The Journal of Engineering Research Vol. 6, No. 2 (2009) 12-19

A Less Complex Algorithmic Procedure for Computing Gray Codes

Afaq Ahmad[*a] and Mohammed M. Bait Suwailam[a]

[*a] *Department of Electrical and Computer Engineering, College of Engineering, Sultan Qaboos University, P.O. Box 33, Postal Code 123, Muscat, Sultanate of Oman*

Received 25 April 2008; accepted 26 October 2008

خطوات خوارزمية اقل تعقيدا لحساب رموز جري

أفاق احمد و محمد بيت سويلم

الخلاصة: يهدف هذا البحث الى تقديم طريقة جديدة سريعة لتوليد رموز جري بحجم ن- رقم ثنائي. لقد تم تقديم اشتقاق وخطوات تصميم و استخدام الخوارزمية الجديدة لتوليد سلسلة رمز جري المنعكسة وذات حجم ن- رقم ثنائي. إن الخوارزمية المطورة مقتبسة من حقيقة توليد و اختيار المواقع للحدود الصغرى المجموعة الحدود العامة للدالة البولينية للدالة المتكونة من ن متغير. تعطي الطريقة الناتجة حلا بصيغة قابلة للتطبيق. هذا بلإضافة إلى أن الطريقة المطورة تنتج حلولا ذات زمن تنفيذ أفضل و بذاكرة مستخدمه أقل حجما.

المفردات المفتاحية: رمز جري، الحدود الصغرى، الدالة البولينية، الخوارزميات، زمن التنفيذ، حجم الذاكرة، النظام الثنائي.

Abstract: The purpose of this paper is to present a new and faster algorithmic procedure for generating the *n*-bit Gray codes. Thereby, through this paper we have presented the derivation, design and implementation of a newly developed algorithm for the generation of an n-bit binary reflected Gray code sequences. The developed algorithm is stemmed from the fact of generating and properly placing the min-terms from the universal set of all the possible min-terms $[m_0\ m_1\ m_2\ \ldots.\ m_N]$ of Boolean function of n variables, where, $0 < N < 2^n-1$. The resulting algorithm is in concise form and trivial to implement. Furthermore, the developed algorithm is equipped with added attributes of optimizing of time and space while executed.

Keywords: Gray code, Min-terms, Boolean function, Algorithm, Processing time, Memory space, Binary

1. Introduction

Although many codes for example Binary Coded Decimal (BCD), Excess-3 Code, Hamming Code, Cyclic Redundancy Code (CRC), Check Sum and many others exist and are in use. But the Gray codes which are named for Frank Gray who patented the use of them in shaft encoders (Gray, 1953) due to its attribute of single distance only, which avoids ambiguous switching situations, is particularly used to handle safely and conveniently the control problems. The term Gray code is sometimes used to refer to any single-distance code, that is, one in which adjacent code words differ by 1 in one digit position only, This property can be seen in Table 1 which shows the Gray codes for size n = 1 to 4 bits.

Unlimited applications are accounted for the Gray code. Some of them are mentioned here as follows. However, more can be learned through the references pro vided as in the references (Sundberg, 1975; Ludman, 1981; Er, 1984, and Proskurowski and Ruskey, 1985; Lee, 1986; Conway, *et al.* 1989; Skiena, 1990; Press, *et al.* 1992; Etzion and Peterson, 1992; Hiltgen, *et al.* 1996; Savage, 1997; Ruskey, 1997; Guan and Dah-Jyu, 1998, Moshe and Tuvi, 1999; Black Paul 2004; Alan and Alessandro Mei, 2004; Jywe-Fei and Lai, 2005; Bitner, *et al.* 2005).

Gray codes were applied to mathematical puzzles before they became known to engineers. The Gray code arises naturally in many situations. Gray's interest in the code was related to what we would now call analog to digital conversion. The goal was to convert an integer value, represented as a voltage, into a series of pulses representing the same number in digital form. The technique, as described in Gray's patent, was to use the voltage being converted to displace vertically an electron beam that is being swept horizontally across the screen of a cathode ray tube. The screen has a mask etched on it that only allows the passage of the beam in certain places; a current is generated only when the beam passes through the mask. The passage of the beam will then give rise to a series of on/off conditions corresponding to the pattern of mask holes that it passes.

