

ECS 409/609

Behavioral Based Assignment

Solutions

Submission Deadline: September 10, 2025

1. Problem 1

Write behavioral verilog code to multiply a 4-bit input by 2.

```
1 module multiply_4_bit_by_2 (input [3:0] A, output [4:0] out);
2     assign out = A << 1;
3 endmodule
```

```

1 module multiply_4_bit_by_2_tb;
2     reg [3:0] A;
3     wire [4:0] out;
4     reg [8*24-1:0] name;
5
6     multiply_4_bit_by_2 dut (.A(A), .out(out));
7
8     initial begin
9         $display("Kunwar_Arpit_Singh");
10        name = "Kunwar_Arpit_Singh_22185";
11        $dumpfile("assign2_problem1_4_bit_multiply_by_2.vcd");
12        $dumpvars(1, multiply_4_bit_by_2_tb);
13        $display("Kunwar_Arpit_Singh");
14
15        $display("A (Binary & Decimal) | Output (Binary) | Output (Decimal)");
16        $display("-----");
17
18        A = 4'b0000; #10;
19        $display("%b| %d | %b | %d", A, A, out, out);
20        A = 4'b0001; #10;

```

```

21     $display("%b%04d%04b%04d", A, A, out, out);
22     A = 4'b0010; #10;
23     $display("%b%04d%04b%04d", A, A, out, out);
24     A = 4'b0011; #10;
25     $display("%b%04d%04b%04d", A, A, out, out);
26     A = 4'b0100; #10;
27     $display("%b%04d%04b%04d", A, A, out, out);
28     A = 4'b0101; #10;
29     $display("%b%04d%04b%04d", A, A, out, out);
30     A = 4'b0110; #10;
31     $display("%b%04d%04b%04d", A, A, out, out);
32     A = 4'b0111; #10;
33     $display("%b%04d%04b%04d", A, A, out, out);
34     A = 4'b1000; #10;
35     $display("%b%04d%04b%04d", A, A, out, out);
36     A = 4'b1001; #10;
37     $display("%b%04d%04b%04d", A, A, out, out);
38     A = 4'b1010; #10;
39     $display("%b%04d%04b%04d", A, A, out, out);
40     A = 4'b1011; #10;
41     $display("%b%04d%04b%04d", A, A, out, out);
42     A = 4'b1100; #10;
43     $display("%b%04d%04b%04d", A, A, out, out);
44     A = 4'b1101; #10;
45     $display("%b%04d%04b%04d", A, A, out, out);
46     A = 4'b1110; #10;
47     $display("%b%04d%04b%04d", A, A, out, out);
48     A = 4'b1111; #10;
49     $display("%b%04d%04b%04d", A, A, out, out);
50     $finish;
51 end
52 endmodule

```

Output in Terminal

```

1 VCD info: dumpfile assign2_problem1_4_bit_multiply_by_2_tb.vcd opened for
  output.
2 Kunwar Arpit Singh
3 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
4 -----
5 0000 | 0 | 00000 | 0
6 0001 | 1 | 00010 | 2
7 0010 | 2 | 00100 | 4
8 0011 | 3 | 00110 | 6
9 0100 | 4 | 01000 | 8
10 0101 | 5 | 01010 | 10
11 0110 | 6 | 01100 | 12
12 0111 | 7 | 01110 | 14
13 1000 | 8 | 10000 | 16
14 1001 | 9 | 10010 | 18

```

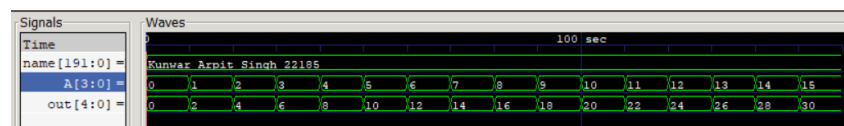
15	1010		10		10100		20
16	1011		11		10110		22
17	1100		12		11000		24
18	1101		13		11010		26
19	1110		14		11100		28
20	1111		15		11110		30

Output Screenshots

Terminal output

```
Kunwar Arpit Singh
VCD info: dumpfile assign2_problem1_4_bit_multiply_by_2.vcd opened for output.
Kunwar Arpit Singh
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
0000 | 0 | 00000 | 0
0001 | 1 | 00010 | 2
0010 | 2 | 00100 | 4
0011 | 3 | 00110 | 6
0100 | 4 | 01000 | 8
0101 | 5 | 01010 | 10
0110 | 6 | 01100 | 12
0111 | 7 | 01110 | 14
1000 | 8 | 10000 | 16
1001 | 9 | 10010 | 18
1010 | 10 | 10100 | 20
1011 | 11 | 10110 | 22
1100 | 12 | 11000 | 24
1101 | 13 | 11010 | 26
1110 | 14 | 11100 | 28
1111 | 15 | 11110 | 30
```

GTKWave waveform



2. Problem 2

Problem statement

Implement a behavioral verilog code for a 4-to-2 encoder and 2-to-4 decoder.

