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

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 Gray code are:

- 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.

The reflected Binary Code (RBC) is the unfamiliar and original name of Gray Code. As the original name implies, the values of the second half of a GREY CODE set are equivalent to the first half but in reverse order. This with the exception for the most significant bit (MSB) which is inverted i.e. 0 changes to 1. GREY CODEs were originated when digital logic circuits were built around vacuum tubes and electromechanical relays. And thus, counters were generating enormous power demands and noise spikes when many bits changed at the same time. A GREY CODE is a binary code with code-words having unitary Hamming distance between each other of the codes. A GREY CODE of length n bits is represented by all the integers from 0 to 2^n-1 in the forms of bit patterns (called “words”) of length n. Consecutive words in a GREY CODE set includes the adjacency.

Property which can be stated in the form that any two consecutive code words differ by only one-bit position from each other. If the words in a GREY CODE set are represented by integers in ascending order, then correspondingly any two neighbouring integers will have the adjacency property by default. In this manner, it is feasible to build counters around GREY CODEs whose increment or decrement affects a single bit only. For such counters, irrespective of their sizes, the effect of the noise will be minimized.

Numerous other binary codes do exist. This includes Binary Coded Decimal (BCD), Excess-3 code, Hamming code, and Cyclic Redundancy Check (CRC) code, just to name a few. However, unlike each of the binary codes, GREY CODEs are particularly well suited to handle control problems in a robust and convenient manner. This can be attributed back to its unitary distance property which can avoid ambiguous switching situations. For such reasons, the number of applications in which GREY CODEs can be accounted for is unlimited.

METHODLOGY

A 4-bit binary counter sequence is represented by D = [D3 D2 D1 D0], while the Gray encoded bit sequence is denoted by A = [A3 A2 A1 A0]. Each row of the binary counter sequences D can generate the following row by simply incrementing 1 to the current row. The binary counter of 4-bits therefore generates 16 unique binary sequences. This follows from the fact that the number of unique bit sequences that can be derived from 4 bits is 2^4 = 16 sequences. Similarly, through observation, for each of the possible 4-bit binary counter sequences there exists a unique Gray encoded sequence. In contrast to the trivial relation that exists between the rows of the binary counter, the simple increment is not directly evident for the case of Gray encoded bits. As such, a more futile method of generating the Gray encoded bits may be to directly observe the transitions from 1 to 0 (High to Low) and 0 to 1 (Low to High) of each column. This observation must be conducted for each column separately and in conjunction with the neighbouring column. To facilitate the observation, waveform timing diagram of each Gray encoded bit is also included.

As the system is based on digital components, it is natural that first waveform in Figure 1 represents a clock waveform, Clk. This is followed by waveforms of bits A0, A1, A2 and B3, where it is assumed at this point that B3 is equivalent to A3. In the diagram, it has further been assumed that each of the four waveforms will only transition states (High to Low or Low to High) on the falling edge of signal Clk. To relate the transition rates of neighbouring waveforms, it is clear from the diagram that waveform A0 transitions at half the rate compared to waveform Clk. Similarly, the transition rate of waveform A1 is half that of A0. Accordingly, it can be generally stated that for every subsequent waveform the transition rate is half compared to the current waveform transition rate.