*Corresponding author's e-mail: afaq@squ.edu.om

Table 1. Gray code patterns of size n = 1 to 4 bits

Base 10	**Binary code**	**Gray code 4-bit**	**Gray code 3-bit**	**Gray code 2-bit**	**Gray code 1-bit**
0	0000	0000	000	00	0
1	0001	0001	001	01	1
2	0010	0011	011	11	
3	0011	0010	010		
4	0100	0110	110		
5	0101	0111	111		
6	0110	0101	101		
7	0111	0100	100		
8	1000	1100			
9	1001	1101			
10	1010	1111			
11	1011	1110			
12	1100	1010			
13	1101	1011			
14	1110	1001			
15	1111	1000			

Mechanical position sensors use Gray code to convert the angular position (angle-measuring devices) of a shaft to digital form. Gray codes were used in telegraphy. The Gray code also forms a Hamiltonian cycle on a hypercube, where each bit is seen as one dimension. In data transmission, Gray codes play an important role in error detection and correction. Solving puzzles such as the Tower of Hanoi and the Brain, the study of bell-ringing, analog-digital signal conversion, classifying of Venn diagrams, continuous space-filling curves, enhancing the resolution of spectrometer for a satellite application, labeling the axes of Karnaugh maps are the processes where Gray codes are used due to its uniqueness. Gray codes are also beneficial in Genetic Algorithms due to its incremental change property. Using Gray codes for addressing the memory results in saving of the power because a few address lines change as the program counter advances to the next location. Also, Gray codes are extensively used by digital system designers for passing multi-bit count information between synchronous logic that operates at different clock frequencies. In some numerical problems, Gray codes can be useful in situations of looping over many values of a bit. Furthermore, due to its attribute Gray code could be a good choice for the search of the optimal test-sequences in digital system testing.

Hence it can be said that the Gray codes which were originally designed to prevent spurious output from electromechanical switches. Today they are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.

Since the Gray code has enormous applications as mentioned above has many facet researches as is evident in surveying the literature (Gray, 1953; Sundberg, 1975; Ludman, 1981; Lee, 1986; Conway, *et al.* 1989; Skiena, 1990; Press, *et al.* 1992; Etzion Peterson, 1992; Hiltgen, *et al.* 1996; Savage, 1997; Ruskey, 1997; Guan Dah-Jyu, 1998, Moshe and Tuvi, 1999; Black Paul 2004; Alan and Alessandro Mei, 2004; Jywe-Fei and Lai, 2005; Bitner, *et al.* 1976; Er, 1984; Proskurowski and Ruskey, 1985; Ruskey, 1993; Dominique, 2000; Lassing, *et al.* 2003, Goddyn and Gvozdjak 2003 and Vajnovszki and Walsh 2006). It is clear through the literature survey that there had been much discovered and written about the Gray code; it is associated with many elegant circuits and algorithms. However, the algorithms generating the Gray code was still done with the crude techniques (Ruskey, 1993; Dominique, 2000; Lassing, *et al.* 2003, Goddyn, Gvozdjak 2003 and Vajnovszki and Walsh 2006). Researches are available in the literature only to script the faster codes but not much deviated from the existing crude algorithms for generating the Gray codes. This paper presents a new concept of generating the Gray code of n-bit size. The developed algorithm is stemmed from the fact of generating and properly placing the min-terms from the universal set of all the possible min-terms (m_0 m_1 m_2 …. m_N) of Boolean function of *n* variables, where, $0 < N < 2^n-1$. The resulting algorithm is in concise form and trivial to implement. We designed an efficient algorithm to write the codes which reduces the processing time and memory space requirements.

2. Gray Code Conversion - Conventional Approaches

No doubt, a simple recursive equation in mod (2) operation can convert a simple binary -…-8-4-2-1 codes to the Gray one. The hardware is also simple which is based on the bank of Exclusive-OR gates. To make the content of this paper more readable to audience the description of the generation procedure of the Gray code is given below:

To convert a binary number [b_{n-1} b_{n-2} …. b_1 b_0] to its corresponding Gray code (g_{n-1} g_{n-2} …. g_1 g_0), start at the left with the bit b_{n-1} (the n^{th}, most significant bit) and use the following recursive equations.

$$g_{n-1} = b_{n-1} \quad (1)$$

$$g_{n-2} = b_{n-1} \oplus b_{n-2} \quad (2)$$

Or, in general for i^{th} bit where *i* varies as, $2 < i < n$, the following Eq. (3) can be used.

