**Department of Electrical Engineering**

**Faculty Member:** Arshad Nazir **Dated:** 05/11/2021

**Semester:** 3rd **Section:** BEE 12C

**Group No.: 7**

**EE-221: Digital Logic Design**

Lab 6: Binary to Gray and Gray to Binary Code Conversion

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4/CLO4** | **PLO4/CLO4** | **PLO5/CLO5** | **PLO8/CLO6** | **PLO9/CLO7** |  |
| **Name** | **Reg. No** | **Viva / Lab Performance** | **Analysis of data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** | **Total marks Obtained** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **25 Marks** |
| **Muhammad Adil Azeem** | **342459** |  |  |  |  |  |  |
| **Muhammad Umer** | **345834** |  |  |  |  |  |  |
| **Raahim Ahmad Waqar** | **342287** |  |  |  |  |  |  |
| **Saad Bakhtiar** | **341150** |  |  |  |  |  |  |
| **Syed Aun Ali Kazmi** | **342384** |  |  |  |  |  |  |

## Lab 6: Binary to Gray and Gray to Binary Code Conversion

This Lab has been divided into two parts:

In first part you are required to design and implement a binary to gray and gray to binary code converter. You will be cascading these two converters thus implementing a binary to gray coder and decoder (gray to binary).

The next part is the Verilog Modeling and Simulation of the Circuit you implemented in you first lab.

**Objectives:**

* Understand steps involved in design of combinational circuits
* Understand binary codes for decimals and their hardware realization
* Write code for combinational circuits using Verilog Gate Level Modeling
* Design a circuit in Verilog by calling different modules

**Lab Instructions**

* This lab activity comprises three parts, namely Pre-lab, Lab tasks, and Post-Lab Viva session.
* The lab report will be uploaded on LMS three days before scheduled lab date. The students will get hard copy of lab report, complete the Pre-lab task before coming to the lab and deposit it with teacher/lab engineer for necessary evaluation. Alternately each group to upload completed lab report on LMS for grading.
* The students will start lab task and demonstrate design steps separately for stepwise evaluation (course instructor/lab engineer will sign each step after ascertaining functional verification)
* Remember that a neat logic diagram with pins numbered coupled with nicely patched circuit will simplify trouble-shooting process.
* After the lab, students are expected to unwire the circuit and deposit back components before leaving.
* The students will complete lab task and submit complete report to Lab Engineer before leaving lab.
* **The Total duration for the lab is 3 hrs. After lab duration, a deduction of 5 marks per day will be done for late submission.**
* **A lab with in-complete lab tasks will not be accepted.**
* There are related questions at the end of this activity. Give complete answers.

**Pre – Lab Tasks**

**Pre – Lab Task 1**

What do you mean by binary codes for the decimal digits? Give some examples and codes (tables) for the decimal digits.

**Answer:**

Coding decimal digits is a system of writing numerals that assigns a four – bit code to each digit 0 through 9 in a decimal (base-10) numeral and such a process is known as encoding. Following table demonstrate the encoding of all the numbers in the decimal base to Stibitz Code as well as BCD Code.

|  |  |  |
| --- | --- | --- |
| **Decimal** | **BCD Code** | **Excess – 3** |
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| 3 | 0011 | 0110 |
| 4 | 0100 | 0111 |
| 5 | 0101 | 1000 |
| 6 | 0110 | 1001 |
| 7 | 0111 | 1010 |
| 8 | 1000 | 1011 |
| 9 | 1001 | 1100 |

**Pre – Lab Task 2**

What is a self-complementing code? Name any two of them; show their complementing nature with examples and describe advantages.

