**Name:** Vansh Bhardwaj **Enroll No. :**  E22CSEU0519

**Course-** B. Tech **Type-** Core

**Course Code-** CSET-105 **Course Name-** Digital Design

**Session-** 2022-23 **Semester-** Even

**Date-** 13-17 February 2022 **Batch-** 18

**Lab Assignment 5\_Set 4**

**Practical title:** Implementation of Logic gates

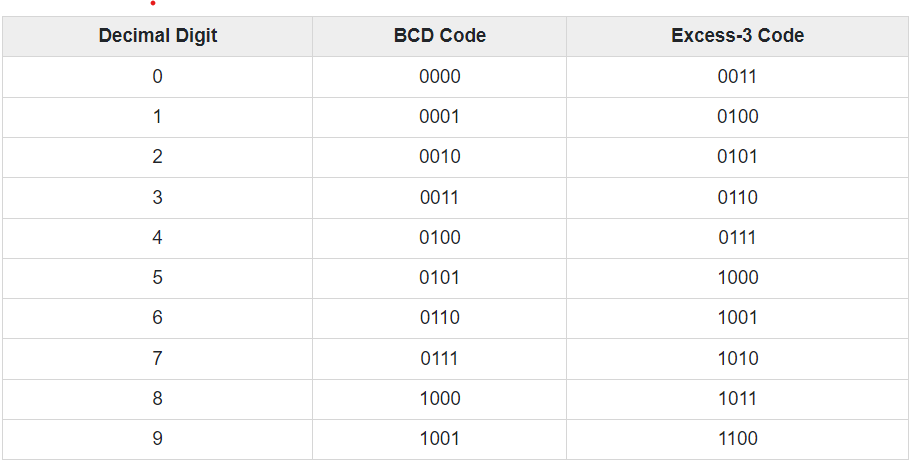
1. **Write the classification of binary codes with appropriate examples. Hint: Weighted, Non-weighted codes, BCD, Ex-3, Gray Code**

**Ans:**

*Weighted Binary Codes:* These codes obey the Positional Weighting Principles. Each number represents a specific weight. The bits are multiplied with indicated weights; the sum gives the decimal equivalent number. E.g.: 8421, 2421 and BCD codes.

*Non-Weighted Binary Codes:* In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and gray code.

*Excess-3 code:* The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding (0011)2 or (3)10 to each code word in 8421. The excess-3 codes are obtained as follows –

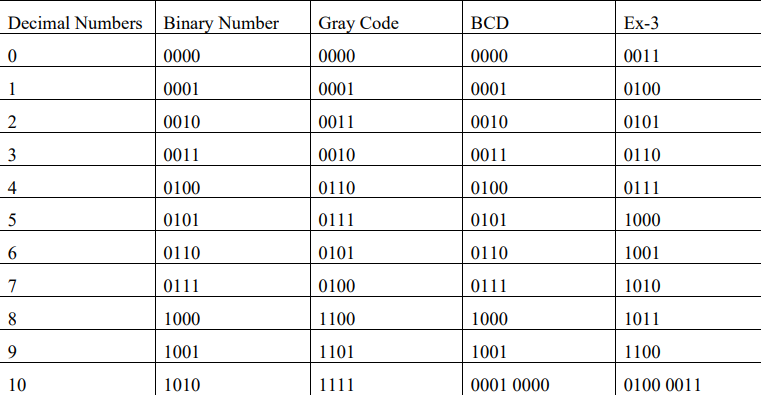
**

*Gray Code:* It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in fig. As only one-bit changes at a time, the gray code is called as a unit distance code. The Gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

**

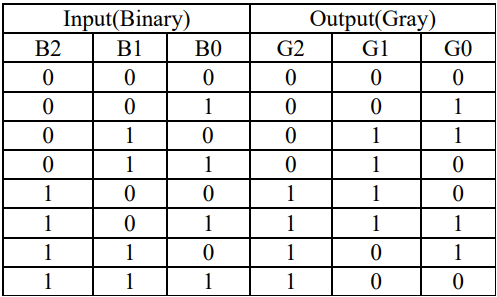
1. **Write the table for conversion of Decimal Numbers from (0-11) to its equivalent Binary, Gray, BCD and Ex-3.**

