**Name: -** Rohit Dilip Baruah

**ID NO: -** 21EL015

**Division: -** 04

**Year: -** 2023-24

**Subject: -** Digital System Design (3EL42)

**Branch: -** Electronics

1. Write a Verilog code for 2X4 decoder.

**DATA FLOW MODELLING:**

module decoder(d0,d1,d2,d3,e,a,b);

input e,a,b; output d0,d1,d2,d3; assign d0=e&~a&~b; assign d1=e&~a&b; assign d2=e&a&~b; assign d3=e&a&b;

endmodule

# //testbench//

module testbench(); reg t\_e, t\_a, t\_b;

wire t\_d0, t\_d1, t\_d2, t\_d3;

decoder dut(.e(t\_e), .a(t\_a), .b(t\_b), .d0(t\_d0), .d1(t\_d1),

.d2(t\_d2), .d3(t\_d3));

initial begin written by 21EL015

t\_e = 0; t\_a = 1; t\_b = 1;

#100;

t\_e = 1; t\_a = 0; t\_b = 0;

#100;

t\_e = 1; t\_a = 0; t\_b = 1;

#100;

t\_e = 1; t\_a = 1; t\_b = 0;

#100;

t\_e = 1; t\_a = 1; t\_b = 1; $finish(); end

endmodule

**BEHAVIORAL MODELLING:**

module decoder(d0,d1,d2,d3,e,a,b);

input e,a,b; output d0,d1,d2,d3; reg d0,d1,d2,d3; always @(e,a,b) begin d0 = e & ~ a & ~ b; d1 = e & ~ a & b; d2 = e & a & ~ b; d3 = e & a & b; end

endmodule

written by 21EL015

//testbench// //verilog code module decoder(d0,d1,d2,d3,e,a,b); input e, a, b; output d0, d1, d2, d3; assign d0 = e & ~a & ~b; assign d1 = e & ~a & b; assign d2 = e & a & ~b;

assign d3 = e & a & b; endmodule

# //testbench//

module testbench(); reg t\_e, t\_a, t\_b;

wire t\_d0, t\_d1, t\_d2, t\_d3;

decoder dut(.e(t\_e), .a(t\_a), .b(t\_b), .d0(t\_d0), .d1(t\_d1),

.d2(t\_d2), .d3(t\_d3));

initial begin

t\_e = 0; t\_a = 1; t\_b = 1;

#100;

t\_e = 1; t\_a = 0; t\_b = 0;

#100;

t\_e = 1; t\_a = 0; t\_b = 1;

#100;

t\_e = 1; t\_a = 1; t\_b = 0;

#100;

t\_e = 1; t\_a = 1; t\_b = 1;

written by 21EL015

$finish(); end

endmodule

**GATE LEVEL MODELLING:**

module decoder(d0,d1,d2,d3,a,b); input a,b; output d0,d1,d2,d3; wire w1,w2; not g1(w1,a); not g2(w2,b); and g3(d0,w1,w2);

and g4(d1,w1,b); and g5(d2,a,w2);

and g6(d3,a,b); endmodule

//testbench// module testbench();

reg a, b;

wire d0, d1, d2, d3;

decoder dut(.a(a), .b(b), .d0(d0), .d1(d1), .d2(d2), .d3(d3));

initial begin

$dumpfile("testbench.vcd");

$dumpvars(0, testbench);

// Test case 1 a = 0; b = 0;

#10;

// Test case 2 a = 0; b = 1;

#10;

// Test case 3 a = 1; b = 0;

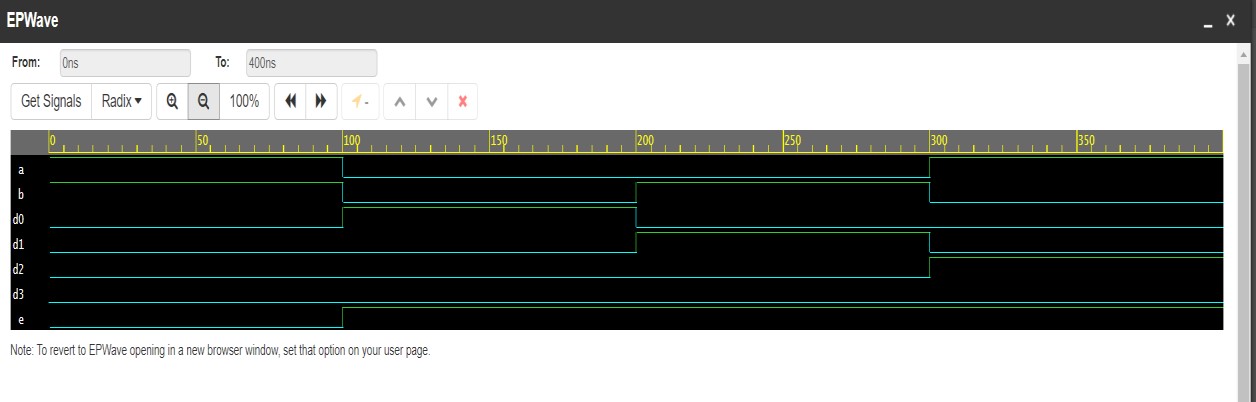
#10;

