh_22185";
12        $display("Kunwar_Arpit_Singh");
13        $dumpfile("assign1_problem6_ALU_tb.vcd");
14        $dumpvars(1, simple_alu_4_bit_tb);
15    end

```


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)

