

Name: - Meet Soni

ID No. :- 21EL035

Division :- 04

Year :- 2023-24

Subject:- Digital System Design (3EL42)

Branch :- Electronics (EL)

Assignment :- 1

Q-1 :-Write a Verilog code for 2*4 decoder.

Dataflow:-

```
4 module decoder_2x4_dataflow (
5     input wire enable,
6     input wire [1:0] select,
7     output wire [3:0] output
8 );
9
10    assign output[0] = (enable && (select == 2'b00)) ? 1'b1 : 1'b0;
11    assign output[1] = (enable && (select == 2'b01)) ? 1'b1 : 1'b0;
12    assign output[2] = (enable && (select == 2'b10)) ? 1'b1 : 1'b0;
13    assign output[3] = (enable && (select == 2'b11)) ? 1'b1 : 1'b0;
14
15 endmodule
16
```

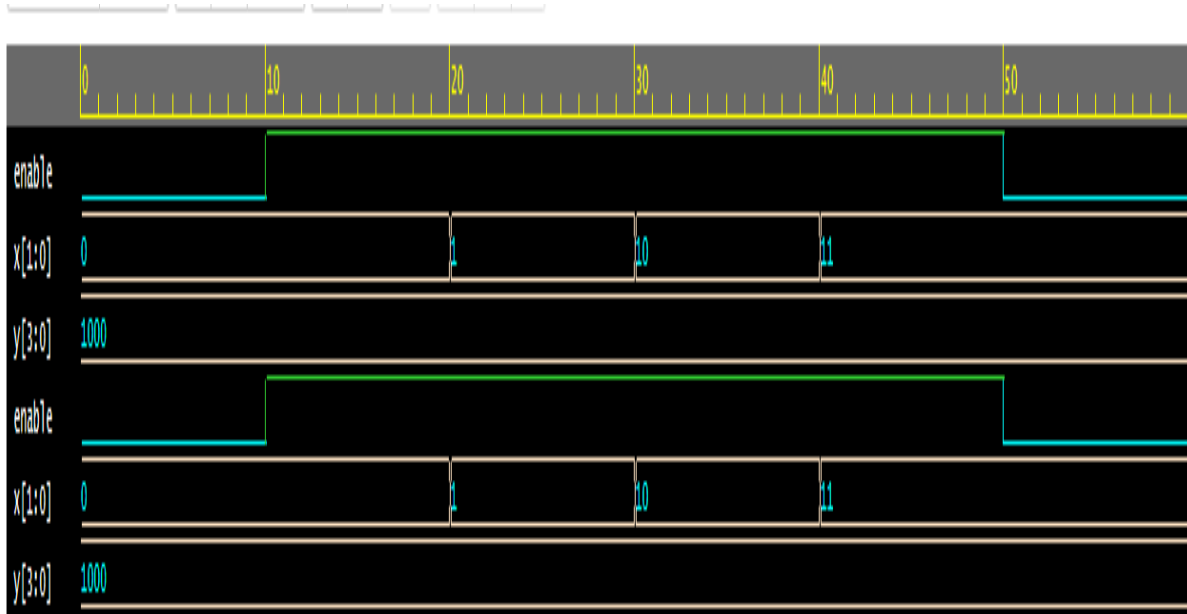
Behavioral:-

```
4 module decoder_2x4_behavioral (
5     input wire enable,
6     input wire [1:0] select,
7     output wire [3:0] output
8 );
9
10    always @(enable or select) begin
11        case (select)
12            2'b00: output = enable ? 4'b0001 : 4'b0000;
13            2'b01: output = enable ? 4'b0010 : 4'b0000;
14            2'b10: output = enable ? 4'b0100 : 4'b0000;
15            2'b11: output = enable ? 4'b1000 : 4'b0000;
16            default: output = 4'b0000;
17        endcase
18    end
19
20 endmodule
21
```