Verilog source (Behavioral)

4-to-2 Encoder:

```
1 module encoder_4_to_2 (input [3:0] A, output [1:0] out);
2     assign out = (A == 4'b1000) ? 2'b11 :
3                 (A == 4'b0100) ? 2'b10 :
4                 (A == 4'b0010) ? 2'b01 :
5                 (A == 4'b0001) ? 2'b00 : 2'bxx;
6 endmodule
```

2-to-4 Decoder:

```
1 module decoder_2_to_4(input [1:0] A, output [3:0] out);
2     assign out = (A == 2'b00) ? 4'b0001 :
3                 (A == 2'b01) ? 4'b0010 :
4                 (A == 2'b10) ? 4'b0100 :
5                 (A == 2'b11) ? 4'b1000 : 4'bxxxx;
6 endmodule
```

Testbench

4-to-2 Encoder:

This testbench shows two tables one for only the valid cases for 4 to 2 encoder and the second for all the cases for 4 to 2 encoder.

```
1 module encoder_4_to_2_tb;
2     reg [3:0] A;
3     wire [1:0] out;
4     reg [8*24-1:0] name;
5
6     encoder_4_to_2 dut (.A(A), .out(out));
7
8     initial begin
9         name = "_Kunwar_Arpit_Singh_22185";
10        $dumpfile("assign2_problem2_encoder_4_to_2.vcd");
11        $dumpvars(1, encoder_4_to_2_tb);
12        $display("Kunwar_Arpit_Singh");
13        $display("Output_for_only_the_valid_cases_for_4_to_2_encoder");
14        $display("A_(Binary_&Decimal)_|_Output(Binary)_|_Output(Decimal)");
15        $display("-----");
16        A = 4'b0001; #10;
17        $display("%b|d|b|d", A, A, out, out);
18        A = 4'b0010; #10;
19        $display("%b|d|b|d", A, A, out, out);
20        A = 4'b0100; #10;
```

```

21     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
22     A = 4'b1000; #10;
23     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
24     $display("\nOutput for all the cases for 4-to-2 encoder");
25     $display("A(Binary & Decimal) | Output(Binary) | Output(Decimal)");
26     $display("-----");
27     A = 4'b0000; #10;
28     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
29     A = 4'b0001; #10;
30     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
31     A = 4'b0010; #10;
32     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
33     A = 4'b0011; #10;
34     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
35     A = 4'b0100; #10;
36     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
37     A = 4'b0101; #10;
38     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
39     A = 4'b0110; #10;
40     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
41     A = 4'b0111; #10;
42     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
43     A = 4'b1000; #10;
44     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
45     A = 4'b1001; #10;
46     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
47     A = 4'b1010; #10;
48     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
49     A = 4'b1011; #10;
50     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
51     A = 4'b1100; #10;
52     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
53     A = 4'b1101; #10;
54     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
55     A = 4'b1110; #10;
56     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
57     A = 4'b1111; #10;
58     $display("%b0000|000%d00|000%b00|000%d", A, A, out, out);
59     $finish;
60     end
61 endmodule

```

2-to-4 Decoder:

```

1 module decoder_2_to_4_tb;
2     reg[1:0] A;
3     wire[3:0] out;
4     reg[8*24-1:0] name;
5
6     decoder_2_to_4 dut (.A(A), .out(out));
7
8     initial begin

```

```

9      name = "\Kunwar_Arpit_Singh_22185";
10     $display("Kunwar_Arpit_Singh");
11     $dumpfile("assign2_problem2_decoder_2_to_4.vcd");
12     $dumpvars(1, decoder_2_to_4_tb);
13     $display("Kunwar_Arpit_Singh");
14     $display("A_(Binary_&Decimal)_Output(Binary)_Output(Decimal)");
15     $display("-----");
16     A = 2'b00; #10;
17     $display("%b_ddd|ddd%d_ddd%b_ddd%d", A, A, out, out);
18     A = 2'b01; #10;
19     $display("%b_ddd|ddd%d_ddd%b_ddd%d", A, A, out, out);
20     A = 2'b10; #10;
21     $display("%b_ddd|ddd%d_ddd%b_ddd%d", A, A, out, out);
22     A = 2'b11; #10;
23     $display("%b_ddd|ddd%d_ddd%b_ddd%d", A, A, out, out);
24     $finish;
25 end
26 endmodule

```

Output in Terminal

4-to-2 Encoder:

```

1 VCD info: dumpfile assign2_problem2_encoder_4_to_2.vcd opened for output.
2 Kunwar Arpit Singh
3 Output for only the valid cases for 4 to 2 encoder
4 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
5 -----
6 0001 | 1 | 00 | 0
7 0010 | 2 | 01 | 1
8 0100 | 4 | 10 | 2
9 1000 | 8 | 11 | 3
10
11 Output for all the cases for 4 to 2 encoder
12 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
13 -----
14 0000 | 0 | xx | x
15 0001 | 1 | 00 | 0
16 0010 | 2 | 01 | 1
17 0011 | 3 | xx | x
18 0100 | 4 | 10 | 2
19 0101 | 5 | xx | x
20 0110 | 6 | xx | x
21 0111 | 7 | xx | x
22 1000 | 8 | 11 | 3
23 1001 | 9 | xx | x
24 1010 | 10 | xx | x
25 1011 | 11 | xx | x
26 1100 | 12 | xx | x
27 1101 | 13 | xx | x
28 1110 | 14 | xx | x