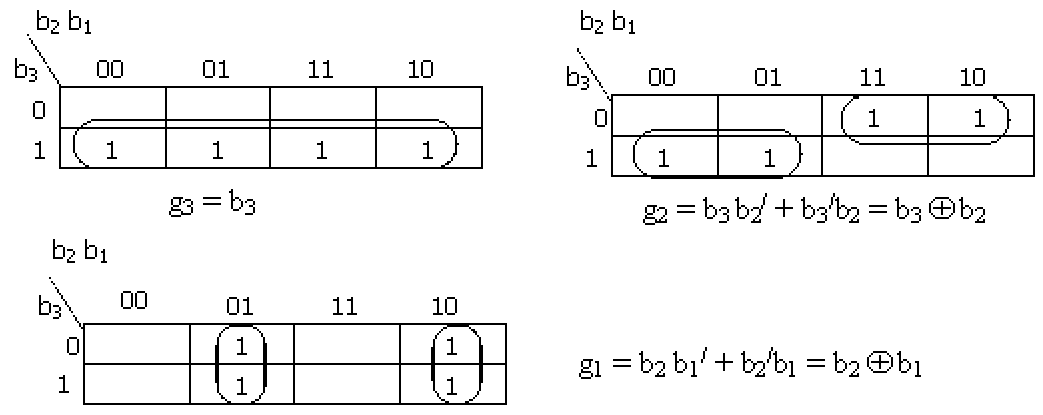
$$g_{n-I} = b_{n-i+1} \oplus b_{n-I} \quad (3)$$

The above Eqs. (1) and (3) are illustrated for a 3-bit Gray code encoder in an example below.

Example 1:

Let a Binary code [b_3 b_2 b_1] = [1 0 0].

So, $g_3 = b_3 = 1$; $g_2 = b_3 \oplus b_2 = (1 + 0) \bmod 2 = 1$; $g_1 = b_2 \oplus b_1 = (0 + 0) \bmod 2 = 0$. Therefore, Binary code [b_3 b_2 b_1] = [1 0 0] when encoded in Gray code gives $\rightarrow$ [g_3 g_2 g_1] = [1 1 0].



$b_2 b_1$
b_3
00 01 11 10
0
1
1 1 1 1
$g_3 = b_3$
1 1
1 1
$g_2 = b_3 b_2' + b_3' b_2 = b_3 \oplus b_2$
1 1
1 1
$g_1 = b_2 b_1' + b_2' b_1 = b_2 \oplus b_1$

Figure 1. K-maps and minimized Boolean functions for g_3, g_2 and g_1

The derivation of the generalized Eqs. (1) - (3) can be computed as below.

From Table 1 above, a set of the following Boolean functions expressed in the form of min-terms $\Sigma\, m_i$ where i varies from 0 to 7, can be derived as:

$$g_3\ (b_3, b_2, b_1) = \Sigma\ (m_4, m_5, m_6, m_7) \qquad (4)$$

$$g_2\ (b_3, b_2, b_1) = \Sigma\ (m_2, m_3, m_4, m_5) \qquad (5)$$

$$g_1\ (b_3, b_2, b_1) = \Sigma\ (m_1, m_2, m_5, m_6) \qquad (6)$$

The minimized version of above Boolean functions using the K-maps and hence the implementations of those minimized functions are as given in Figs. 1 and 2 respec-

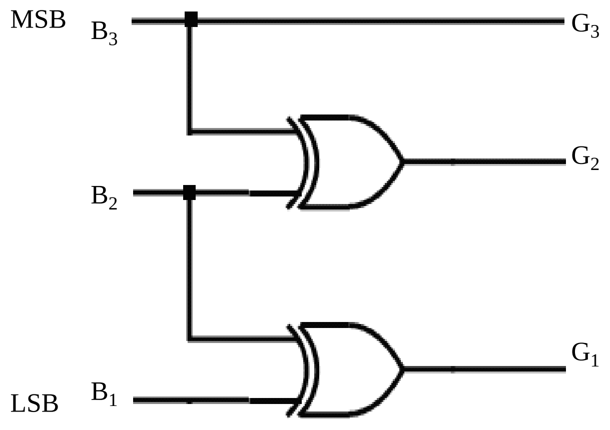


Figure 2. A 3-bit Binary to Gray Converter Circuit

tively.

