Experiment No. 3	
Code Converters	
Course Code: CPET 6L	**Program:**
Course Title: Introduction to HDL	**Date Performed: OCT. 2, 2020**
Section: BET-CpET 2AS	**Date Submitted: OCT 3, 2020**
Name: DENNE JOSHUA SUELAN	**Instructor: Engr. Johnathan Richard A. Barrios**

1. Objective(s):

The activity aims to simulate a 4-bit binary to gray code converter and vice versa using VHDL codes.

2. Intended Learning Outcomes (ILOs)

The students should be able to:

- 2.1 Create the process VHDL codes and test bench of code converters.
- 2.2 Create a simulation timing diagram of code converters.
- 2.3 Create a truth table in 10 ns.
- 2.4 Compare the difference between 4-bit binary to gray code and gray code to 4-bit binary.

3. Discussion:

Code Converters

The converters, which convert one code to other code are called as code converters. These code converters basically consist of Logic gates. Gray codes are non-weighted codes, where two successive values differ only on one bit. This is to simulate two simple gate level VHDL codes for converting binary number to Gray and vice versa.

Binary code to Gray code converter

Let us implement a converter, which converts a 4-bit binary code WXYZ into its equivalent Gray code ABCD. The following table shows the Truth table of a 4-bit binary code to Gray code converter.

Since the outputs depend only on the present inputs, this 4-bit Binary code to Gray code converter is a combinational circuit. Similarly, you can implement other code converters.

Table 3.1 Truth Table of the 4-bit Binary Code to Gray Code Converter

Binary code WXYZ	WXYZ Gray code ABCD
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Table 3-1 shows the conversion of binary to gray code

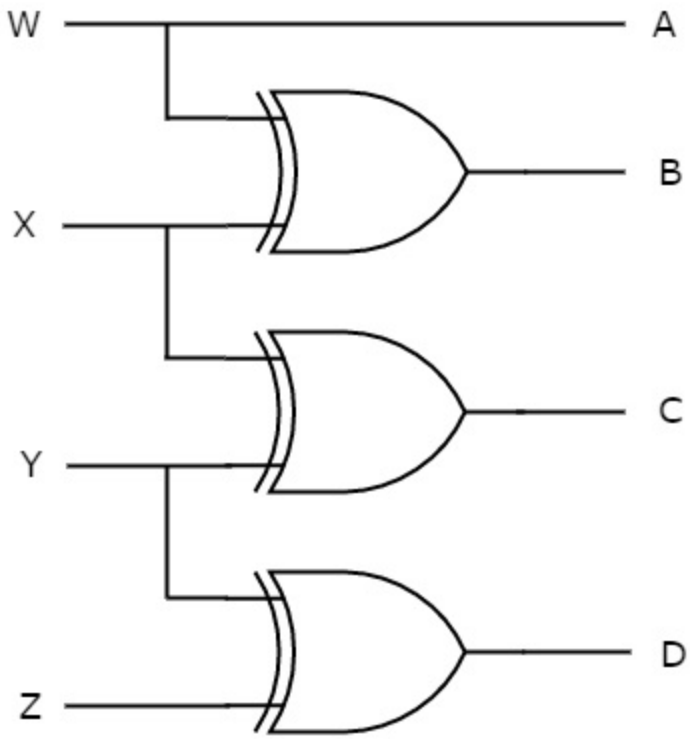


Figure 3.1 Circuit Diagram of a 4-bit binary code to Gray Code Converter

The circuit on Fig 3.1 shows the logic diagram from the K-Map method of the variable w, x, y, and z.

4. Resources:

Computer System and Xilinx application
Spartan 601 Board (Optional)

5. Procedure:

1. Create a VHDL project (inside Xilinx Application) file for each of the following:
    a. 4-bit binary code to gray code
    b. Gray code to 4-bit binary code
2. Create the VHDL source code for the two-logic converter (see experiment 1 for the procedure of creating VHDL module and test bench).
3. Create test a bench of the two-logic converter.
4. Create a simulation timing diagram of the two-logic converter.
5. Generate RTL schematic diagram of the two-logic converter, identify the number of slices, 4 input LUTs and bonded IOB's.
6. And show the results on the space provided.