```

29 | 1111 | 15 | xx | x

2-to-4 Decoder:

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign2_problem2_decoder_2_to_4.vcd opened for output.
3 Kunwar Arpit Singh
4 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
5 -----
6 00 | 0 | 0001 | 1
7 01 | 1 | 0010 | 2
8 10 | 2 | 0100 | 4
9 11 | 3 | 1000 | 8

```

Output Screenshots

Terminal output (4-to-2 Encoder)

```

VCD info: dumpfile assign2_problem2_encoder_4_to_2.vcd opened for output.
Kunwar Arpit Singh
Output for only the valid cases for 4 to 2 encoder:
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
0001 | 1 | 00 | 0
0010 | 2 | 01 | 1
0100 | 4 | 10 | 2
1000 | 8 | 11 | 3

Output for all the cases for 4 to 2 encoder
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
0000 | 0 | xx | x
0001 | 1 | 00 | 0
0010 | 2 | 01 | 1
0011 | 3 | xx | x
0100 | 4 | 10 | 2
0101 | 5 | xx | x
0110 | 6 | xx | x
0111 | 7 | xx | x
1000 | 8 | 11 | 3
1001 | 9 | xx | x
1010 | 10 | xx | x
1011 | 11 | xx | x
1100 | 12 | xx | x
1101 | 13 | xx | x
1110 | 14 | xx | x
1111 | 15 | xx | x

```

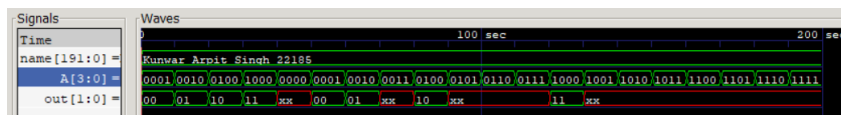
Terminal output (2-to-4 Decoder)

```

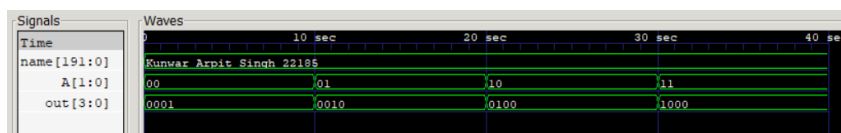
Kunwar Arpit Singh
VCD info: dumpfile assign2_problem2_decoder_2_to_4.vcd opened for output.
Kunwar Arpit Singh
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
00 | 0 | 0001 | 1
01 | 1 | 0010 | 2
10 | 2 | 0100 | 4
11 | 3 | 1000 | 8

```

GTKWave waveform (4-to-2 Encoder)



GTKWave waveform (2-to-4 Decoder)



3. Problem 3

Problem statement

Design an 8-to-1 multiplexer using a case statement in behavioral verilog.

Verilog source (Behavioral)

```
1 module multiplexer_8_to_1(input [7:0] A, input [2:0] sel, output out);
2     assign out = (sel == 3'b000) ? A[0] :
3         (sel == 3'b001) ? A[1] :
4         (sel == 3'b010) ? A[2] :
5         (sel == 3'b011) ? A[3] :
6         (sel == 3'b100) ? A[4] :
7         (sel == 3'b101) ? A[5] :
8         (sel == 3'b110) ? A[6] :
9         (sel == 3'b111) ? A[7] :
10        1'bx;
11 endmodule
```

Testbench

```
1 module multiplexer_8_to_1_tb;
2     reg [7:0] A;
3     reg [2:0] sel;
4     wire out;
5     reg [8*24-1:0] name;
6
7     multiplexer_8_to_1 uut (.A(A), .sel(sel), .out(out));
8
9     initial begin
10         name = "_Kunwar_Arpit_Singh_22185";
11         $display("Kunwar_Arpit_Singh");
12         $dumpfile("assign2_problem3_mutiplexer_8_to_1.vcd");
13         $dumpvars(1, multiplexer_8_to_1_tb);
14         $display("Kunwar_Arpit_Singh");
15         A = 8'b11001010;
16         $display("Output_for_all_the_cases");
17         $display("A_(Binary_&Decimal)_|_Output(Binary)_|_Output(Decimal)");
18         $display("-----");
19         sel = 3'b000; #10;
20         $display("%b_|_|%d_|_|%b_|_|%d_|_|%b_|_|%d", A, A, sel, sel, out
21             , out);
22         sel = 3'b001; #10;
23         $display("%b_|_|%d_|_|%b_|_|%d_|_|%b_|_|%d", A, A, sel, sel, out
24             , out);
25         sel = 3'b010; #10;
26         $display("%b_|_|%d_|_|%b_|_|%d_|_|%b_|_|%d", A, A, sel, sel, out
27             , out);
28         sel = 3'b011; #10;
```