Similarly, the reversal of Gray code bits [$g_n\ g_{n-1} \dots g_2.\ g_1$] again into -…-8-4-2-1 weighted binary code bits [$b_n\ b_{n-1} \dots b_2.\ b_1$] can be performed by using the equations as given below:

$b_n = g_n$ (7)

$b_{n-1} = b_n \oplus g_{n-1}$

(8)

$b_1 = b_2 \oplus g_1$

.

.

(9)

Or, in general, to compute the binary code bits b_{n-1}, … , b_2, and. b_1 the following recursion equation can be used where, i varies from 1 to n-1.

$b_{n-i} = b_{n-i+1} \oplus g_{n-i}$

So, $b_3 = g_3 = 1$; $b_2 = b_3 \oplus g_2 = (1 + 1) \bmod 2 = 0$; $b_1 = b_2 \oplus g_1 = (0 + 0) \bmod 2 = 0$. Therefore, Gray code [$g_3\ g_2\ g_1$] = [1 0 0] when encoded in Binary code [$b_3\ b_2\ b_1$] gives → = [1 1 0].

(10)

Example 2:

Let a Gray code [$g_3\ g_2\ g_1$] = [1 1 0].

The derivation of the generalized Eqs. (7) - (10) can be easily computed as:

From Table 1 above, a set of the following Boolean functions expressed in the form of min-terms $\Sigma\, m_i$ where i varies from 0 to 7, can be derived as:

$$b_3\ (g_3, g_2, g_1) = \Sigma\ (m_4, m_5, m_6, m_7) \qquad (11)$$

$$b_2\ (g_3, g_2, g_1) = \Sigma\ (m_2, m_3, m_4, m_5) \qquad (12)$$

$$b_1\ (g_3, g_2, g_1) = \Sigma\ (m_1, m_2, m_4, m_7) \qquad (13)$$

The above Boolean functions can be minimized using the K-maps as demonstrated below (see Fig. 3). Whereas, the implementation of those minimized Boolean functions is shown in Fig. 4.

Since the Gray code encoder needs first to obtain the

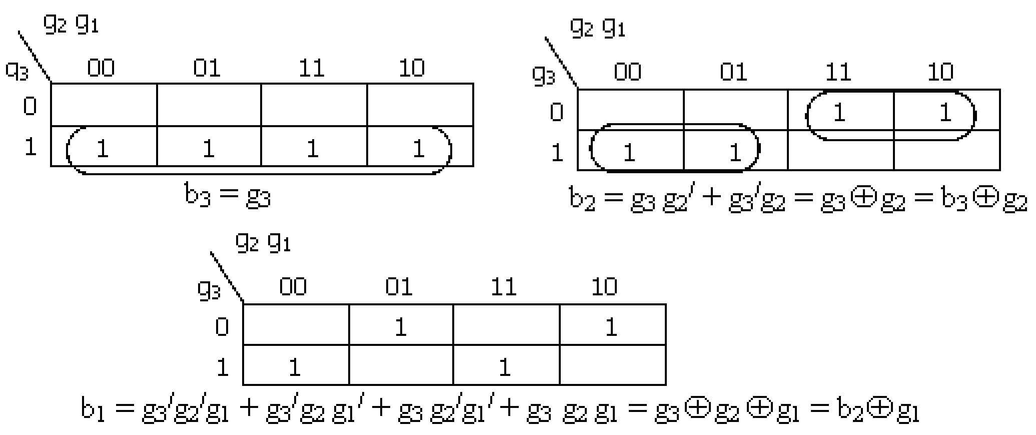

g2 g1
g3
00 01 11 10
0
1
$b_3 = g_3$
$b_2 = g_3 g_2' + g_3' g_2 = g_3 \oplus g_2 = b_3 \oplus g_2$
$b_1 = g_3' g_2' g_1 + g_3' g_2 g_1' + g_3 g_2' g_1' + g_3 g_2 g_1 = g_3 \oplus g_2 \oplus g_1 = b_2 \oplus g_1$