5.2.1Binary to Gray Code Conversion VHDL CODE

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity bin2gray is
port(
bin : in std_logic_vector(3 downto 0);  --binary input

G : out std_logic_vector(3 downto 0)  --gray code output
       );
end bin2gray;

architecture gate_level of bin2gray is

begin

--xor gates.
G(3) <= bin(3);
G(2) <= bin(3) xor bin(2);
G(1) <= bin(2) xor bin(1);
G(0) <= bin(1) xor bin(0);

end;
```

5.2.2Gray Code to Binary Conversion VHDL Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity gray2bin is
port(
G : in std_logic_vector(3 downto 0);
  --gray code input

bin : out std_logic_vector(3 downto 0)  --binary output
       );
end gray2bin;

architecture gate_level of gray2bin is

begin

--xor gates.
bin(3) <= G(3);
bin(2) <= G(3) xor G(2);
bin(1) <= G(3) xor G(2) xor G(1);
bin(0) <= G(3) xor G(2) xor G(1) xor G(0);

end;
```

5.3.1 Test Bench for both the designs

```
library ieee;
use ieee.std_logic_1164.all;

entity tb is
end tb;

architecture behavior of tb is

   -- component declaration for the unit under test's (uut)
component bin2gray is
   port(   bin : in std_logic_vector(3 downto 0);
          g : out std_logic_vector(3 downto 0)
          );
end component;
```

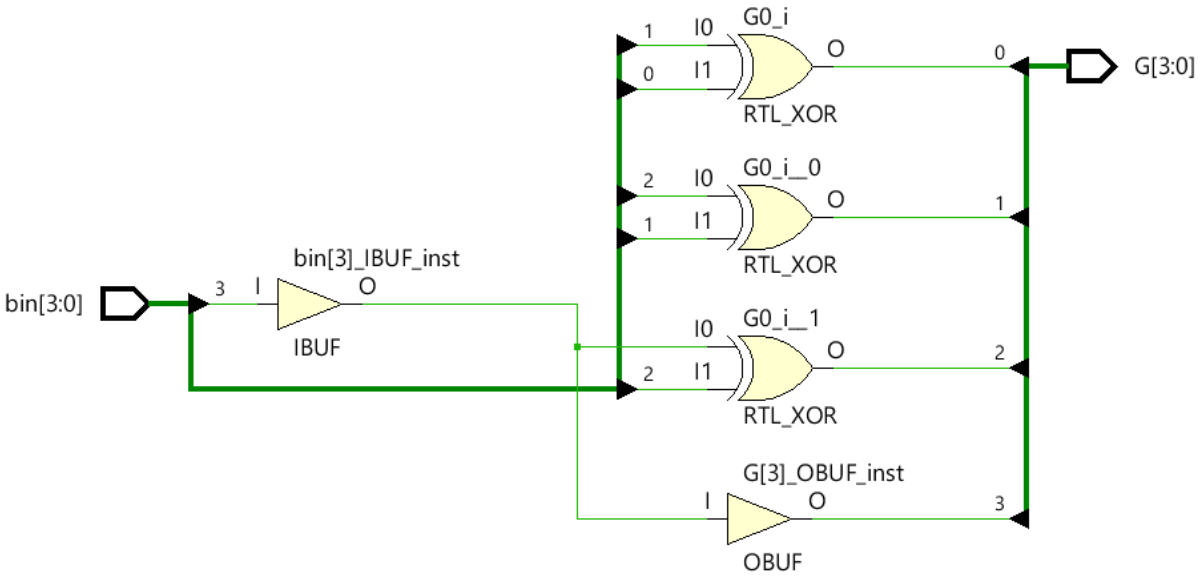
```
component gray2bin is
port(   g : in std_logic_vector(3 downto 0);
       bin : out std_logic_vector(3 downto 0)
       );
end component;

signal bin,g,bin_out : std_logic_vector(3 downto 0) := (others => '0');

begin
   -- instantiate the unit under test's (uut)
  uut1: bin2gray port map (
        bin => bin,
        g => g
       );

  uut2: gray2bin port map (
        g => g,
        bin => bin_out
       );

  -- stimulus process
  stim_proc: process
  begin
       bin <= "0000";    wait for 10 ns;
       bin <= "0001";  wait for 10 ns;
       bin <= "0010";  wait for 10 ns;
       bin <= "0011";  wait for 10 ns;
       bin <= "0100";    wait for 10 ns;
       bin <= "0101";  wait for 10 ns;
       bin <= "0110";  wait for 10 ns;
       bin <= "0111";  wait for 10 ns;
       bin <= "1000";    wait for 10 ns;
       bin <= "1001";  wait for 10 ns;
       bin <= "1010";  wait for 10 ns;
       bin <= "1011";  wait for 10 ns;
       bin <= "1100";    wait for 10 ns;
       bin <= "1101";  wait for 10 ns;
       bin <= "1110";  wait for 10 ns;
       bin <= "1111";  wait for 10 ns;
     wait;
  end process;

end;
```