**Answer:**

Self-complementing codes are the one with the property that 9’s complement of a decimal number is obtained directly from the code by taking 1’s complement of its bit in the pattern.

|  |  |  |
| --- | --- | --- |
| **Decimal** | **2 4 2 1 Code** | **Stibitz Code** |
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| 3 | 0011 | 0110 |
| 4 | 0100 | 0111 |
| 5 | 1011 | 1000 |
| 6 | 1100 | 1001 |
| 7 | 1101 | 1010 |
| 8 | 1110 | 1011 |
| 9 | 1111 | 1100 |

**Example:**

* 27610 is 0101 1010 1001 in Excess-3. Taking 1’s complement, we have 1010 0101 0110 (Excess-3) which is 723 in decimal and is equal to the 9’s complement of the given number.

**Advantages:**

* Hardware design is easier (9’s complement subtraction using adder circuit)
* Increase in Speed of Operation.

**Pre – Lab Task 3**

In the lab you would be implementing a gray to binary and binary to gray code converter. Make a truth table for both the codes by filling in the following tables and simplify the expressions for W, X, Y, Z in terms of A, B, C, D and vice versa. (Use backside of the page if necessary). Also give some applications in which gray code could be used.

**Binary to Gray Converter**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Binary** | | | | **Gray** | | | |
| **A** | **B** | **C** | **D** | **W** | **X** | **Y** | **Z** |
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 1 | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| 2 | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| 3 | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| 4 | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** |
| 5 | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** |
| 6 | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** |
| 7 | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **0** |
| 8 | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| 9 | **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** |
| 10 | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** |
| 11 | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** |
| 12 | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| 13 | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| 14 | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** |
| 15 | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |

**Simplified Expressions**

|  |  |  |
| --- | --- | --- |
| **W** | **A** | **A** |
| **X** | **A’.B + A.B’** | **A ⊕ B** |
| **Y** | **B’.C + B.C’** | **B ⊕ C** |
| **Z** | **C’. D + C.D’** | **C ⊕ D** |

**Gray to Binary Converter**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Gray** | | | | **Binary** | | | |
| **W** | **X** | **Y** | **Z** | **A** | **B** | **C** | **D** |
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 1 | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| 2 | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| 3 | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| 4 | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **0** |
| 5 | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **1** |
| 6 | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| 7 | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| 8 | **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |
| 9 | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| 10 | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** |
| 11 | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** |
| 12 | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** |
| 13 | **1** | **0** | **1** | **1** | **1** | **1** | **0** | **1** |
| 14 | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** |
| 15 | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |

**Simplified Expressions**

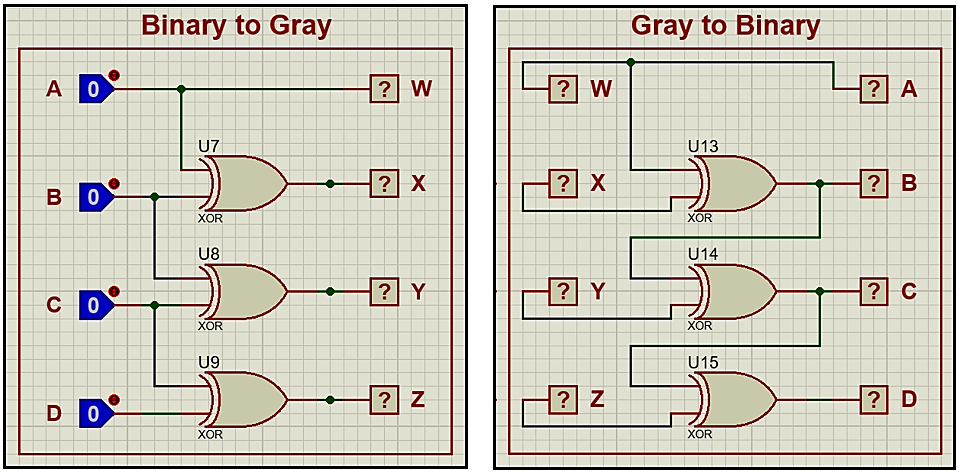
|  |  |
| --- | --- |
| **A** | **W** |
| **B** | **W ⊕ X** |
| **C** | **W ⊕ X ⊕ Y** |
| **D** | **W ⊕ X ⊕ Y ⊕ Z** |