Figure 3. K-maps and minimized Boolean functions for b_3, b_2 and b_1

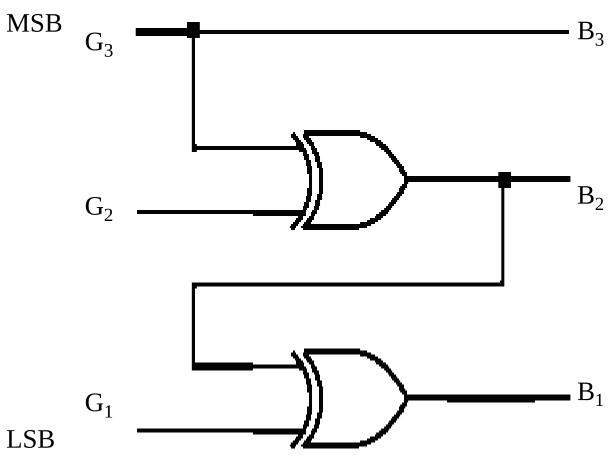


Figure 4. A 3-bit Gray to Binary converter circuit

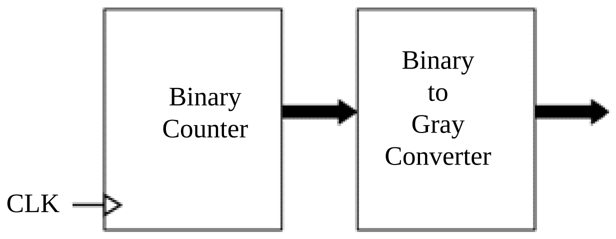


Figure 5. Black model of Binary to Gray code conversion

binary data (through a binary counter) before generating the Gray code. This process requires 2-stages as shown in Fig. 5. Thus, it is imperative to derive a mechanism to avoid this situation which needs much time and space to implement. The ensuing section is a consequence to it.

3. Computing Gray Codes - Proposed Methodology

If we look to the Table 2 and 3 which lists the Gray codes, respective min-terms and equivalent decimals for n = 1 to 6 forced us to derive the following conclusions.

1. Gray code of size n has a specific pattern relationship between min-terms of the Gray code of its predecessor code of size n-1.
2. The Gray code of size n can be directly scripted using the n-bit K-map where the min-terms cells are to be read clock-wise and down to the row as explained in the Fig. 6.

Table 2. Decimal, Gray code, and (min-terms): For, n = 1 to 5

Decimal	5-bit	4-bit	3-bit	2-bit	1-bit
0	00000 (m_0)	0000 (m_0)	000 (m_0)	00 (m_0)	00 (m_0)
1	00001 (m_1)	0001 (m_1)	001 (m_1)	01 (m_1)	01 (m_1)
3	00011 (m_3)	0011 (m_3)	011 (m_3)	11 (m_3)	
2	00010 (m_2)	0010 (m_2)	010 (m_2)	10 (m_2)	
6	00110 (m_6)	0110 (m_6)	110 (m_6)		
7	00111 (m_7)	0111 (m_7)	111 (m_7)		
5	00101 (m_5)	0101 (m_5)	101 (m_5)		
4	00100 (m_4)	0100 (m_4)	100 (m_4)		
12	01100 (m_{12})	1100 (m_{12})			
13	01101 (m_{13})	1101 (m_{13})			
15	01111 (m_{15})	1111 (m_{15})			
14	01110 (m_{14})	1110 (m_{14})			
10	01010 (m_{10})	1010 (m_{10})			
11	01011 (m_{11})	1011 (m_{11})			
9	01001 (m_9)	1001 (m_9)			
8	01000 (m_8)	1000 (m_8)			
24	11000 (m_{24})				
25	11001 (m_{25})				
27	11011 (m_{27})				
26	11010 (m_{26})				
30	11110 (m_{30})				
31	11111 (m_{31})				
29	11101 (m_{29})				
28	11100 (m_{28})				
20	10100 (m_{20})				
21	10101 (m_{21})				
23	10111 (m_{23})				
22	10110 (m_{22})				
18	10010 (m_{18})				
19	10011 (m_{19})				
17	10001 (m_{17})				
16	10000 (m_{16})				