5.5.1 Show the resulting RTL Schematic of the Binary to Gray Code Converter

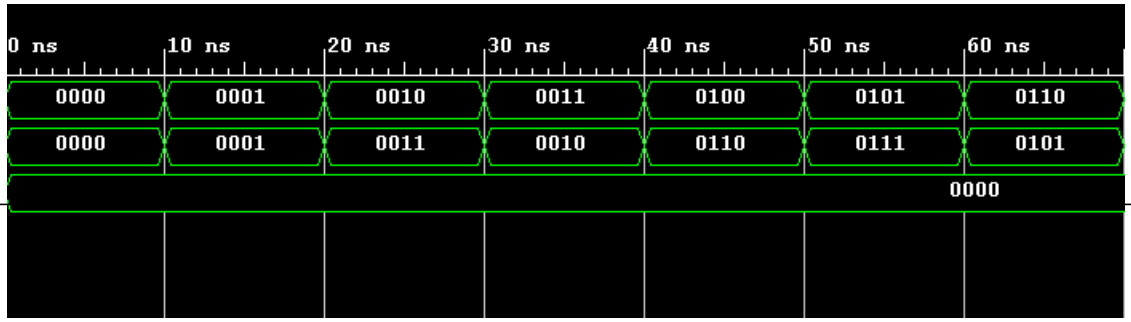
Number of Slices: _2__

Number of 4 input LUTs : _2__

Number of bonded IOBs: _8__



5.4.1 Show the simulation timing diagram of Binary to Gray Code Converter

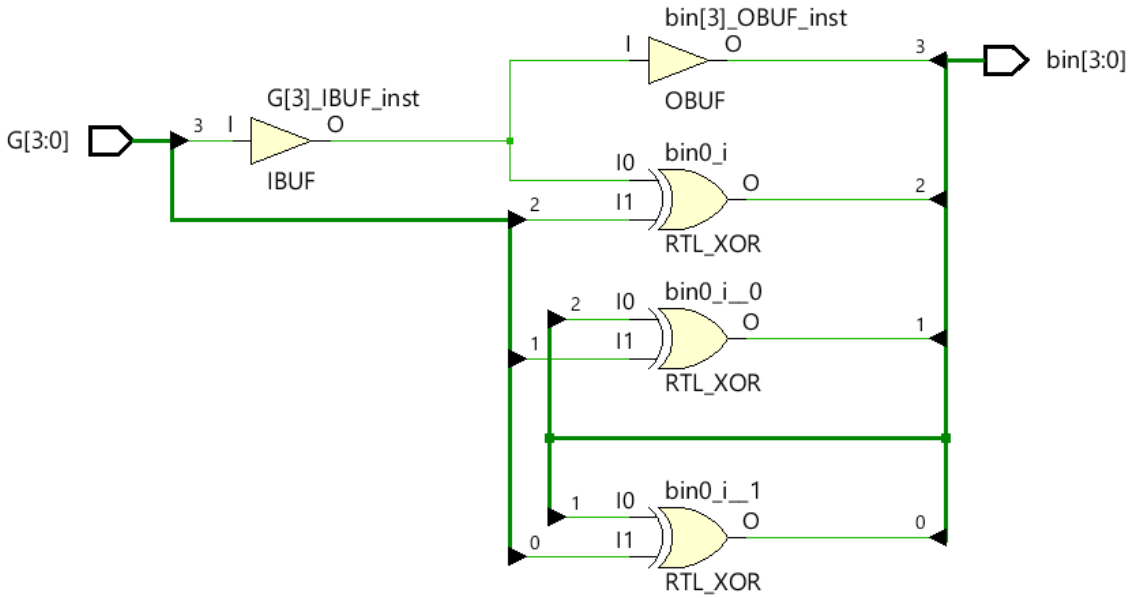


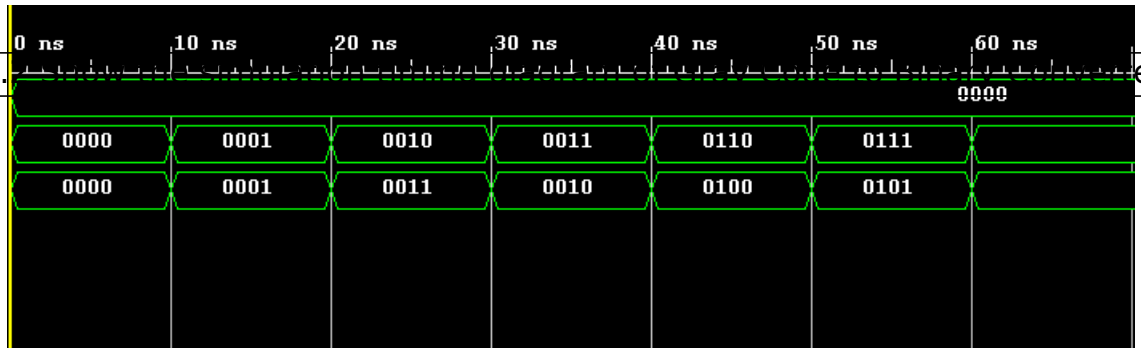
5.5.2 Show the resulting RTL Schematic of the Gray code to Binary Code Converter

Number of Slices: 2

Number of 4 input LUTs : 2

Number of bonded IOBs: 8





5.4. er

6. Questions:

a. Relate the VHDL module to test bench in terms of the Code Converter source codes.

Well to relate VHDL to test bench in terms of code converter, let us breakdown to each part.

First is the module: The module consists of codes to define the inputs and outputs while on behalf of that you can't run your module without test bench. The role of test bench is to process the inputs and simulate later. Well, based on the activity it is necessary to change the stimulus process in able to get the correct conversion of bin2gray and gray2bin.

b. Explain the behavior of the code converter's test bench.

It is simple that by converting, code must be in correct value and also changing the entities will make a huge impact on the results.

7. Data and Results:

(Show the equivalent truth table of each logic gates in 10 nano seconds)
Note: Write or indicate the input and output variable used in the program and give analysis sentence for each table. You may change the column and rows of each table according to the number of input and output used.

Table 3.2 Binary to Gray Code Truth Table

Every 10 ns	Input (bin)	Output(gray)
10ns	0000	0000
20ns	0001	0001
30ns	0010	0011
40ns	0011	0010
50ns	0100	0110

Table 3.2 Gray code to Binary Code Truth Table

Every 10 ns	Input (gray)	Output(bin)
10ns	0000	0000
20ns	0001	0001

30ns	0010	0010
40ns	0011	0011
50ns	0110	0100

8. Observation and Analysis:

1st observation on bin2gray: the code converts binary to gray code by simulation.

2nd observation on gray2bin: same process but this time, gray converted to binary and you can't convert it without replacing the stimulus process. Changing values of data input to gray will convert it binary.

Analysis: if the code created is correct therefore there is no exception that both coded will work. By putting the correct stimulus process you are able to execute the code with good result.

9. Conclusion:

Based on the conducted activity, I am able to distinct the difference of converting bin to gray and gray to bin by using the correct code and process in vivado. By putting correct values of bin and gray you will execute the code correctly.

10. Assessment (Rubric for Laboratory Performance):