Structural :-

```
4 module AND2 (
5     input wire a,
6     input wire b,
7     output wire y
8 );
9     assign y = a & b;
10 endmodule
11
12 module NOT (
13     input wire a,
14     output wire y
15 );
16     assign y = ~a;
17 endmodule
18
19 module decoder_2x4_structural (
20     input wire enable,
21     input wire [1:0] select,
22     output wire [3:0] output
23 );
24
25     wire w_not_sel0, w_not_sel1;
26
27     NOT not_sel0 (.a(select[0]), .y(w_not_sel0));
28     NOT not_sel1 (.a(select[1]), .y(w_not_sel1));
29
30     AND2 and00 (.a(enable), .b(w_not_sel1), .y(output[0]));
31     AND2 and01 (.a(enable), .b(w_not_sel0), .y(output[1]));
32     AND2 and10 (.a(enable), .b(select[1]), .y(output[2]));
33     AND2 and11 (.a(enable), .b(select[0]), .y(output[3]));
34
35 endmodule
36
```

Output:-



Q-2 :- Write a Verilog code for full subtractor.

Dataflow:-

```
4 module full_subtractor_dataflow (
5     input wire A, B, Bin,
6     output wire Diff, Bout, Borr
7 );
8     assign {Borr, Diff, Bout} = A ^ B ^ Bin;
9 endmodule
```

Behavioural :-

```
4 module full_subtractor_behavioral (
5     input wire A, B, Bin,
6     output wire Diff, Bout, Borr
7 );
8     assign Diff = (A ^ B) ^ Bin;
9     assign Bout = (~A & B) | (Bin & (~A ^ B));
10    assign Borr = (~A & B) | (Bin & ~B);
11 endmodule
```

Structural :

```
4 module XOR2 (  
5     input wire a,  
6     input wire b,  
7     output wire y  
8 );  
9     assign y = a ^ b;  
10 endmodule  
11  
12 module AND2 (  
13     input wire a,  
14     input wire b,  
15     output wire y  
16 );  
17     assign y = a & b;  
18 endmodule  
19  
20 module NOT (  
21     input wire a,  
22     output wire y  
23 );  
24     assign y = ~a;  
25 endmodule  
26  
27 module full_subtractor_structural (  
28     input wire A, B, Bin,  
29     output wire Diff, Bout, Borr  
30 );  
31     wire x1, x2, x3, x4, x5, x6;  
32     XOR2 xor1 (.a(A), .b(B), .y(x1));  
33     XOR2 xor2 (.a(x1), .b(Bin), .y(Diff));  
34     AND2 and1 (.a(~A), .b(B), .y(x2));  
35     AND2 and2 (.a(x2), .b(Bin), .y(x3));  
36     AND2 and3 (.a(~A), .b(Bin), .y(x4));  
37     AND2 and4 (.a(B), .b(~Bin), .y(x5));  
38     OR2 or1 (.a(x3), .b(x5), .y(Bout));  
39     OR2 or2 (.a(x4), .b(x5), .y(Borr));  
40 endmodule  
41 |
```

Output:-

```
[2023-08-23 16:18:53 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
At time 0: a=0 b=0, Bin=0, difference=0, borrow=0
```

```
At time 1: a=0 b=0, Bin=1, difference=1, borrow=1
```

```
At time 2: a=0 b=1, Bin=0, difference=1, borrow=1
```

```
At time 3: a=0 b=1, Bin=1, difference=0, borrow=1
```

```
At time 4: a=1 b=0, Bin=0, difference=1, borrow=0
```

```
At time 5: a=1 b=0, Bin=1, difference=0, borrow=0
```

```
At time 6: a=1 b=1, Bin=0, difference=0, borrow=0
```

```
At time 7: a=1 b=1, Bin=1, difference=1, borrow=1
```

Q-3 :- Write a Verilog code for 2-bit comparator.