The example of Fig. 6 is a 4-variable K-map used to

Table 3. Decimal, Gray code, and (min-terms): For, $n = 6$

Decimal	Gray code (m)	Decimal	Gray code (m)
0	000000 (m_0)	48	110000 (m_{48})
1	00001 (m_1)	49	110001 (m_{49})
3	000011 (m_3)	51	110011 (m_{51})
2	000010 (m_2)	50	110010 (m_{50})
6	000110 (m_6)	54	110110 (m_{54})
7	000101 (m_7)	55	110111 (m_{55})
5	000101 (m_5)	53	110101 (m_{53})
4	000100 (m_4)	52	110100 (m_{52})
12	001100 (m_{12})	60	111100 (m_{60})
13	001101 (m_{13})	61	111101 (m_{61})
15	001111 (m_{15})	63	111111 (m_{63})
14	001110 (m_{14})	62	111110 (m_{62})
10	001010 (m_{10})	58	111010 (m_{58})
11	001011 (m_{11})	59	111011 (m_{59})
9	001001 (m_9)	57	111001 (m_{57})
8	001000 (m_8)	56	111000 (m_{56})
24	011000 (m_{24})	40	101000 (m_{40})
25	011001 (m_{25})	41	101001 (m_{41})
27	011011 (m_{27})	43	101011 (m_{43})
26	011010 (m_{26})	42	101010 (m_{42})
30	011110 (m_{30})	46	101110 (m_{46})
31	011111 (m_{31})	47	101111 (m_{47})
29	011101 (m_{29})	45	101101 (m_{45})
28	011100 (m_{28})	44	101100 (m_{44})
20	010100 (m_{20})	36	100100 (m_{36})
21	010101 (m_{21})	37	100101 (m_{37})
23	010111 (m_{23})	39	100111 (m_{39})
22	010110 (m_{22})	38	100110 (m_{38})
18	010010 (m_{18})	34	100010 (m_{34})
19	010011 (m_{19})	35	100011 (m_{35})
17	010001 (m_{17})	33	100001 (m_{33})
16	010000 (m_{16})	32	100000 (m_{32})

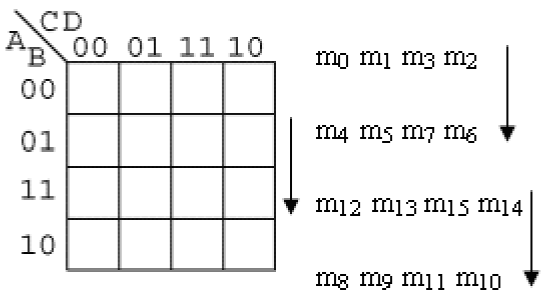


Figure 6. A 4-variable *K*-map

generate Gray code which reads as $[m_0\ m_1\ m_3\ m_2\ m_6\ m_7\ m_5\ m_4\ m_{12}\ m_{13}\ m_{15}\ m_{14}\ m_{10}\ m_{11}\ m_9\ m_8]^{/}$. The derivation 2 is not feasible since generating the K-map for higher variables are difficult to manipulate. Therefore, the derivation 1 is the point of our work.

4. Proposed Algorithm

The following Eqs. (14) to (18) describe the Gray codes for bit size 1 to 5 respectively. A specific relation between the min-terms patterns is visible and summarized in the form of a theorem below:

$$G(1) = [m_0 \mid m_1]^{/} \tag{14}$$

$$G(2) = [m_0\ \ m_1 \mid m_3\ m_2]^{/} = [G(1);\ [m_3\ m_2]^{/}] \tag{15}$$

$$\begin{aligned} G(3) &= [m_0\ \ m_1\ \ m_3\ m_2 \mid m_6\ m_7\ m_5\ m_4]^{/} \\ &= [G(2);\ [m_6\ m_7\ m_5\ m_4]^{/}] \end{aligned} \tag{16}$$

