

A Systematic Method to Generate Cyclic Gray Code Sequences with Constrained State-Space to Count Upto Arbitrary Even Numbers

Srinivasan Ramasubramanian
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721
srini@ece.arizona.edu

Monday July 13, 2009

1 Background

In summer 2007, Programmable Logic Design Line posted a problem on constructing Gray code counters to count up to arbitrary even numbers [PLD-1]. We developed a systematic method to construct such codes [Srini2007]. Recently, this article was published on Programmable Logic Design Line website [PLD-2]. From one of the comments on the article, we gathered that the problem of generating Gray code sequences to count up an arbitrary even number, N , using only the binary representations of 0 through $N - 1$ may still be open and this article develops a solution for the same.

2 Problem Statement

Design a Gray code counter to count up to a given number of words, N , using only the binary representations of numbers from 0 through $N - 1$.

3 A Solution

Designing a Gray code counter to count up to any power of 2 is straight-forward and several different sequences are possible. For our purpose, we will consider the most well-known method of generating k -bit Gray code sequence and use an interesting property of this sequence to obtain the desired solution.

3.1 k -Bit Code Sequence Using Mirroring

Let G_k denote a k -bit Gray code sequence. Let G'_k denote the reverse sequence of G_k . We may obtain a k -bit Gray code sequence from $k - 1$ bit Gray code sequence iteratively as follows:

$$G_k = \{0 \bullet G_{k-1}\} + \{1 \bullet G'_{k-1}\}$$

where the notations $\{0 \bullet G_{k-1}\}$ denotes the sequence obtained by adding 0 as the prefix to every element of G_{k-1} and $\{1 \bullet G'_{k-1}\}$ denotes the sequence obtained by adding 1 as the prefix to every element of G'_{k-1} . The $+$ notation indicates the order in which the sequences are merged. It is straight-forward to verify that G_k obtained as above is indeed a Gray code sequence.

3.2 k -Bit Cyclic Gray Codes

A Gray code sequence is said to be *cyclic* if the first element and the last element of the sequence differ in exactly one bit position. Interestingly, if we employ the above mirroring technique from $k = 2$ onwards, assuming $G_1 = \{0, 1\}$, we will obtain cyclic Gray codes. The sequence thus obtained has an interesting property: (1) the first element in G_k is 0; (2) the second element differs from the first element in the least significant bit; and (3) the last element differs from the first element in the most significant bit.

Observe that if every element of G_k is rotated left by L bits, the sequence will still retain its Gray code property. We denote the sequence obtained by rotating G_k left by L bits as $\{G_k \ll L\}$, where $L = 0, 1, \dots, k - 1$. In G_k , the first element and the second element different bit position 0. In $\{G_k \ll L\}$,

the first and the second element differ at position L . Similarly, in G_k , the first and the last element differ in position $k - 1$. In $\{G_k \ll L\}$, the first and the last element differ in position $(L + k - 1) \% k$, where $\%$ denotes the *modulo* operator. We will employ the above interesting relationship of the cyclic Gray codes to obtain our desired sequence.

3.3 Constructing the Desired Sequence

In this section, we construct the Gray code sequence for counting up to arbitrary even numbers that are not powers of two. We assume that cyclic Gray code sequence G_k for any k is available to us. Let $B = \lceil \log N \rceil$ denote the number of bits required to represent N numbers. Let $n_{B-1}n_{B-2} \dots n_1n_0$ denote the binary representation of N . As we consider only even numbers, n_0 is always 0.

Procedure ComputeCyclicGrayCodes

Input: An integer, N . (Binary representation: $n_{B-1}n_{B-2} \dots n_1n_0$)

Output: Gray code sequence, R .

1. Initialize $R \leftarrow \phi$; $leftShift \leftarrow 0$.
 2. For $i = 1$ through $B - 1$ do:
 - (a) If $n_i = 1$ then:
 - i. $F = \{0 \bullet \{G_i \ll leftShift\}\}$.
 - ii. $R \leftarrow \{1 \bullet R\}$.
 - iii. Insert sequence R into F between the first and the second element in F .
 - iv. $R \leftarrow F$.
 - v. $leftShift \leftarrow (leftShift + i - 1) \% i$.
 - (b) Otherwise, $R \leftarrow \{0 \bullet R\}$.
 3. Return R .
-

Figure 1: Procedure to compute cyclic Gray code sequence to count up to an arbitrary even number N , using only the binary representations of numbers from 0 through $N - 1$.