Dataflow:-

```
module twobitcomparator(xgtyin,xltyin,x,y,xgty,xlty,xety);  
    //I/O  
    output xgty, xety, xlty;  
    input  xgtyin, xltyin;  
    input [1:0] x,y;  
  
    assign xgty = xgtyin | (~xltyin & ((x[1] > y[1]) | ((x[1] == y[1]) &  
(x[0] > y[0]))));  
    assign xlty = xltyin | (~xgtyin & ((x[1] < y[1]) | ((x[1] == y[1]) &  
(x[0] < y[0]))));  
  
    assign xety = ~(xlty | xgty);  
  
endmodule
```

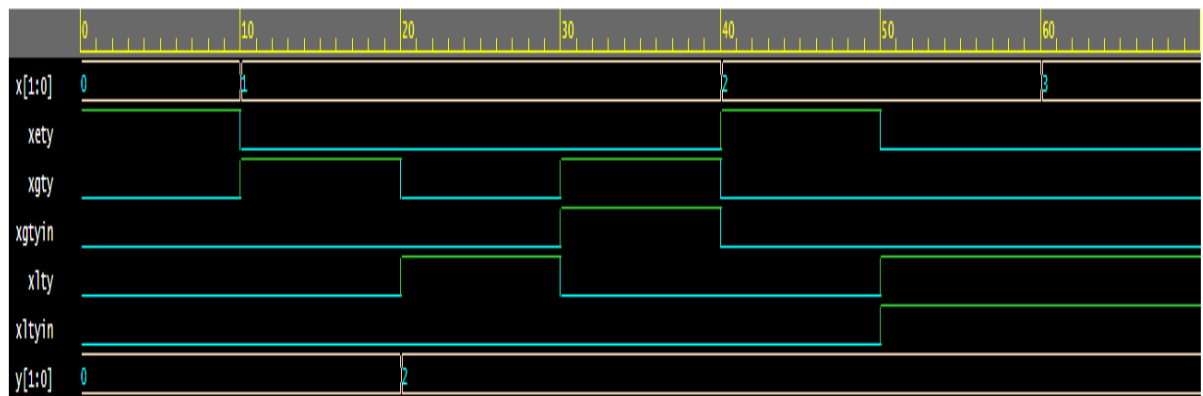
Behavioural :-

```
4 module comparator_2bit_behavioral (  
5     input wire [1:0] A,  
6     input wire [1:0] B,  
7     output wire equal,  
8     output wire A_greater,  
9     output wire B_greater  
10 );  
11     assign equal = (A[1:0] == B[1:0]);  
12     assign A_greater = (A[1] && !B[1]) || (A[1] == B[1] && A[0] > B[0]);  
13     assign B_greater = (B[1] && !A[1]) || (B[1] == A[1] && B[0] > A[0]);  
14 endmodule
```


Structural :-

```
4 module AND2 (
5     input wire a,
6     input wire b,
7     output wire y
8 );
9     assign y = a & b;
10 endmodule
11
12 module OR2 (
13     input wire a,
14     input wire b,
15     output wire y
16 );
17     assign y = a | b;
18 endmodule
19
20 module NOT (
21     input wire a,
22     output wire y
23 );
24     assign y = ~a;
25 endmodule
26
27 module comparator_2bit_structural (
28     input wire [1:0] A,
29     input wire [1:0] B,
30     output wire equal,
31     output wire A_greater,
32     output wire B_greater
33 );
34     wire a0_b0, a1_b1, a0_b0_not, a1_b1_not, a0_eq_b0, a1_eq_b1;
35
36     XOR2 xor0 (.a(A[0]), .b(B[0]), .y(a0_b0));
37     XOR2 xor1 (.a(A[1]), .b(B[1]), .y(a1_b1));
38     NOT not0 (.a(a0_b0), .y(a0_b0_not));
39     NOT not1 (.a(a1_b1), .y(a1_b1_not));
40     AND2 and0 (.a(a0_b0_not), .b(a1_b1_not), .y(equal));
41     AND2 and1 (.a(a0_b0_not), .b(A[1]), .y(a0_eq_b0));
42     AND2 and2 (.a(a1_b1_not), .b(A[0]), .y(a1_eq_b1));
43     OR2 or0 (.a(a0_eq_b0), .b(a1_eq_b1), .y(A_greater));
44     OR2 or1 (.a(a0_b0), .b(a1_b1_not), .y(B_greater));
45 endmodule
46
```

Output:-



Q-4:- Write a Verilog code for 3 bit binary to gray convertor.