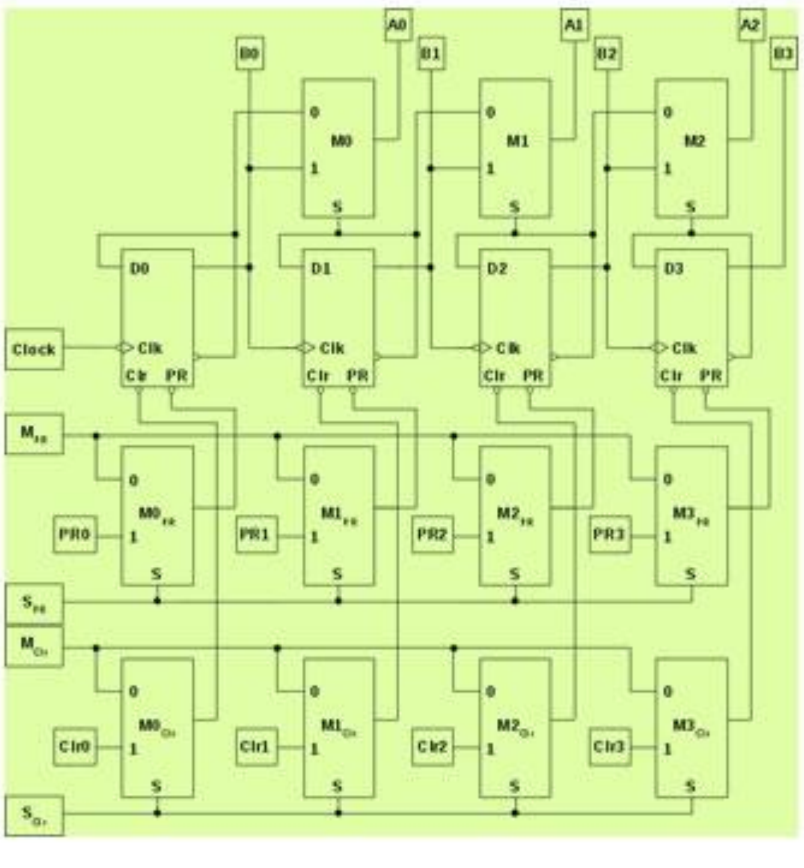
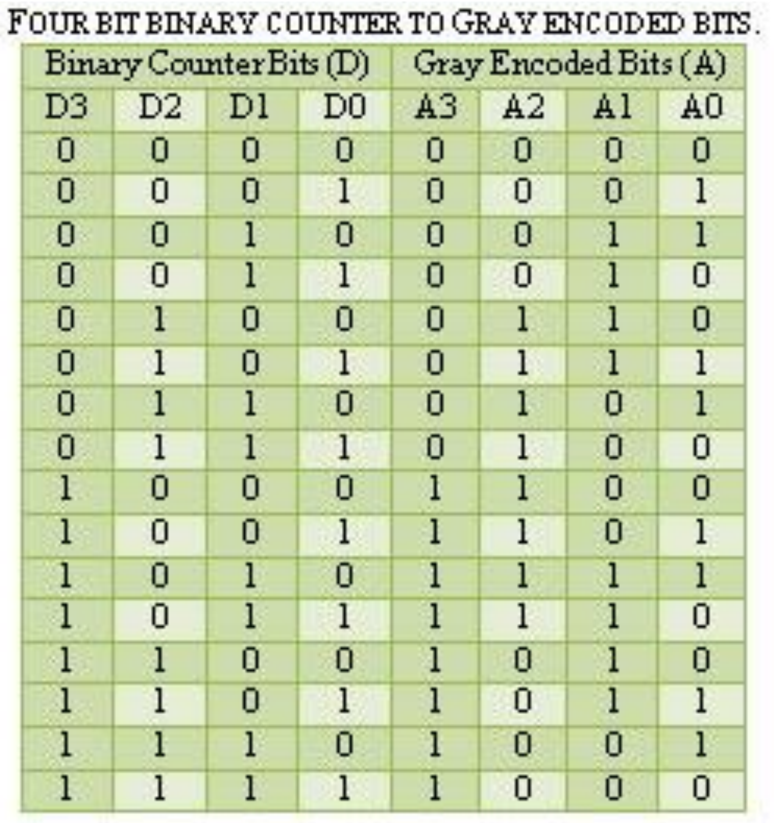


Fig.2: A D-FF/MUX 4-bit Binary to Gray Encoder/Counter


FOUR BIT BINARY COUNTER TO GRAY ENCODED BITS.

Binary Counter Bits (D)				Gray Encoded Bits (A)			
D3	D2	D1	D0	A3	A2	A1	A0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