**Applications of Gray Code:**

* Gray codes are used in linear and rotary position encoders in preference to weighted binary encoding. This avoids the possibility that, when multiple bits change in the binary representation of a position, a misread will result from some of the bits changing before others.
* Gray codes are also used in labelling the axes of Karnaugh maps since 1953as well as in Händler circle graphs since 1958, both graphical methods for logic circuit minimization.
* Digital logic designers use gray codes extensively for passing multi-bit count information between synchronous logic that operates at different clock frequencies. The logic is considered operating in different "clock domains". It is fundamental to the design of large chips that operate with many different clocking frequencies.
* In modern digital communications, gray codes play an important role in error correction.

**Pre – Lab Task 4**

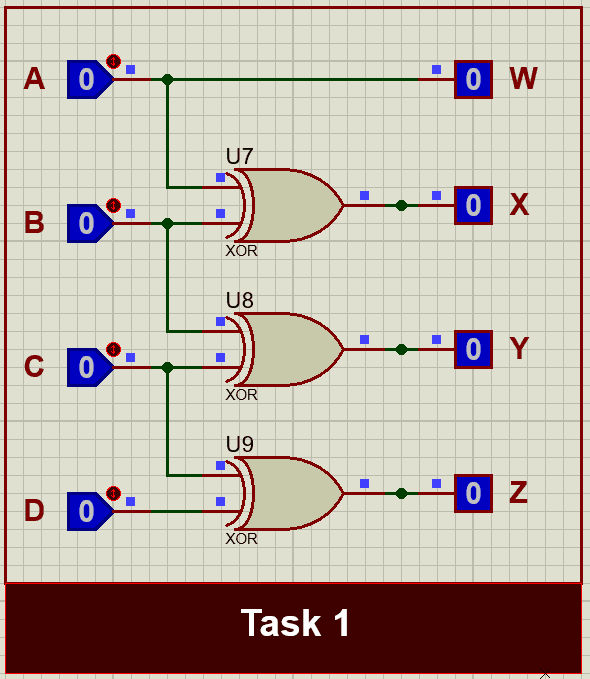
Draw the logic diagram for the Binary-to-Gray and Gray-to-Binary code converters using Exclusive-OR gates in the space provided below.

****

**Lab Tasks**

**Lab Task 1**

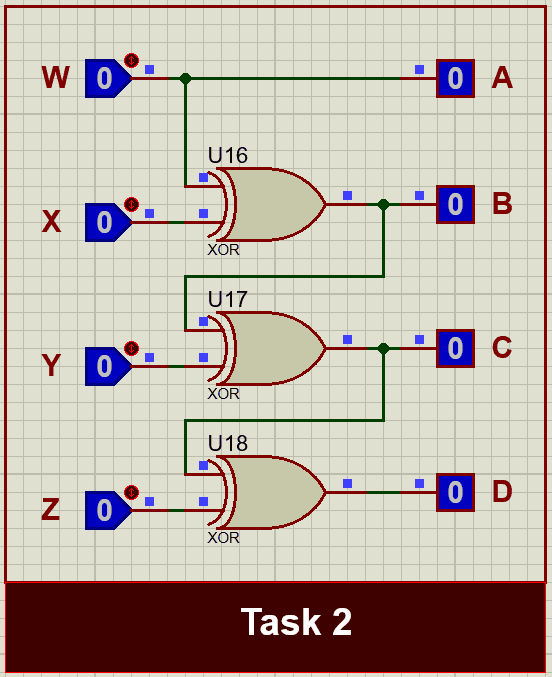
Implement the Binary to Gray Code Converter using logic gates. Make the Schematic Diagram. Show the results to your Teacher/Lab Engr. What and how many gates did you use?