The procedure for constructing the desired Gray code sequence is shown in Figure 1. The idea behind generating the desired sequence is to use the cyclic Gray code sequences of smaller powers of two. We denote by desired sequence by R , initialized to an empty set. After every operation, we ensure that R remains a cyclic Gray code sequence. We keep track of the bit position in which the first element and the last element of R differ, which is denoted by $leftShift$. We begin with the least significant bit of the binary representation of N . If the value at a bit position i is 1, then we need to add a Gray code sequence of size 2^i to R . We increase the size of R in three steps. In the first step, we obtain the Gray code sequence of size 2^i using G_i , however we left shift all the elements by $leftShift$. We then prefix every element of $\{G_i \ll leftShift\}$ with 0. We denote the resultant sequence by F . In the second step, we prefix every element of R by 1. Observe a few interesting points here: (1) The elements of F and R have the same width; (2) The first element of F and the first element of R differ only in one bit. (3) R is still a Gray code sequence. (4) The last element of R and the second element of F differ only in one bit. Therefore, we may simply insert the sequence R between the first and the second elements of F , increasing the size of the Gray code sequence. We update R to be the increased sequence as obtained above. Since sequence G_i was left-shifted by $leftShift$ number of bits, the first element and the last element in F would now differ in bit position $(leftShift + i - 1) \% i$. We update $leftShift$ to $(leftShift + i - 1) \% i$. If the value of bit position is 0, then we do not need to increase the size of R . However, we need to increase the width of every element in R , hence we update the sequence by adding 0 as prefix to every element (Step 2b).

It is fairly straight-forward to prove that the above construction employs only binary values from 0 through $N - 1$, hence is left as an exercise for the reader.

4 Examples

We illustrate the working of the above procedure using four examples. We show the evolution of the desired sequence R as we complete Step 2 of the procedure for different values of i .

4.1 Example 1: $N = 10$

Binary representation of N : 1010.

$i = 1$:

$$\begin{aligned} F &= \{00, 01\} \\ R &= \{00, 01\} \\ \text{leftShift} &= 0 \end{aligned}$$

$i = 2$:

$$R = \{000, 001\}$$

$i = 3$:

$$\begin{aligned} F &= \{0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100\} \\ R &= \{0000, 1000, 1001, 0001, 0011, 0010, 0110, 0111, 0101, 0100\} \\ \text{leftShift} &= 2 \end{aligned}$$

4.2 Example 2: $N = 12$

Binary representation of N : 1100.

$i = 1$:

$$R = \phi$$

$i = 2$:

$$\begin{aligned} F &= \{000, 001, 011, 010\} \\ R &= \{000, 001, 011, 010\} \\ \text{leftShift} &= 1 \end{aligned}$$

$i = 3$:

$$\begin{aligned} F &= \{0000, 0010, 0110, 0100, 0101, 0111, 0011, 0001\} \\ R &= \{0000, 1000, 1001, 1011, 1010, 0010, 0110, 0100, 0101, 0111, 0011, 0001\} \\ \text{leftShift} &= 0 \end{aligned}$$

4.3 Example 3: $N = 14$

Binary representation of N : 1110.

$i = 1$:

$$\begin{aligned} F &= \{00, 01\} \\ R &= \{00, 01\} \\ \text{leftShift} &= 0 \end{aligned}$$

$i = 2$:

$$\begin{aligned} F &= \{000, 001, 011, 010\} \\ R &= \{000, 100, 101, 001, 011, 010\} \\ \text{leftShift} &= 1 \end{aligned}$$

$i = 3$:

$$\begin{aligned} F &= \{0000, 0010, 0110, 0100, 0101, 0111, 0011, 0001\} \\ R &= \{0000, 1000, 1100, 1101, 1001, 1011, 1010, 0010, 0110, 0100, 0101, 0111, 0011, 0001\} \\ \text{leftShift} &= 0 \end{aligned}$$

4.4 Example 4: $N = 26$

Binary representation of N : 11010.

$i = 1$:

$$F = \{00, 01\}$$

$$R = \{00, 01\}$$

$$leftShift = 0$$

$i = 2$:

$$R = \{000, 001\}$$

$i = 3$:

$$F = \{0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100\}$$

$$R = \{0000, 1000, 1001, 0001, 0011, 0010, 0110, 0111, 0101, 0100\}$$

$$leftShift = 2$$

$i = 4$:

$$F = \{00000, 00100, 01100, 01000, 01001, 01101, 00101, 00001, 00011,$$

$$00111, 01111, 01011, 01010, 01110, 00110, 00010\}$$

$$R = \{00000, 10000, 11000, 11001, 10001, 10011, 10010, 10110, 10111, 10101, 10100,$$

$$00100, 01100, 01000, 01001, 01101, 00101, 00001, 00011,$$

$$00111, 01111, 01011, 01010, 01110, 00110, 00010\}$$

$$leftShift = 1$$

5 Final Remarks

This article develops a systematic method to construct cyclic Gray code sequences for counting up to any given even number, N , using only the binary representations of numbers from 0 through $N - 1$. In order to construct the desired sequence, we employed the Gray code sequence for powers of 2 obtained using the mirroring technique. We note that we may employ any cyclic Gray code sequence in the procedure developed in this paper, however the number of bits by which a sequence is left-shifted needs to be appropriately adjusted depending on the sequence employed.

References

- PLD-1 An interesting Gray code FIFO counter question: <http://pldesignline.com/199701541>.
- Srini2007 S. Ramasubramanian, A systematic method to generate Gray code sequences to count up to arbitrary even numbers," <http://ece.arizona.edu/~srini/papers/Srini-Pulse-GrayCode.pdf>.
- PLD-2 A systematic method to generate Gray code: <http://www.pldesignline.com/218401093>.