// Test case 4 a = 1; b = 1; #10;

$finish; end

endmodule

written by 21EL015



2.Write a Verilog code for Full subtractor.

**Dataflow modelling:**

module full\_subtractor( bout, d, a, b, c

); input a, b, c;

output bout, d;

assign d = a ^ b ^ c;

assign bout = (~a & b) || (c & ~(a ^ b)); endmodule

## //Testbench//

module testbench(); reg t\_a, t\_b, t\_c; wire t\_bout, t\_d;

full\_subtractor dut(.a(t\_a), .b(t\_b), .c(t\_c), .bout(t\_bout),

.d(t\_d));

initial begin

$dumpfile("testbench.vcd"); $dumpvars(0, testbench);

end

initial begin

t\_a = 0; t\_b = 0; t\_c = 0; // Test case 1

#10; t\_a = 0; t\_b = 0; t\_c = 1; // Test case 2 #10; t\_a = 0; t\_b = 1; t\_c = 0; // Test case 3

#10;

t\_a = 0; t\_b = 1; t\_c = 1; // Test case 4

#10;

t\_a = 1; t\_b = 0; t\_c = 0; // Test case 5

#10;

t\_a = 1; t\_b = 0; t\_c = 1; // Test case 6

#10;

t\_a = 1; t\_b = 1; t\_c = 0; // Test case 7

#10;

t\_a = 1; t\_b = 1; t\_c = 1; // Test case 8

#10; $finish(); end

endmodule

**Behavioral modelling:**

module full\_subtractor(bout,d,a,b,c);

input a,b,c; output bout,d; wire w1,w2,w3,w4,w5; xor g3(w1,a,b); not g1(w2,a); not g2(w3,w1); and g4(w4,w2,b); and g5(w5,w3,c); xor g6(d,w1,c); or g7(bout,w4,w5); endmodule //Testbench//

module testbench;

reg a, b, c;

wire bout, d;

full\_subtractor dut(

.a(a),

.b(b),

.c(c),

.bout(bout),

.d(d)

);

initial begin

$dumpfile("testbench.vcd");

$dumpvars(0, testbench);

// Test case 1 a = 0; b = 0; c = 0;

#10;

// Test case 2 a = 0; b = 0; c = 1;

#10;

// Test case 3 a = 0; b = 1; c = 0;

#10;

// Test case 4

a = 0; b = 1; c = 1;

#10;

// Test case 5 a = 1; b = 0; c = 0;

#10;

// Test case 6 a = 1; b = 0; c = 1;

#10;

written by 21EL015

// Test case 7 a = 1; b = 1; c = 0;

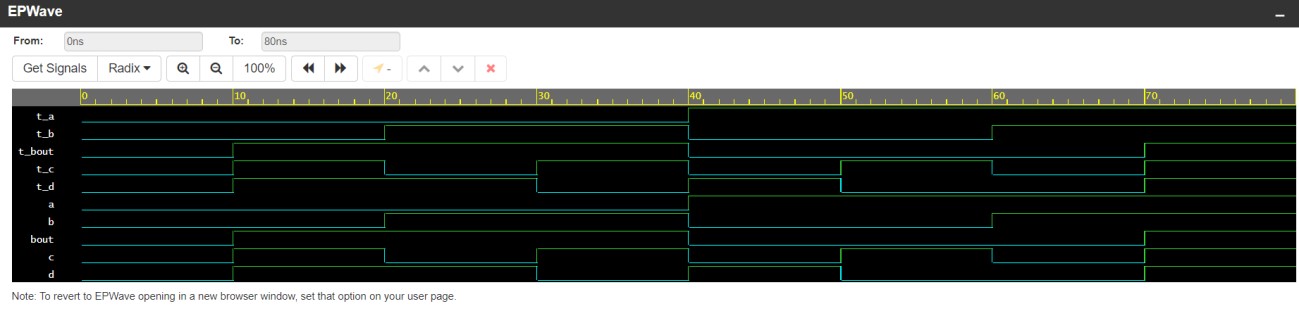
#10;