*Which is in accoradance with the truth table laid out in the Task 3 of Pre – Lab.*

**Lab Task 2**

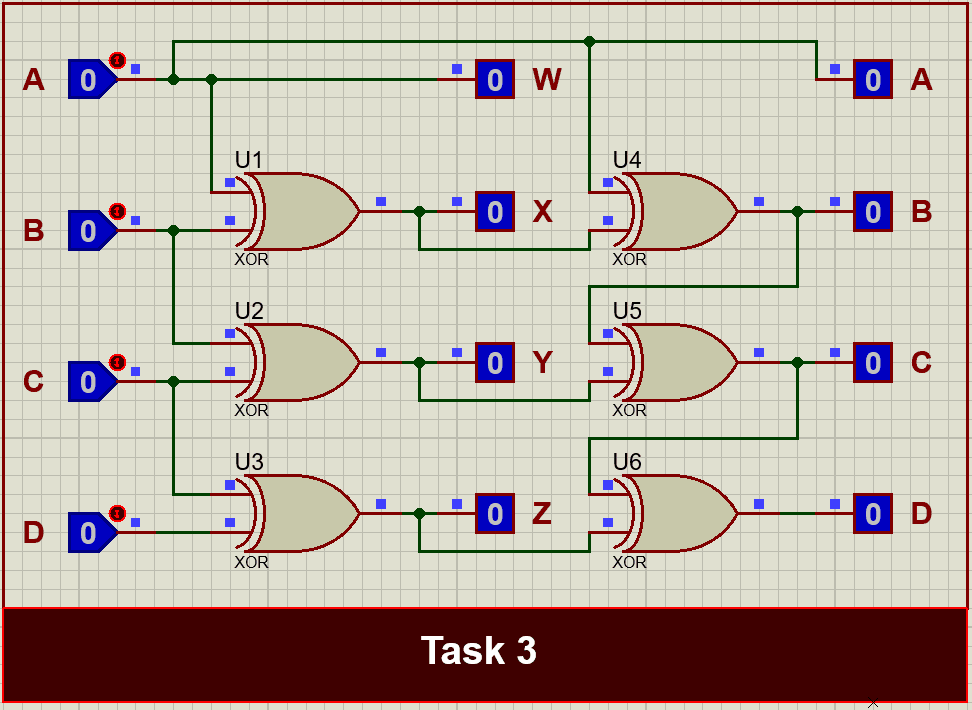
Realize the Gray to Binary Code Converter using exclusive-OR gates. Make the Schematic Diagram. Show the results to your Teacher/ Lab Engr. What and how many gates did you use?