The second task is to establish a relation with respect to different portions of the same waveform. Consider that the first two levels of waveform A0, which spans over clock cycles 0 to 2, constitutes a waveform Basis Function for A0. Denote this basis function as FA0, which is depicted as the top left basis function in Figure 1. Then at the next two clock cycles, 2 and 3, waveform A0 is currently the inverted version of FA0. A0 then resumes its original non-inverted state in the following two clock cycles 4 and 5. This process continues to repeat as can be directly observed from the waveform figure. The sequence A0 can therefore be constructed from basis function FA0 as A0 = {FA0, -FA0, FA0, -FA0, ….}. The number of elements in sequence A0 can be denoted as NA0 and is directly deduced from Figure 1 as NA0 = 8. Using the same line of thought, the first two levels of waveform A1 that constitutes the wave basis function FA1, now spans over two A0 cycles, or equivalently 4 cycles of signal Clk. The next two levels of waveform A1 is simply an inverted version of FA1. The process then continues for the remaining portions of waveform A1, with the total number elements making up sequence A1, NA1, is 4. This established relation can be generalized for every waveform as shown in the diagram, except for the last waveform, B3. Along these lines, the generalization to construct each Gray encoded bit waveform basis function begins by generating the current waveform basis function. This is achieved by initializing the basis function with a Low state. This state is then held over one complete time duration of the previous basis function. The current basis function then complements its state over another complete cycle of the previous Basis Function. If, however, the Basis Function belongs to the initializing waveform, A0, then it is a single cycle of the waveform Clk that acts as the previous waveform Basis Function. Besides B3 no corresponding Basis Function is shown. The reason being is that it can be easily deduced from table 1. From the table it can be observed that neighboring Gray codes exhibit only a single bit change. Once the code reaches the end of the table at code 1000, it therefore changes to 0000. Through observation, the leftmost bit over all Gray code words simply alternates between a 0 and 1, every 8 clock cycles. In this case it is needless to construct a Basis Function as NB3 = 1.

IMPLEMENTATION/Code

4 Bit Grey code counter:

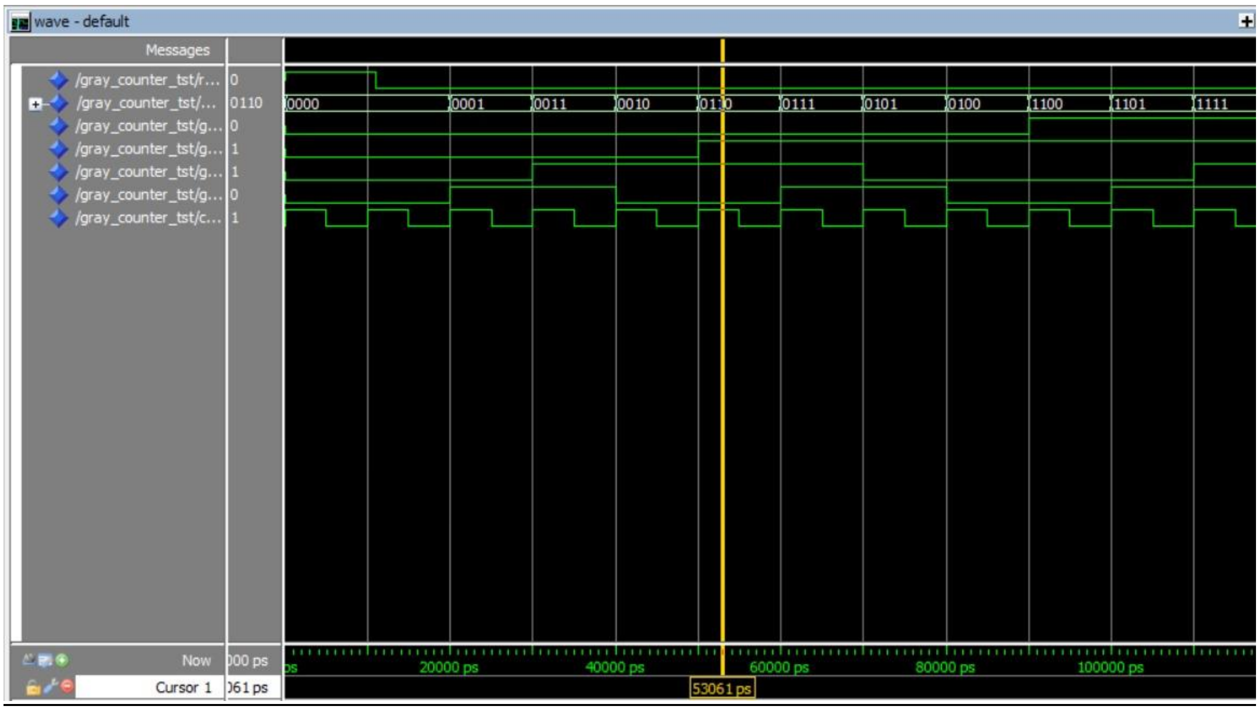
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

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_Testbench:

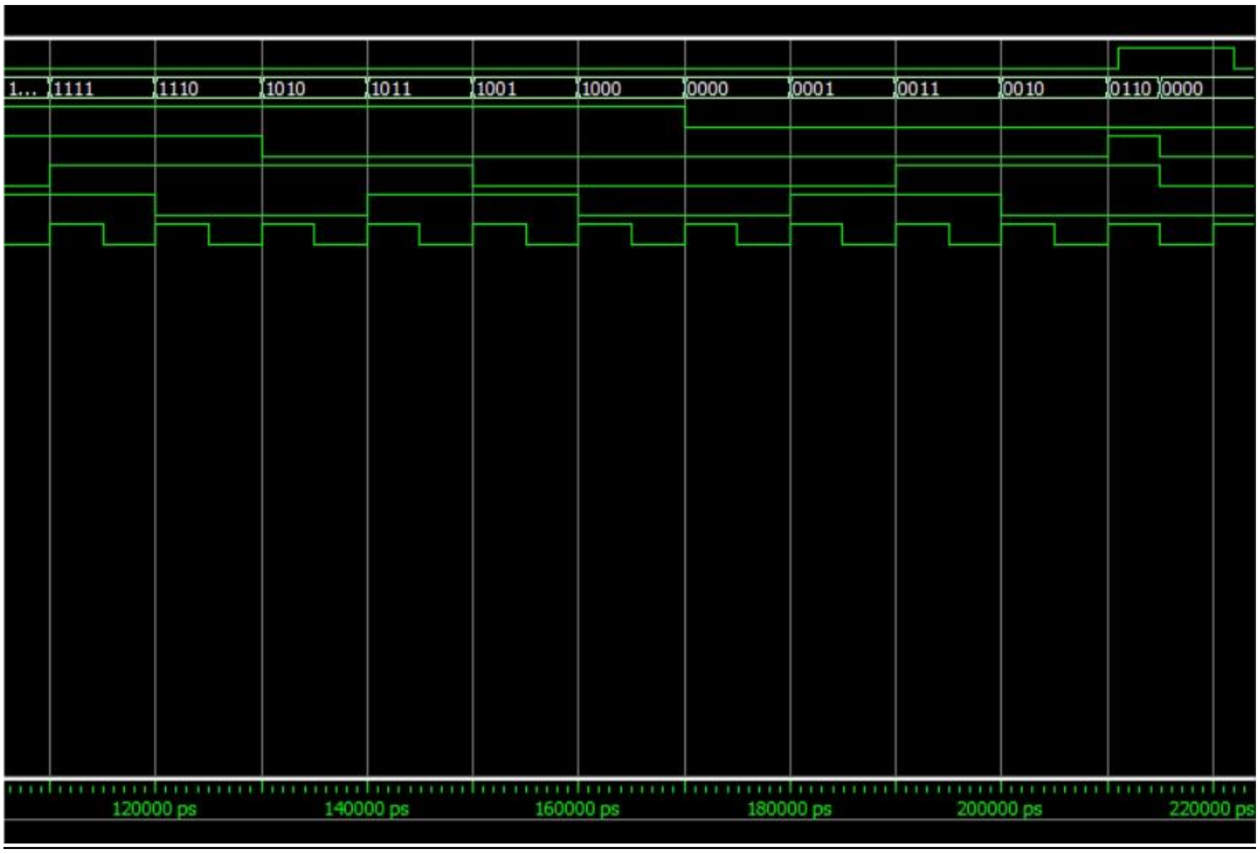
```
library ieee;
use ieee.std_logic_1164.all;
entity gray_counter_tst is
end gray_counter_tst;
architecture beh of gray_counter_tst is
component gray_counter is
port(clk: in std_logic;
  rst: in std_logic;
  gray_code: out std_logic_vector (3 downto 0));
end component;
 signal rst_s : std_logic;
signal 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:



Waveform Continuation:



Applications of Grey 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 applications.

Conclusion:
The design implementation of 4-Bit Grey Code counter is based upon a 4-bit encoder. However, the design can accommodate outputs for 1, 2 and 3-bit encoders at no extra required connection cost.

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
- http://www.computerscijournal.org/vol10no3/design-of-a-gray-encoder-and-counter-using-d-ffs-and-muxs/#:~:text=Encoder%20Design,bits%20is%20a%20trivial%20task.