Dataflow:-

```
4 module bin_to_gray_converter(  
5     input wire [2:0] binary,  
6     output wire [2:0] gray  
7 );  
8     assign gray[2] = binary[2];  
9     assign gray[1] = binary[2] ^ binary[1];  
10    assign gray[0] = binary[1] ^ binary[0];  
11 endmodule  
12  
13 module comparator_bin_to_gray_dataflow (  
14     input wire [2:0] A,  
15     input wire [2:0] B,  
16     output wire equal,  
17     output wire A_greater,  
18     output wire B_greater  
19 );  
20  
21     wire [2:0] gray_A, gray_B;  
22  
23     bin_to_gray_converter bin_to_gray_A (.binary(A), .gray(gray_A));  
24     bin_to_gray_converter bin_to_gray_B (.binary(B), .gray(gray_B));  
25  
26     assign equal = (gray_A == gray_B);  
27     assign A_greater = (gray_A > gray_B);  
28     assign B_greater = (gray_A < gray_B);  
29 endmodule  
30 |
```

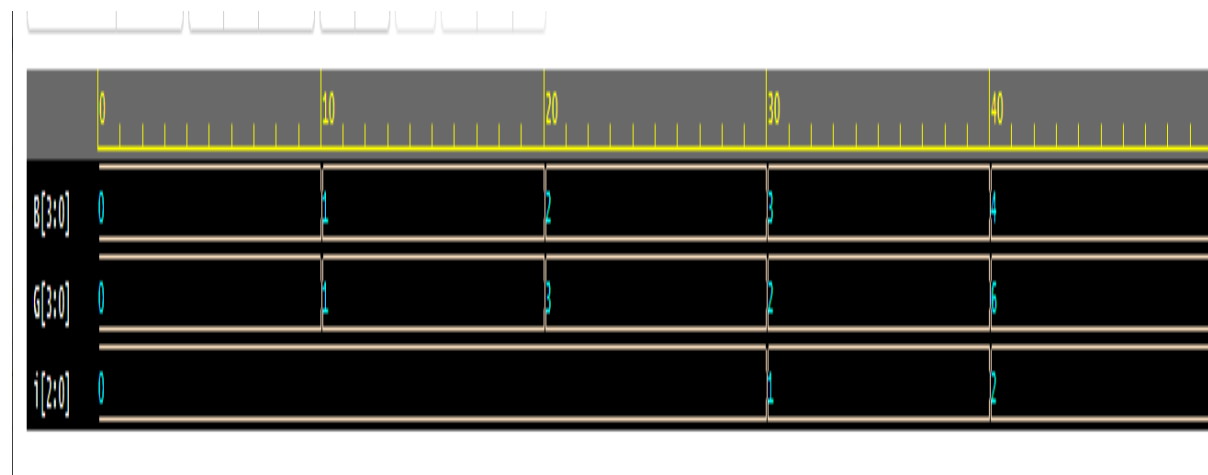
Behavioural :-

```
4 module comparator_bin_to_gray_behavioral (  
5     input wire [2:0] A,  
6     input wire [2:0] B,  
7     output wire equal,  
8     output wire A_greater,  
9     output wire B_greater  
10 );  
11  
12     wire [2:0] gray_A, gray_B;  
13  
14     assign gray_A[2] = A[2];  
15     assign gray_A[1] = A[2] ^ A[1];  
16     assign gray_A[0] = A[1] ^ A[0];  
17  
18     assign gray_B[2] = B[2];  
19     assign gray_B[1] = B[2] ^ B[1];  
20     assign gray_B[0] = B[1] ^ B[0];  
21  
22     assign equal = (gray_A == gray_B);  
23     assign A_greater = (gray_A > gray_B);  
24     assign B_greater = (gray_A < gray_B);  
25 endmodule  
26
```