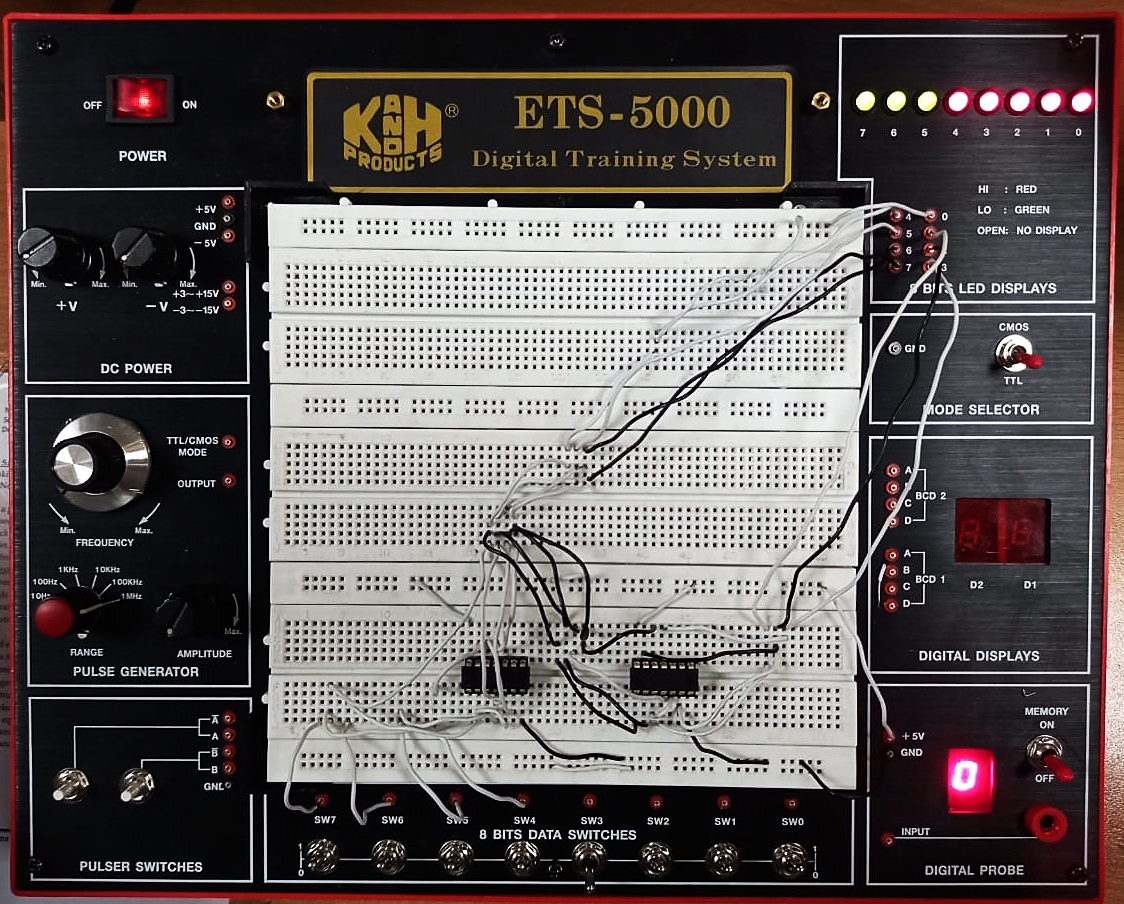
*Which is in accoradance with the truth table laid out in the Task 3 of Pre – Lab.*

**Lab Task 3**

Now cascade the two circuits in series by connecting the outputs of binary-to-gray converter to the inputs of the gray-to-binary converter. You should be able to get the binary input at output as well. Show the results to your Teacher/Lab Engr. Use LEDs to show input-output relationship.



**Hardware Implementation**



Where the LEDs from 0 – 7 represent:

|  |  |
| --- | --- |
| **Number** | **Variable** |
| **0** | **A** |
| **1** | **B** |
| **2** | **C** |
| **3** | **D** |
| **4** | **W** |
| **5** | **X** |
| **6** | **Y** |
| **7** | **Z** |

**Lab Task 4**

Design and simulate the gate-level model of the circuit you patched in **task 3**. Give the code in the space provided below.

module b2g2b(oa, ob, oc, od, w, x, y, z, a, b, c, d);

    input a, b, c, d;

    output oa, ob, oc, od, w, x, y, z;

    wire w1, w2, w3;

    // B2G

    buf b1(w, a);

    buf b2(oa, a);

    xor x1(w1, a, b);

    buf b3(x, w1);

    xor x2(w2, b, c);

    buf b4(y, w2);

    xor x3(w3, c, d);

    buf b5(z, w3);

    // G2B

    xor x4(ob, a, w1);

    xor x5(oc, ob, w2);

    xor x6(od, oc, w3);

endmodule

module testbench();

    reg A, B, C, D;

    wire OA, OB, OC, OD, W, X, Y, Z;

    b2g2b t1(OA, OB, OC, OD, W, X, Y, Z, A, B, C, D);

    initial begin

        #100 A = 0; B = 0; C = 0; D = 0;

