# K L Deemed to be University

**Department of Electronics & Communication Engineering**

**B. Tech ECE Third Year Semester-I A.Y.2020-21, ODD Semester**

**Course Title: Technical Proficiency & Training -1**

|  |  |
| --- | --- |
| ID Number | Name |
| 180040256 | N.Eswar |
| 180040257 | K Yogananda Sagar |
| 180040263 | P.Sri Vidya |

**4-BIT GRAY CODE COUNTER USING VHDL**

## CONTENTS

|  |  |
| --- | --- |
| Abstract | 1 |
| Introduction | 1 |
| Methodology | 2-5 |
| Implementation | 6-8 |
| Results | 9 |
| Applications&  Advantages and Disadvantages | 10 |
| References | 10 |
| THANK YOU | 11 |
|  |  |

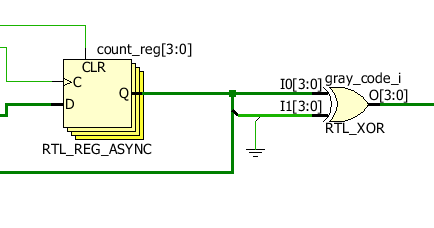
**ABSTRACT**

## A Gray code counter which has an iterative and relatively simple structure is described. The code is shown to be the reflected binary Gray code, implying simple conversion of the count into binary code. A Gray Code is an encoding of integers as sequences of bits with the property that the representations of adjacent integers differ in exactly one binary position. Gray Codes have many practical applications that go beyond research interests.Counter is a digital device and the output of the counter includes a predefined state based on the clock pulse applications. The output of the counter can be used to count the number of pulses. Generally, counters consist of a flip-flop arrangement which can be synchronous counter or asynchronous counter. Counters have a primary function of producing a specified output sequence and are thus sometimes referred to as pattern generators.

## INTRODUCTION

Gray code was invented by Frank Gray of Bell Labs in 1947. He called his code as reflected binary code. Later others started calling it as Gray code. This code has also come to be known as mirror code. Both the names, viz., reflected binary code and mirror code, stem from its mirror-reflection property. The main features of the Graycodeare:-It is a binary code just like to the conventional binary code.  
The code can be constructed using its mirror-reflecting property.  
 Adjacent numbers of the code differ in one bit-position only. Gray code is not weighted that means it does not depends on positional value of digit. This cyclic variable code that means every transition from one value to the next value involves only one bit change.

## METHODLOGY

 In a Gray code only one bit changes at a one time. This design code has two inputs clock and reset signals and one four bit output that will generate Gray code.

In the first if rst signal is high then output will be zero .

As soon as rst will go low, on the rising edge of clk, design will generate four bit Gray code and continue to generate at every rising edge of clk signal.

No 4 bit binary code greycode

* 0 0000 0000
* 1 0001 0001
* 2 0010 0011 ……. So on

## Explanation

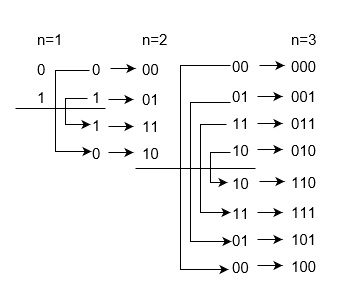
## Constructing an n-bit Gray code

n-bit Gray code can be generated recursively using reflect and prefix method which is explained as following below.

* Generate code for n=1: 0 and 1 code.
* Take previous code in sequence: 0 and 1.
* Add reversed codes in the following list: 0, 1, 1 and 0.
* Now add prefix 0 for original previous code and prefix 1 for new generated code: 00, 01, 11, and 10.

| **For n = 1 bit** | | **For n = 2 bit** | | **For n = 3 bit** | |
| --- | --- | --- | --- | --- | --- |
| Binary | Gray | Binary | Gray | Binary | Gray |
| 0 | 0 | 00 | 00 | 000 | 000 |
| 1 | 1 | 01 | 01 | 001 | 001 |
|  | | 10 | 11 | 010 | 011 |
| 11 | 10 | 011 | 010 |
|  | | 100 | 110 |
| 101 | 111 |
| 110 | 101 |
| 111 | 100 |

Therefore, Gray code 0 and 1 are for Binary number 0 and 1 respectively. Gray codes: 00. 01, 11, and 10 are for Binary numbers: 00, 01, 10, and 11 respectively. Similarly you can construct Gray code for 3 bit binary numbers:



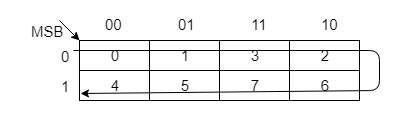
Therefore, Gray codes are as following below,

Iterative method of generating G(n+1) from Gn are given below. This is simpler method to contract Gray code of n-bit Binary numbers. Each bit is inverted if the next higher bit of the input value is set to one. The nth Gray code is obtained by computing n⊕(floor(n/2)).