```

26     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
27         , out);
28     sel = 3'b100; #10;
29     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
30         , out);
31     sel = 3'b101; #10;
32     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
33         , out);
34     sel = 3'b110; #10;
35     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
36         , out);
37     sel = 3'b111; #10;
38     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
39         , out);
40     A = 8'b00110101;
41     $display("Output_for_all_the_cases");
42     $display("A(Binary&Decimal)|Output(Binary)|Output(Decimal)");
43     $display("-----");
44     sel = 3'b000; #10;
45     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
46         , out);
47     sel = 3'b001; #10;
48     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
49         , out);
50     sel = 3'b010; #10;
51     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
52         , out);
53     sel = 3'b011; #10;
54     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
55         , out);
56     sel = 3'b100; #10;
57     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
58         , out);
59     sel = 3'b101; #10;
60     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
61         , out);
62     sel = 3'b110; #10;
63     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
64         , out);
65     sel = 3'b111; #10;
66     $display("%b|%d|%b|%d|%b|%d", A, A, sel, sel, out
67         , out);
68     $finish;
69 end
70 endmodule

```

Output in Terminal

```

1 Kunwar Arpit Singh

```

```

2 VCD info: dumpfile assign2_problem3_mutlplexer_8_to_1.vcd opened for output.
3 Kunwar Arpit Singh
4 Output for all the cases
5 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
6 -----
7 11001010 | 202 | 000 | 0 | 0 | 0
8 11001010 | 202 | 001 | 1 | 1 | 1
9 11001010 | 202 | 010 | 2 | 0 | 0
10 11001010 | 202 | 011 | 3 | 1 | 1
11 11001010 | 202 | 100 | 4 | 0 | 0
12 11001010 | 202 | 101 | 5 | 0 | 0
13 11001010 | 202 | 110 | 6 | 1 | 1
14 11001010 | 202 | 111 | 7 | 1 | 1
15 Output for all the cases
16 A (Binary & Decimal) | Output(Binary) | Output(Decimal)
17 -----
18 00110101 | 53 | 000 | 0 | 1 | 1
19 00110101 | 53 | 001 | 1 | 0 | 0
20 00110101 | 53 | 010 | 2 | 1 | 1
21 00110101 | 53 | 011 | 3 | 0 | 0
22 00110101 | 53 | 100 | 4 | 1 | 1
23 00110101 | 53 | 101 | 5 | 1 | 1
24 00110101 | 53 | 110 | 6 | 0 | 0
25 00110101 | 53 | 111 | 7 | 0 | 0

```

Output Screenshots

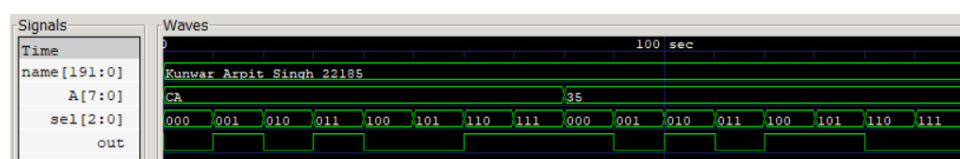
Terminal output (8-to-1 Multiplexer)

```

Kunwar Arpit Singh
VCD info: dumpfile assign2_problem3_mutlplexer_8_to_1.vcd opened for output.
Kunwar Arpit Singh
Output for all the cases
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
11001010 | 202 | 000 | 0 | 0 | 0
11001010 | 202 | 001 | 1 | 1 | 1
11001010 | 202 | 010 | 2 | 0 | 0
11001010 | 202 | 011 | 3 | 1 | 1
11001010 | 202 | 100 | 4 | 0 | 0
11001010 | 202 | 101 | 5 | 0 | 0
11001010 | 202 | 110 | 6 | 1 | 1
11001010 | 202 | 111 | 7 | 1 | 1
Output for all the cases
A (Binary & Decimal) | Output(Binary) | Output(Decimal)
-----
00110101 | 53 | 000 | 0 | 1 | 1
00110101 | 53 | 001 | 1 | 0 | 0
00110101 | 53 | 010 | 2 | 1 | 1
00110101 | 53 | 011 | 3 | 0 | 0
00110101 | 53 | 100 | 4 | 1 | 1
00110101 | 53 | 101 | 5 | 1 | 1
00110101 | 53 | 110 | 6 | 0 | 0
00110101 | 53 | 111 | 7 | 0 | 0

```

GTKWave waveform (8-to-1 Multiplexer)



4. Problem 4

Problem statement

Design an 8-to-33 priority encoder using the if else statement in behavioral Verilog.

Verilog source (Behavioral)

8-to-3 Priority Encoder