Structural :-

```
4 module XOR2 (
5     input wire a,
6     input wire b,
7     output wire y
8 );
9     assign y = a ^ b;
10 endmodule
11
12 module bin_to_gray_converter(
13     input wire [2:0] binary,
14     output wire [2:0] gray
15 );
16     wire g1, g2;
17
18     assign gray[2] = binary[2];
19     XOR2 xor1 (.a(binary[2]), .b(binary[1]), .y(g1));
20     XOR2 xor2 (.a(binary[1]), .b(binary[0]), .y(g2));
21     assign gray[1] = g1;
22     assign gray[0] = g2;
23 endmodule
24
25 module comparator_bin_to_gray_structural (
26     input wire [2:0] A,
27     input wire [2:0] B,
28     output wire equal,
29     output wire A_greater,
30     output wire B_greater
31 );
32
33     wire [2:0] gray_A, gray_B;
34
35     bin_to_gray_converter bin_to_gray_A (.binary(A), .gray(gray_A));
36     bin_to_gray_converter bin_to_gray_B (.binary(B), .gray(gray_B));
37
38     wire [2:0] xor_outputs_A, xor_outputs_B;
39
40     XOR2 xor0_A (.a(gray_A[2]), .b(gray_B[2]), .y(xor_outputs_A[2]));
41     XOR2 xor1_A (.a(gray_A[1]), .b(gray_B[1]), .y(xor_outputs_A[1]));
42     XOR2 xor2_A (.a(gray_A[0]), .b(gray_B[0]), .y(xor_outputs_A[0]));
43
44     XOR2 xor0_B (.a(gray_B[2]), .b(gray_A[2]), .y(xor_outputs_B[2]));
45     XOR2 xor1_B (.a(gray_B[1]), .b(gray_A[1]), .y(xor_outputs_B[1]));
46     XOR2 xor2_B (.a(gray_B[0]), .b(gray_A[0]), .y(xor_outputs_B[0]));
47
48     assign equal = ~(xor_outputs_A[2] & xor_outputs_A[1] & xor_outputs_A[0] & xor_outputs_B[2] & xor_outputs_B[1] & xor_outputs_B[0]);
49     assign A_greater = (xor_outputs_A[2] & ~xor_outputs_A[1] & ~xor_outputs_A[0]) | (xor_outputs_A[1] & ~xor_outputs_A[0] & ~xor_outputs_B[2]);
50     assign B_greater = (xor_outputs_B[2] & ~xor_outputs_B[1] & ~xor_outputs_B[0]) | (xor_outputs_B[1] & ~xor_outputs_B[0] & ~xor_outputs_A[2]);
51 endmodule
52
```

Output:-



Q-5 :- Write a Verilog code for BCD to excess 3 convertors.

Dataflow :-

```
4 module bcd_to_xs3_dataflow (
5     input wire [3:0] bcd,
6     output wire [3:0] xs3
7 );
8     assign xs3[3] = 1'b0;
9     assign xs3[2] = bcd[3] ^ 1'b1;
10    assign xs3[1] = bcd[2] ^ 1'b1;
11    assign xs3[0] = bcd[1] ^ 1'b1;
12 endmodule
13
```

Behavioural :-

```
4 module bcd_to_xs3_behavioral (
5     input wire [3:0] bcd,
6     output wire [3:0] xs3
7 );
8     assign xs3[3] = 1'b0;
9     assign xs3[2] = ~bcd[3];
10    assign xs3[1] = ~bcd[2];
11    assign xs3[0] = ~bcd[1];
12 endmodule
13
```

Structural:-

```
4 module NOT (
5     input wire a,
6     output wire y
7 );
8     assign y = ~a;
9 endmodule
10
11 module bcd_to_xs3_structural (
12     input wire [3:0] bcd,
13     output wire [3:0] xs3
14 );
15     wire not_bcd3, not_bcd2, not_bcd1;
16
17     NOT not_b3 (.a(bcd[3]), .y(not_bcd3));
18     NOT not_b2 (.a(bcd[2]), .y(not_bcd2));
19     NOT not_b1 (.a(bcd[1]), .y(not_bcd1));
20
21     assign xs3[3] = 1'b0;
22     assign xs3[2] = not_bcd3;
23     assign xs3[1] = not_bcd2;
24     assign xs3[0] = not_bcd1;
25 endmodule
26 |
```

Output:-

	0	10	20	30	40
b[3:0]	0	1	10	11	100
e[3:0]	11	100	101	110	111
b[3:0]	0	1	10	11	100
e[3:0]	11	100	101	110	111