**Ans:**

****

1. **Write a Verilog code to implement 3-bit binary to Gray Code converter for the circuit as shown in Fig. Write the corresponding Testbench code for the verification of your Verilog code. Hint: Truth Table of 3-bit binary to Gray Code Converter**

**Ans:**

****

*Code:*

//Ikshit\_Mangal\_E22CSEU0519

module bin\_gray(

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:*

//Ikshit\_Mangal\_E22CSEU0519

module tb\_bin\_gray;

reg [2:0]B;

wire [2:0]G;

bin\_gray a1 (.b(B),.g(G));

initial

begin

B[2] = 0; B[1] =0; B[0]=0; #5;

B[2] = 0; B[1] =0; B[0]=1; #5;

B[2] = 0; B[1] =1; B[0]=0; #5;

B[2] = 0; B[1] =1; B[0]=1; #5;

B[2] = 0; B[1] =1; B[0]=1; #5;

B[2] = 1; B[1] =0; B[0]=0; #5;

B[2] = 1; B[1] =0; B[0]=1; #5;

B[2] = 1; B[1] =1; B[0]=0; #5;

B[2] = 1; B[1] =1; B[0]=1; #5;

end

initial

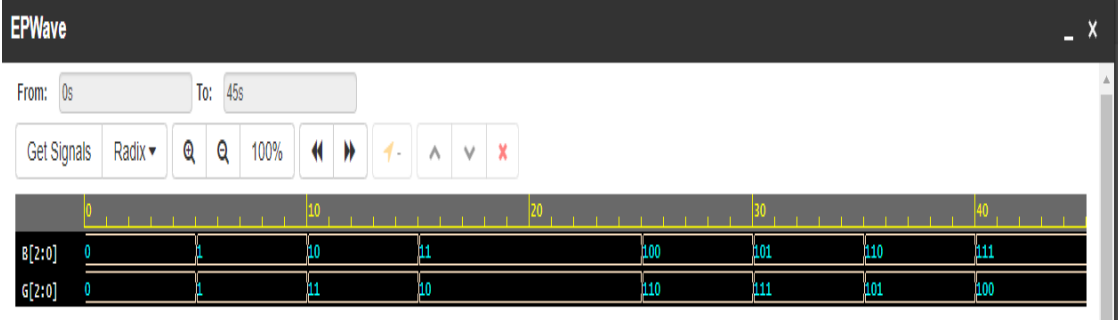
begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule



1. **Draw the logic diagram for the given logical function. Write a Verilog code to implement the following Boolean Expression. Practical Write the corresponding Testbench code for the verification of input combination 0001, 0101 and 1001of your Verilog code. Identify the code converter that matches to this operation.**