```
1 module priority_encoder_8_to_3(input [7:0] A, output reg [2:0] out, output reg
  valid);
2   always @(*) begin
3       valid = |A;
4       if (A[7]) out = 3'b111;
5       else if (A[6]) out = 3'b110;
6       else if (A[5]) out = 3'b101;
7       else if (A[4]) out = 3'b100;
8       else if (A[3]) out = 3'b011;
9       else if (A[2]) out = 3'b010;
10      else if (A[1]) out = 3'b001;
11      else if (A[0]) out = 3'b000;
12      else out = 3'bxxx;
13  end
14 endmodule
```

Testbench

8-to-3 Priority Encoder

```
1 module priority_encoder_8_to_3_tb_small;
2   reg [7:0] A;
3   wire [2:0] out;
4   wire valid;
5   reg[8*24-1:0] name;
6   integer i;
7   priority_encoder_8_to_3 dut (.A(A), .out(out), .valid(valid));
8
9   initial begin
10      name = "_Kunwar_Arpit_Singh_22185";
11      $display("Kunwar_Arpit_Singh");
12      $dumpfile("assign2_problem4_priority_encoder_8_to_3_small.vcd");
13      $dumpvars(1, priority_encoder_8_to_3_tb_small);
14      $display("Kunwar_Arpit_Singh");
15      $display("Outputs_for_Single_Active_Inputs");
16      $display("Output_for_No_Active_Input_is")
17      $display("A(Input)____out(Output)____Valid");
18      A = 8'b00000000; #10; $display("%b_|_|%b_|_|%b_|", A, out, valid);
19      A = 8'b00000001; #10; $display("%b_|_|%b_|_|%b_|", A, out, valid);
20      A = 8'b00000010; #10; $display("%b_|_|%b_|_|%b_|", A, out, valid);
21      A = 8'b00000100; #10; $display("%b_|_|%b_|_|%b_|", A, out, valid);
```

```

22     A = 8'b00001000; #10; $display("%b|_%b|_%b|", A, out, valid);
23     A = 8'b00010000; #10; $display("%b|_%b|_%b|", A, out, valid);
24     A = 8'b00100000; #10; $display("%b|_%b|_%b|", A, out, valid);
25     A = 8'b01000000; #10; $display("%b|_%b|_%b|", A, out, valid);
26     A = 8'b10000000; #10; $display("%b|_%b|_%b|", A, out, valid);
27
28     $display("Outputs_for_Multiple_Active_Inputs");
29     $display("A(Input) | out(Output) | Valid");
30     A = 8'b00000001; #10; $display("%b|_%b|_%b|", A, out, valid);
31     A = 8'b00000011; #10; $display("%b|_%b|_%b|", A, out, valid);
32     A = 8'b00000110; #10; $display("%b|_%b|_%b|", A, out, valid);
33     A = 8'b00001100; #10; $display("%b|_%b|_%b|", A, out, valid);
34     A = 8'b00010001; #10; $display("%b|_%b|_%b|", A, out, valid);
35     A = 8'b00101000; #10; $display("%b|_%b|_%b|", A, out, valid);
36     A = 8'b01010000; #10; $display("%b|_%b|_%b|", A, out, valid);
37     A = 8'b11111111; #10; $display("%b|_%b|_%b|", A, out, valid);
38     $finish;
39 end
40 endmodule

```

Output in Terminal

8-to-3 Priority Encoder

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign2_problem4_priority_encoder_8_to_3_small.vcd opened
  for output.
3 Kunwar Arpit Singh
4 Outputs for Single Active Inputs
5 A(Input) | out(Output) | Valid
6 00000000 | xxx | 0 |
7 00000001 | 000 | 1 |
8 00000010 | 001 | 1 |
9 00000100 | 010 | 1 |
10 00001000 | 011 | 1 |
11 00010000 | 100 | 1 |
12 00100000 | 101 | 1 |
13 01000000 | 110 | 1 |
14 10000000 | 111 | 1 |
15 Outputs for Multiple Active Inputs
16 A(Input) | out(Output) | Valid
17 00000001 | 000 | 1 |
18 00000011 | 001 | 1 |
19 00000110 | 010 | 1 |
20 00001100 | 011 | 1 |
21 00010001 | 100 | 1 |
22 00101000 | 101 | 1 |
23 01010000 | 110 | 1 |
24 11111111 | 111 | 1 |

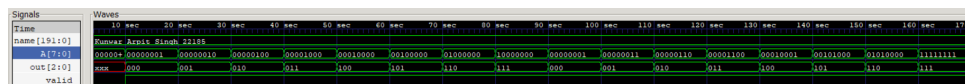
```