* Gn is unique numbers for the permutation from 0  to (2n-1).
* Gn is embedded as the first half of G(n+1) and second half as the reverse order of G(n+1).
* Prefix 0 in each digit of first half and 1 in each digit of second half.

The hamming distance of two neighbours Gray codes is always 1 and also first Gray code and last Gray code also has Hamming distance is always 1, so it is also called Cyclic codes.

You can construct Gray codes using other methods but they may not be performed in parallel like given above method. For example, 3 bit Gray codes can be contracted using K-map which is given as following below:



| **Decimal** | **Binary** | **Gray Code** |
| --- | --- | --- |
| 0 | 000 | 000 |
| 1 | 001 | 001 |
| 2 | 010 | 011 |
| 3 | 011 | 010 |
| 4 | 100 | 110 |
| 5 | 101 | 111 |
| 6 | 110 | 101 |
| 7 | 111 | 100 |

## Types of Gray Codes

There are also other types of Gray codes, like Beckett-Gray code, Single track Gray codes etc.

* N-ary Gray code, where non-Boolean values are included like sequences of 1, 2, 3.
* Two dimensional (n,k) Gray codes are used for error correction.
* Balanced Gray codes has equal transition counts.

## Uses of Gray codes

Gray codes are used in rotary and optical encoders, Karnaugh maps, and error detection.

**IMPLEMENTATION/Code**

*4 Bit Grey code counter:*

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity gray\_counter is

port(clk: in std\_logic;

rst: in std\_logic;

gray\_code: out std\_logic\_vector (3 downto 0));

end gray\_counter;

architecture beh of gray\_counter is

signal count : std\_logic\_vector(3 downto 0) := "0000";

begin

process(clk)

begin

if (rst='1') then

count <= "0000";

elsif (rising\_edge(clk)) then

count <= count + "0001";

end if;

end process;

gray\_code <= count xor ('0' & count(3 downto 1));

end beh;

*4 Bit Grey code counter­\_Tb:*

library ieee;

use ieee.std\_logic\_1164.all;

entity gray\_counter\_tst is

end gray\_counter\_tst;

architecture beh of gray\_counter\_tst is

component gray\_counter is

port(clk: in std\_logic;

rst: in std\_logic;

gray\_code: out std\_logic\_vector (3 downto 0));

end component;

signal rst\_s : std\_logic;

signal gray\_code\_s : std\_logic\_vector(3 downto 0);

signal clk\_s : std\_logic := '0'; -- clk signal

constant clk\_period : time := 10 ns;

begin -- arch

DUT : gray\_counter port map (

rst => rst\_s,

clk => clk\_s,

gray\_code => gray\_code\_s)

process

begin -- process clockk

clk\_s <= not clk\_s;

wait for clk\_period/2;

end process;

process

begin

rst\_s <= '1';

wait for 11 ns;

rst\_s <= '0';

wait for 200 ns;

end process;

end beh;

# RESULTS:

4-BIT GRAY CODE COUNTER USING VHDL:-

A screenshot of a computer

Description automatically generated

TEST BENCH Results:-

A picture containing computer

Description automatically generated

## Applications of Comparators Using Grey

Some other applications of gray code:

Boolean circuit minimization

Communication between clock domains

Error correction

Genetic algorithms

Mathematical puzzles

Position encoders

**Advantages of Gray Code:-**

Better for error minimization in converting analog signals to digital signals

Reduces the occurrence of “Hamming Walls” (an undesirable state) when used in genetic algorithms

Can be used to in to minimize a logic circuit

Useful in clock domain crossing

**Disadvantages of Gray Code:-**

Not suitable for arithmetic operations

Limited practical use outside of a few specific application

## REFERENCES:

* <https://www.chegg.com/homework-help/questions-and-answers/1-design-3-bit-synchronous-gray-code-counter-enable-preset-outputs-follow-reflected-gray-c-q42232381>
* <https://electronics.stackexchange.com/questions/26677/3bit-gray-counter-using-d-flip-flops-and-logic-gates>

## Thank you