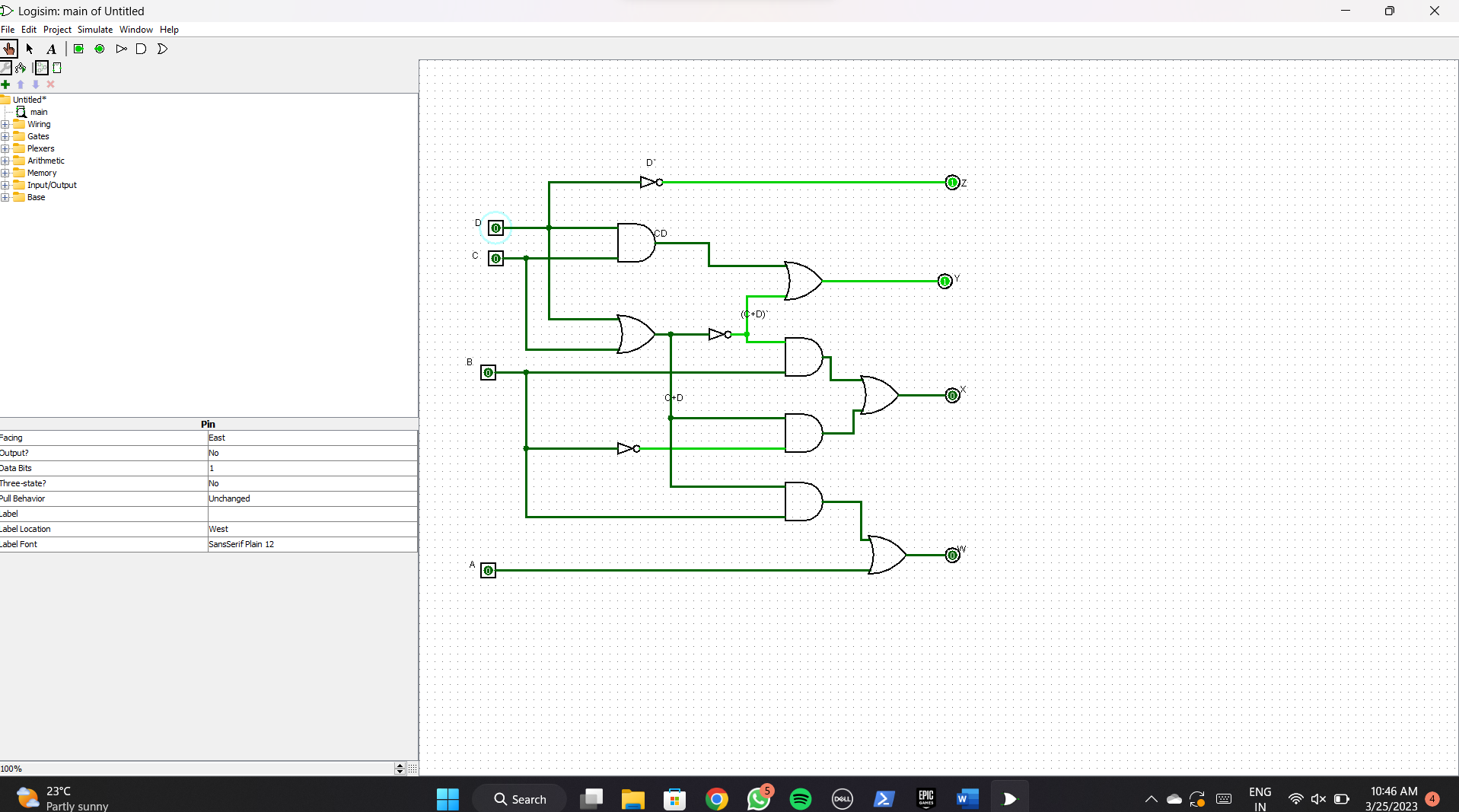
**W = A+BC+BD**

**X=B`C+B`D+BC`D`**

**Y=CD+C`D`**

**Z=D`**

**Ans:**

****

*Code:*

// Ikshit\_Mangal\_E22CSEU0519

module cir1(input a,b,c,d, output w,x,y,z);

assign w=a|b&c|b&d;

assign x=(b&~c&~d)| ~b&c|~b&d;

assign y=~(c^d);

assign z=~d;

endmodule

*TestBench:*

//Ikshit\_Mangal\_E22CSEU0519

module tb();

reg A,B,C,D;

wire W,X,Y,Z;

cir1 c1 (.w(W),.x(X),.y(Y),.z(Z),.a(A),.b(B),.c(C),.d(D));

initial

begin

A =0; B=0; C=0; D=1; #3;

A=0; B=1; C=0; D=1; #3;