Output Screenshots

Terminal output (8-to-3 Priority Encoder)

```
Kunwar Arpit Singh
VCD info: dumpfile assign2_problem4_priority_encoder_8_to_3_small.vcd opened for output.
Kunwar Arpit Singh
Outputs for Single Active Inputs
A(Input) | out(Output) | Valid
00000000 | xxx | 0 |
00000001 | 000 | 1 |
00000010 | 001 | 1 |
00000100 | 010 | 1 |
00001000 | 011 | 1 |
00010000 | 100 | 1 |
00100000 | 101 | 1 |
01000000 | 110 | 1 |
10000000 | 111 | 1 |
Outputs for Multiple Active Inputs
A(Input) | out(Output) | Valid
00000001 | 000 | 1 |
00000011 | 001 | 1 |
00000110 | 010 | 1 |
00001100 | 011 | 1 |
00010001 | 100 | 1 |
00101000 | 101 | 1 |
01010000 | 110 | 1 |
11111111 | 111 | 1 |
```

GTKWave waveform (8-to-3 Priority Encoder)



5. Problem 5

Problem statement

Write behavioral Verilog code for a 4-bit Carry Look-Ahead Adder.

Verilog source (Behavioral)

4-bit Carry Look-Ahead Adder

```
1 module carry_look_ahead_adder_4_bit(input [3:0] A, input [3:0] B, input C_IN,
2   output [3:0] SUM, output C_OUT);
3   wire [3:0] P;
4   wire [3:0] G;
5   wire [3:0] C;
6
7   assign P = A ^ B;
8   assign G = A & B;
9
10  assign C[0] = C_IN;
11  assign C[1] = G[0] | (P[0] & C[0]);
12  assign C[2] = G[1] | (P[1] & C[0]) | (P[1] & P[0] & C[0]);
13  assign C[3] = G[2] | (P[2] & C[1]) | (P[2] & P[1] & G[0]) | (P[2] & P[1] &
14    P[0] & C[0]);
15  assign C_OUT = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1]) | (P[3] & P[2] &
16    P[1] & G[0]) | (P[3] & P[2] & P[1] & P[0] & C[0]);
17
18  assign SUM = P ^ C;
19 endmodule
```

Testbench

4-bit Carry Look-Ahead Adder

```
1 module carry_look_ahead_adder_4_bit_tb;
2   reg [3:0] A;
3   reg [3:0] B;
4   reg C_IN;
5   wire [3:0] SUM;
6   wire C_OUT;
7   reg [8*24-1:0] name;
8   carry_look_ahead_adder_4_bit dut (.A(A), .B(B), .C_IN(C_IN), .SUM(SUM), .
9     C_OUT(C_OUT));
10
11  initial begin
12    name = "KunwarArpitSingh22185";
13
14    $display("KunwarArpitSingh");
15
16    $dumpfile("assign2_problem5_carry_look_ahead_adder_4_bit.vcd");
17    $dumpvars(1, carry_look_ahead_adder_4_bit_tb);
```

```

17
18     $display("Outputs for some sample input (in both Binary and Decimal)");
19
20     $display("A(B) | A(D) | B(D) | B(D) | Cin | Cout | Sum");
21     $display("-----");
22
23     A=4'b0101; B = 4'b0010; C_IN = 1'b0; #10;
24     $display("%04b | %04b | %04b | %04b | %04b | %04b | %04b", A,
25             A, B, B, C_IN, C_IN, C_OUT, C_OUT, SUM, SUM);
26
27     A=4'b1010; B = 4'b1000; C_IN = 1'b0; #10;
28     $display("%04b | %04b | %04b | %04b | %04b | %04b | %04b", A,
29             A, B, B, C_IN, C_IN, C_OUT, C_OUT, SUM, SUM);
30
31     A=4'b1001; B = 4'b0011; C_IN = 1'b1; #10;
32     $display("%04b | %04b | %04b | %04b | %04b | %04b | %04b", A,
33             A, B, B, C_IN, C_IN, C_OUT, C_OUT, SUM, SUM);
34
35     A=4'b1111; B = 4'b0000; C_IN = 1'b1; #10;
36     $display("%04b | %04b | %04b | %04b | %04b | %04b | %04b", A,
37             A, B, B, C_IN, C_IN, C_OUT, C_OUT, SUM, SUM);
38
39     $finish;
40 end
endmodule

```

Output in Terminal

4-bit Carry Look-Ahead Adder

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign2_problem5_carry_look_ahead_adder_4_bit.vcd opened
  for output.
3 Outputs for some sample input (in both Binary and Decimal)
4 A(B) | A(D) | B(D) | B(D) | Cin | Cout | Sum
5 -----
6 0101 | 5 | 0010 | 2 | 0 | 0 | 0 | 0 | 0111 | 7 |
7 1010 | 10 | 1000 | 8 | 0 | 0 | 1 | 1 | 0010 | 2 |
8 1001 | 9 | 0011 | 3 | 1 | 1 | 0 | 0 | 1101 | 13 |
9 1111 | 15 | 0000 | 0 | 1 | 1 | 1 | 1 | 0000 | 0 |
10 1111 | 15 | 1111 | 15 | 1 | 1 | 1 | 1 | 1111 | 15 |