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

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

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

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

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

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

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

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

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

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

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

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

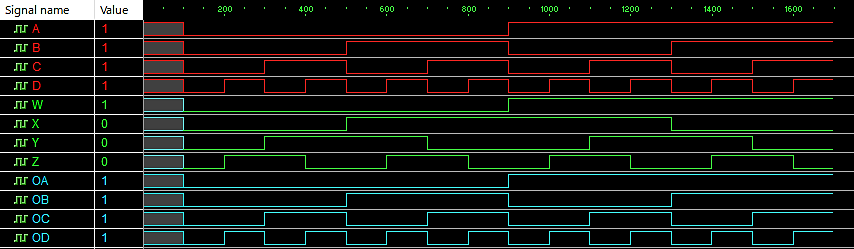
        #100 A = 1; B = 1; C = 0; D = 1;

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

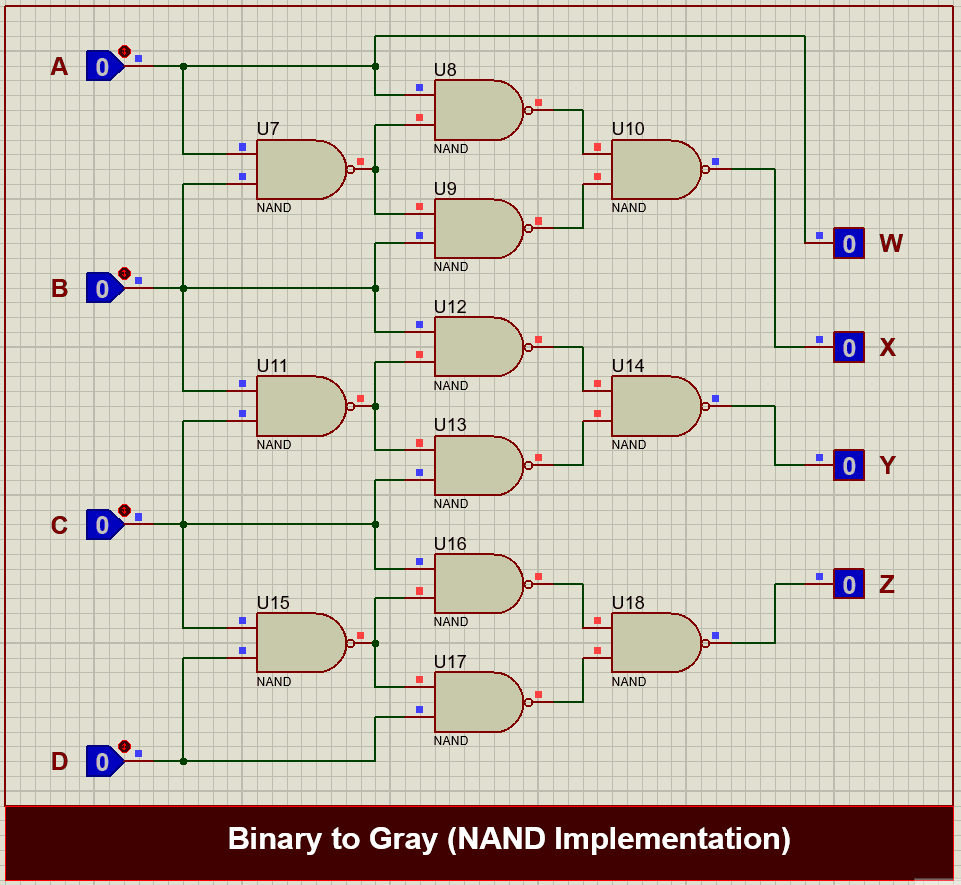
        #100 A = 1; B = 1; C = 1; D = 1;

    end

endmodule



**Alternative Implementation (Using NAND Gates Only)**



**Observations/Comments:**

In this lab we implemented the binary to gray and gray to binary circuit using:

1. Basic gates (AND, NOT and OR)
2. Using XOR.

We identified the outputs of the converters using LEDs and wrote the Verilog code for the circuit which converted binary to gray to binary.