$$\begin{aligned} G(4) = {} & [m_0\ \ m_1\ \ m_3\ m_2\ \ m_6\ m_7\ m_5\ m_4 \mid \\ & m_{12}\ m_{13}\ m_{15}\ m_{14}\ m_{10}\ m_{11}\ m_9\ m_8]^{/} \\ = {} & [G(3);\ [m_{12}\ m_{13}\ m_{15}\ m_{14}\ m_{10}\ m_{11}\ m_9\ m_8]^{/} \end{aligned} \tag{17}$$

$$\begin{aligned} G(5)_{SETI} &= [m_0\ \ m_1\ \ m_3\ \ m_2\ \ m_6\ \ m_7\ \ m_5\ \ m_4\ \ m_{12} \\ & \quad m_{13}\ \ m_{15}\ \ m_{14}\ \ m_{10}\ \ m_{11}\ \ m_9\ \ m_8]^{/} = G(4) \\ G(5)_{SET\,II} &= [m_{24}\ m_{25}\ \ m_{27}\ m_{26}\ m_{30}\ m_{31}\ m_{29} \\ & \quad m_{28}\ \ m_{20}\ \ m_{21}\ \ m_{23}\ m_{22}\ \ m_{18}\ m_{19}\ \ m_{17}\ \ m_{16}]^{/} \\ G(5) &= [G(4);\ [m_{24}\ \ m_{25}\ \ m_{27}\ m_{26}\ \ m_{30} \\ & \quad m_{31}\ \ m_{29}\ m_{28}\ \ m_{20}\ m_{21}\ \ m_{23}\ \ m_{22}\ \ m_{18}\ \ m_{19} \\ & \quad m_{17}\ m_{16}]^{/}] \end{aligned} \tag{18}$$

Theorem 1:

G (n) is a matrix of order 2^n *x n* in binary format of n-bit forming 2^n min-terms of n variables. By analyzing the patterns we reach to the conclusion that $G(n)$ can be obtained first by writing the min-terms of G(n-1) then appending the min-terms by advancing each of the min-terms starting from the last to the first by a value of 2^{n-1}.

Proof:

Proof is as visible through the Eqs. from (14) to (18).

An algorithm is designed on the basis of the study of Theorem 1 and is as given below.

Algorithm

STEP 0:
START by inputting the bit size (n) of the Gray code to be generated;

STEP 1:
Initialize a vector V = [0 1];

STEP 2:
Count a loop for k = 1 to n;

STEP 3:
Check that k = n or not, if YES GO TO **STEP 8**;

STEP 4:
Increment the counter k by 1 i.e. k = k +1;

STEP 5:

```
        For i = 0 to $2^{n-1}-1$;
        Vi = [V($2^{n-1}$-i) + $2^{n-1}$];

STEP 6:
        Modify vector V as V = [V, Vi];
STEP 7:
        GO TO STEP 3;
        Convert $V^{/}$ into binary format i.e. $(V^{/})_{10} \longrightarrow (V^{/})_2$ =
Gray code of n-bit size i.e. a matrix G (n) of size (N+1)
by n;

STEP 8:
        Transpose V i.e. $V^{/}$;

STEP 9:
STEP 10:
        Output G(n);

STEP 11:
        STOP
```

A Debug Example:

Table 4. A debug test of algorithm for, *n* = 3

i	*V$_i$*	*V*	*$(V^{/})_{base\ 2}$*
-		**[0 1]**	-
0 to 1	[3 2]	[0 1 3 2]	-
0 to 3	[6 7 5 4]	[0 1 2 3 6 7 5 4]	-
			000
			001
			011
			010
			110
			111
			101
			100

Example 3:

To elaborate the computing of variables in the above mentioned steps a debug test of the above algorithm is carried out for *n* = 3 and is presented below in Table 4.

5. Implementation of Algorithm

The above designed algorithm is implemented using the MATLAB code. The out put of the m-file with the name "gray_generator_proposed" can be visualized as shown in Fig. 7. A sample output of the program shown in the figure is only for *n* = 4. This is provided just to make it more readable, otherwise the output for the higher values of n will require much space to present. Similarly, we also encoded the conventional method of Gray code conversion into MATLAB script. To compare the efficiency amongst these two approaches we run both of the programs for *n* = 2 to 10 and some of the results are presented and made available through Figs. 8 to 9 in the ensuing section below.

```
>>gray_generator_proposed
Please enter a valid positive integer ( > 1) : 4
Gran output (s)
==================
        0       0       0       0
        0       0       0       1
        0       0       1       1
        0       0       1       0
        0       1       1       0
        0       1       1       1
        0       1       0       1
        0       1       0       0
        1       1       0       0
        1       1       0       1
        1       1       1       1
        1       1       1       0
        1       0       1       0
        1       0       1       1
        1       0       0       1
        1       0       0       1
```