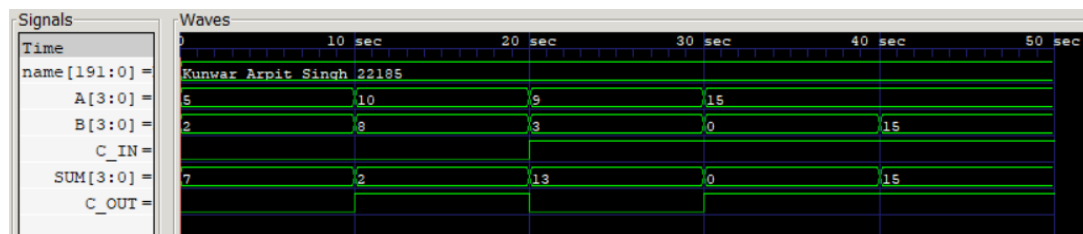
```

Output Screenshots

Terminal output (4-bit Carry Look-Ahead Adder)

```
Kunwar Arpit Singh
VCD info: dumpfile assign2_problem5_carry_look_ahead_adder_4_bit.vcd opened for output.
Outputs for some sample input (in both Binary and Decimal)
A(B) | A(D) | B(D) | B(D) | Cin | Cout | Sum
-----
0101 | 5 | 0010 | 2 | 0 | 0 | 0 | 0111 | 7 |
1010 | 10 | 1000 | 8 | 0 | 0 | 1 | 1 | 0010 | 2 |
1001 | 9 | 0011 | 3 | 1 | 1 | 0 | 0 | 1101 | 13 |
1111 | 15 | 0000 | 0 | 1 | 1 | 1 | 1 | 0000 | 0 |
1111 | 15 | 1111 | 15 | 1 | 1 | 1 | 1 | 1111 | 15 |
```

GTKWave waveform (4-bit Carry Look-Ahead Adder)



6. Problem 6

Problem statement

Design behavioral Verilog code for a 4-bit Wallace Tree Multiplier.

Verilog source (Behavioral)

```
1 module wallace_tree_multiplier_4_bit(input [3:0] A, input [3:0] B, output
  [7:0] out);
2
3   wire PP_00, PP_01, PP_02, PP_03;
4   wire PP_10, PP_11, PP_12, PP_13;
5   wire PP_20, PP_21, PP_22, PP_23;
6   wire PP_30, PP_31, PP_32, PP_33;
7
8   assign PP_00 = A[0] & B[0];
9   assign PP_01 = A[0] & B[1];
10  assign PP_02 = A[0] & B[2];
11  assign PP_03 = A[0] & B[3];
12
13  assign PP_10 = A[1] & B[0];
14  assign PP_11 = A[1] & B[1];
15  assign PP_12 = A[1] & B[2];
16  assign PP_13 = A[1] & B[3];
17
18  assign PP_20 = A[2] & B[0];
19  assign PP_21 = A[2] & B[1];
20  assign PP_22 = A[2] & B[2];
21  assign PP_23 = A[2] & B[3];
22
23  assign PP_30 = A[3] & B[0];
24  assign PP_31 = A[3] & B[1];
25  assign PP_32 = A[3] & B[2];
26  assign PP_33 = A[3] & B[3];
27
28  wire S1_1, C1_1;
29  wire S2_1, C2_1;
30  wire S3_1, C3_1;
31  wire S4_1, C4_1;
32  wire S5_1, C5_1;
33  wire S6_1, C6_1;
34
35  wire S2_2, C2_2;
36  wire S3_2, C3_2;
37  wire S4_2, C4_2;
38  wire S5_2, C5_2;
39  wire C6_sum_final, C6_carry_final;
40  wire S3_temp, C3_temp, C3_temp2, C_to_C5;
41
42  assign S1_1 = PP_01 ^ PP_10;
```