// Test case 8 a = 1; b = 1; c = 1;

#10;

$finish; end

endmodule



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

**DATAFLOW MODELLING:**

module comparator\_2bit(

input [1:0] A, input [1:0] B, output reg EQ, output reg GT, output reg LT

);

assign EQ = (A[1] & ~B[1] & A[0] & ~B[0]) | (~A[1] & B[1] & ~A[0] & B[0]);

assign GT = (A[1] & ~B[1]) | ((A[1] ^ B[1]) & ~A[0]); assign LT = (B[1] & ~A[1]) | ((A[1] ^ B[1]) & ~B[0]);

endmodule

//Testbench//

module testbench;

reg [1:0] A; reg [1:0] B; wire EQ, GT, LT;

// Instantiate the comparator comparator\_2bit uut (

.A(A),

.B(B),

.EQ(EQ),

.GT(GT),

.LT(LT)

);

initial begin

$dumpfile("testbench.vcd"); $dumpvars(0, testbench); end

// Initialize inputs and monitor outputs initial begin

$monitor("Time = %0t: A = %b, B = %b, EQ = %b, GT = %b, LT = %b", $time, A, B, EQ, GT, LT);

1. = 2'b00;
2. = 2'b01;

#5;

1. = 2'b10;
2. = 2'b01;

#5;

1. = 2'b01;
2. = 2'b01;

#5;

// Add more test cases here

$finish; end

endmodule

**Behavioral modelling:**

module comparator\_2bit(

input [1:0] A, input [1:0] B, output reg EQ, output reg GT, output reg LT

);

always @\* begin

EQ = (A == B);

GT = (A > B); LT = (A < B); end

endmodule

//Testbench//

module testbench;

reg [1:0] A; reg [1:0] B;

wire EQ, GT, LT;

// Instantiate the comparator comparator\_2bit uut (

.A(A),

.B(B),

.EQ(EQ),

.GT(GT),

.LT(LT)

);

initial begin

$dumpfile("testbench.vcd"); $dumpvars(0,testbench);

end

// Initialize inputs and monitor outputs initial begin

$monitor("Time = %0t: A = %b, B = %b, EQ = %b, GT = %b, LT =

%b", $time, A, B, EQ, GT, LT);

1. = 2'b00;
2. = 2'b01;

#5;

1. = 2'b10;
2. = 2'b01;

#5;

1. = 2'b01;
2. = 2'b01;

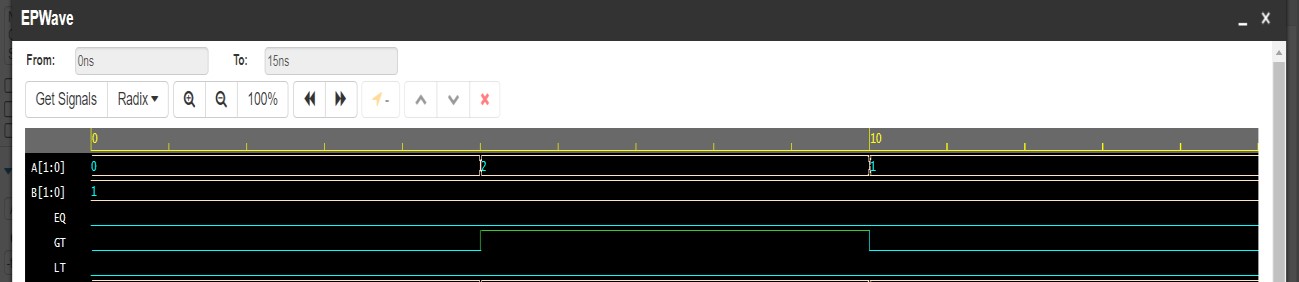
#5;

// Add more test cases here

$finish;

end

endmodule



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

**Dataflow modelling:**

module binary\_to\_gray(g,b); input [2:0]b;

output [2:0]g; assign g[2]=b[2]; assign g[1]=b[2]^b[1]; assign g[0]=b[1]^b[0]; endmodule