A=1; B=0; C=0; D=1; #3;

end

initial

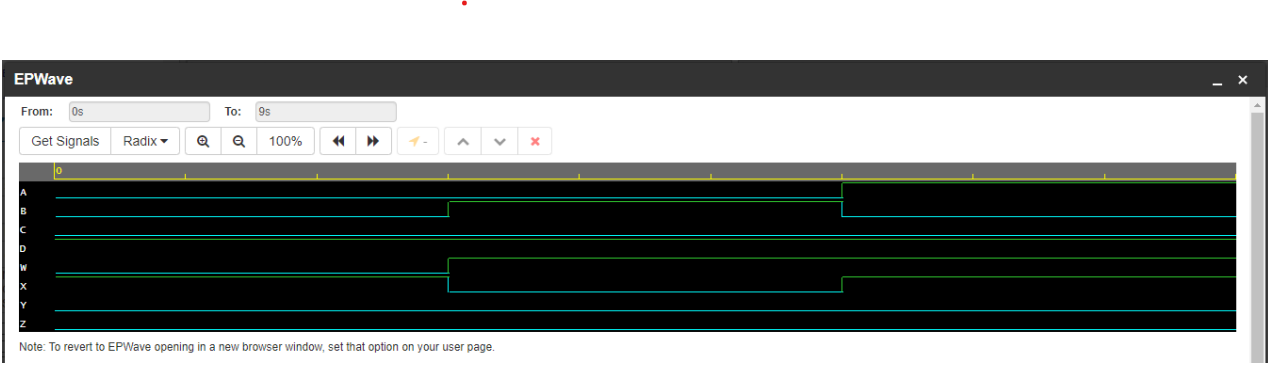
begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

****

1. **Write a Verilog code to implement the following given logical function for Binary to BCD code converter. Write the corresponding Testbench code for the verification of your Verilog code. Given inputs as 1000 and 1100 and 1111.**

**W=AB+AC**

**X=AB`C`**

**Y=A`B+BC**

**Z=ABC`+A`C**

**E=D**

**Ans:**

*Code:*

//Ikshit\_Mangal\_E22CSEU0519

module cir1(input a,b,c,d, output p,q,r,s,t);

assign p=(a&b|a&c);

assign q=a&~b&~c;

assign r=(~a&b|b&c);

assign s=(a&b&~c|~a&c);

assign t=d;

endmodule

*TestBench:*

//Ikshit\_Mangal\_E22CSEU0519

module tb();

reg A,B,C,D;

wire P,Q,R,S,T;

cir1 c1(.a(A),.b(B),.c(C),.d(D),.p(P),.q(Q),.r(R),.s(S),.t(T));

initial

begin

A =1; B=0; C=0; D=0; #3;

A=1; B=1; C=0; D=0; #3;

A=1; B=1; C=1; D=1; #3;

end

initial

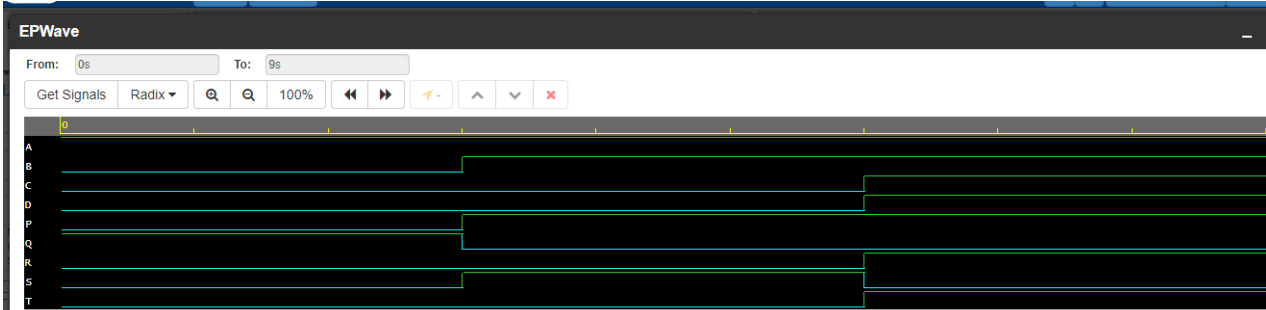
begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

****