```

43  assign C1_1 = PP_01 & PP_10;
44
45  assign S2_1 = PP_02 ^ PP_11 ^ PP_20;
46  assign C2_1 = (PP_02 & PP_11) | (PP_11 & PP_20) | (PP_02 & PP_20);
47
48  assign S3_temp = PP_03 ^ PP_12 ^ PP_21;
49  assign C3_temp = (PP_03 & PP_12) | (PP_12 & PP_21) | (PP_03 & PP_21);
50  assign S3_1 = S3_temp ^ PP_30;
51  assign C3_temp2 = S3_temp & PP_30;
52  assign C3_1 = C3_temp ^ C3_temp2;
53  assign C_to_C5 = C3_temp & C3_temp2;
54
55  assign S4_1 = PP_13 ^ PP_22 ^ PP_31;
56  assign C4_1 = (PP_13 & PP_22) | (PP_22 & PP_31) | (PP_13 & PP_31);
57
58  assign S5_1 = PP_23 ^ PP_32;
59  assign C5_1 = PP_23 & PP_32;
60
61  assign S6_1 = PP_33;
62  assign C6_1 = 1'b0;
63
64  assign S2_2 = S2_1 ^ C1_1;
65  assign C2_2 = S2_1 & C1_1;
66
67  assign S3_2 = S3_1 ^ C2_1;
68  assign C3_2 = S3_1 & C2_1;
69
70  assign S4_2 = S4_1 ^ C3_1;
71  assign C4_2 = S4_1 & C3_1;
72
73  assign S5_2 = S5_1 ^ C4_1;
74  assign C5_2 = S5_1 & C4_1;
75
76  assign C6_sum_final = C5_1 ^ C5_2;
77  assign C6_carry_final = C5_1 & C5_2;
78
79  wire [7:0] VEC1, VEC2;
80
81  assign VEC1[0] = PP_00;
82  assign VEC1[1] = S1_1;
83  assign VEC1[2] = S2_2;
84  assign VEC1[3] = S3_2;
85  assign VEC1[4] = S4_2;
86  assign VEC1[5] = S5_2;
87  assign VEC1[6] = S6_1;
88  assign VEC1[7] = C6_carry_final;
89
90  assign VEC2[0] = 1'b0;
91  assign VEC2[1] = 1'b0;
92  assign VEC2[2] = 1'b0;
93  assign VEC2[3] = C2_2;

```

```

94     assign VEC2[4] = C3_2;
95     assign VEC2[5] = C4_2 | C_to_C5;
96     assign VEC2[6] = C6_sum_final;
97     assign VEC2[7] = 1'b0;
98
99     assign out = VEC1 + VEC2;
100
101 endmodule

```

Testbench

```

1 module wallace_tree_multiplier_4_bit_tb;
2     reg [3:0] A;
3     reg [3:0] B;
4     wire [7:0] out;
5     reg [8*24-1:0] name;
6
7     wallace_tree_multiplier_4_bit dut (.A(A), .B(B), .out(out));
8
9     initial begin
10         name = "_Kunwar_Arpit_Singh_22185";
11         A = 0;
12         B = 0;
13
14         $display("Kunwar_Arpit_Singh");
15         $dumpfile("assign2_problem6_wallace_tree_multiplier_4_bit.vcd");
16         $dumpvars(1, wallace_tree_multiplier_4_bit_tb);
17         $display("Kunwar_Arpit_Singh");
18         $display("|_A(B)_|_A(D)_|_B(B)_|_B(D)_|_Product(B)_|_Product(D)_|");
19
20         A = 4'b0001; B = 4'b0001; #10;
21         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
22
23         A = 4'b0101; B = 4'b1010; #10;
24         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
25
26         A = 4'b1111; B = 4'b1111; #10;
27         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
28
29         A = 4'b1001; B = 4'b0111; #10;
30         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
31
32         A = 4'b1100; B = 4'b0011; #10;
33         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
34
35         A = 4'b0100; B = 4'b1111; #10;
36         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);
37
38         A = 4'b0000; B = 4'b1111; #10;
39         $display("|_b_|_d_|_b_|_d_|_b_|_d_|", A, A, B, B, out, out);

```

```

40
41         $finish;
42     end
43 endmodule

```

Output in Terminal

```

1 Kunwar Arpit Singh
2 VCD info: dumpfile assign2_problem6_wallace_tree_multiplier_4_bit.vcd opened
  for output.
3 Kunwar Arpit Singh
4 | A(B) | A(D) | B(B) | B(D) | Product(B) | Product(D) |
5 | 0001 | 1 | 0001 | 1 | 00000001 | 1 |
6 | 0101 | 5 | 1010 | 10 | 00110010 | 50 |
7 | 1111 | 15 | 1111 | 15 | 11100001 | 225 |
8 | 1001 | 9 | 0111 | 7 | 00111111 | 63 |
9 | 1100 | 12 | 0011 | 3 | 00100100 | 36 |
10 | 0100 | 4 | 1111 | 15 | 00111100 | 60 |
11 | 0000 | 0 | 1111 | 15 | 00000000 | 0 |

```

Output Screenshots

Terminal output (4-bit Wallace Tree Multiplier)

```

Kunwar Arpit Singh
VCD info: dumpfile assign2_problem6_wallace_tree_multiplier_4_bit.vcd opened for output.
Kunwar Arpit Singh
| A(B) | A(D) | B(B) | B(D) | Product(B) | Product(D) |
| 0001 | 1 | 0001 | 1 | 00000001 | 1 |
| 0101 | 5 | 1010 | 10 | 00110010 | 50 |
| 1111 | 15 | 1111 | 15 | 11100001 | 225 |
| 1001 | 9 | 0111 | 7 | 00111111 | 63 |
| 1100 | 12 | 0011 | 3 | 00100100 | 36 |
| 0100 | 4 | 1111 | 15 | 00111100 | 60 |
| 0000 | 0 | 1111 | 15 | 00000000 | 0 |

```

GTKWave waveform (4-bit Wallace Tree Multiplier)