//Testbench//

module testbench(); reg [2:0]t\_b; wire [2:0]t\_g; binary\_to\_gray dut(.g(t\_g), .b(t\_b));

initial begin

$dumpfile("testbench.vcd"); $dumpvars(0,testbench);

end initial begin

t\_b[2]=0;t\_b[1]=0;t\_b[0]=0;

#100

t\_b[2]=0;t\_b[1]=0;t\_b[0]=1;

#100

t\_b[2]=0;t\_b[1]=1;t\_b[0]=0;

#100

t\_b[2]=0;t\_b[1]=1;t\_b[0]=1;

#100

t\_b[2]=1;t\_b[1]=0;t\_b[0]=0;

#100

t\_b[2]=1;t\_b[1]=0;t\_b[0]=1;

#100 t\_b[2]=1;t\_b[1]=1;t\_b[0]=0; #100

t\_b[2]=1;t\_b[1]=1;t\_b[0]=1;

#100 $finish(); end

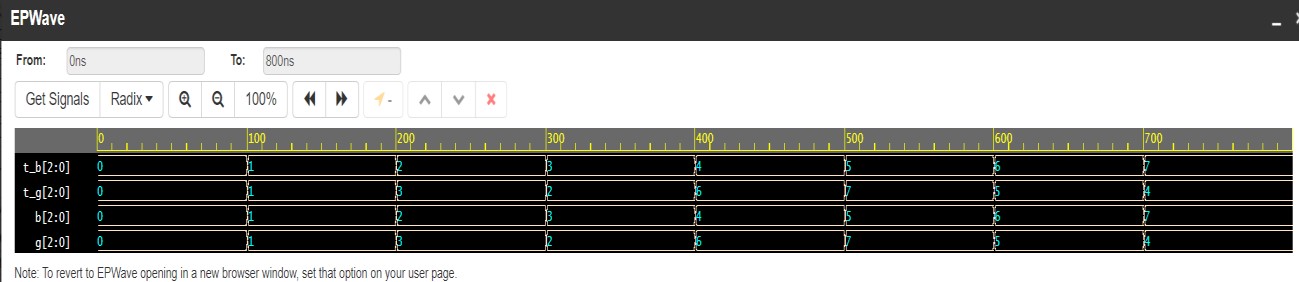
endmodule

**Behavioral modelling:**

module binary\_to\_gray(g,b); input [2:0]b; output [2:0]g; reg [2:0]g;

always @(b[2],b[1],b[0]); initial begin g[2]=b[2]; g[1]=b[2]^b[1]; g[0]=b[1]^b[0]; end

endmodule



5. Write a Verilog code for BCD to excess 3 convertors.**Dataflow modelling:**

module bcd\_to\_excess3(e,b); input [3:0]b; output [3:0]e;

assign e[3]=((b[3])||(b[2]&b[0]||(b[2]&b[1])));

assign e[2]=((b[2]&~b[1]&~b[0])||(~b[2]&b[0])||(~b[2]&b[1])); assign e[1]=(~b[1]&~b[0]||(b[1]&b[0])); assign e[0]=~b[0]; endmodule

## //Testbench//

module testbench;

reg [3:0] b; wire [3:0] e;

// Instantiate the bcd\_to\_excess3 module bcd\_to\_excess3 uut (

.b(b),

.e(e)

);

initial begin

$dumpfile("testbench.vcd"); $dumpvars(0,testbench); end

// Initialize inputs and monitor outputs initial begin

$monitor("Time = %0t: BCD = %b, Excess-3 = %b", $time, b, e);

b = 4'b0000;

#5;

b = 4'b0101;

#5;

b = 4'b1001;

#5;

b = 4'b1111;

#5;

// Add more test cases here

written by 21EL015

$finish; end

endmodule

**Behavioral modelling:**

module bcd\_to\_excess3(

input [3:0] b, output reg [3:0] e

);

always @\* begin

e[3] = (b[3]) || (b[2] & b[0]) || (b[2] & b[1]);

e[2] = (b[2] & ~b[1] & ~b[0]) || (~b[2] & b[0]) || (~b[2] & b[1]); e[1] = (~b[1] & ~b[0]) || (b[1] & b[0]); e[0] = ~b[0]; end

endmodule