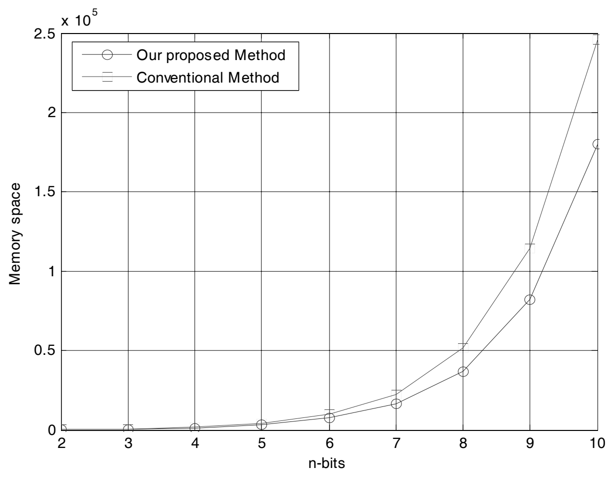


Figure 8. Memory requirements comparison

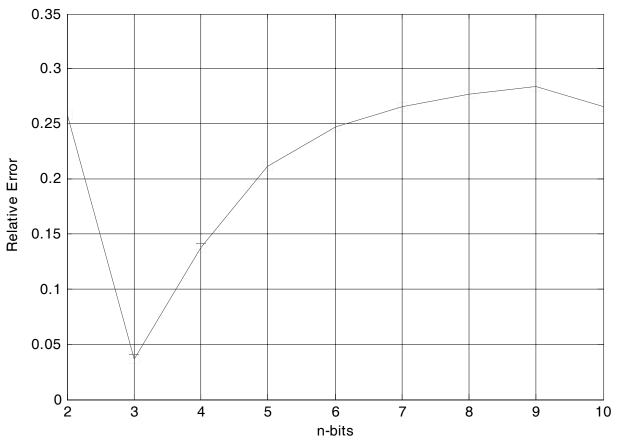


Figure 9. Relative error judgment

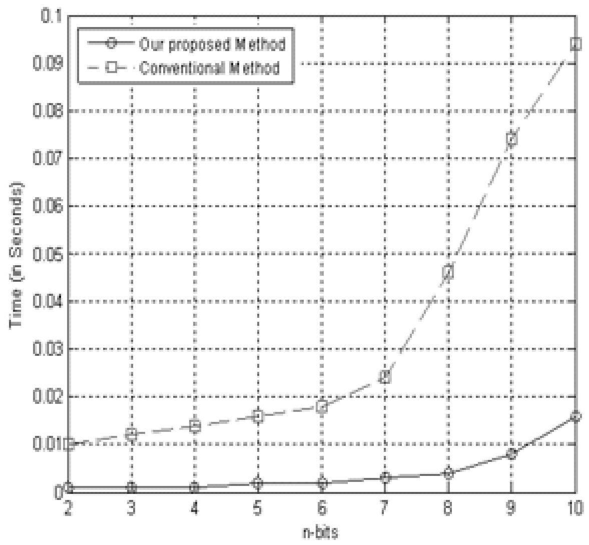


Figure 10. Run time requirements comparison

Elapsed time is 0.000000 seconds.

Figure 7. MATLAB command window output

6. Results

MATLAB 7.0 on a P 4 CPU 1.5 GHz, 512 MB of RAM is used as the bench marks for testing the codes for both of the computing methodologies. The processing time and memory space required to implement both of the programs (the conventional and the proposed one) are recorded while running the programs.

The results for a subset of the study, for n = 2 to 10 where the memory requirement comparison of the conventional approach of Gray code converter and the proposed approach is presented in Fig. 8. The memory requirement is mapped in bytes. Further, to judge the efficiency of the proposed algorithm, a relative error plot is shown in Fig. 9. With respect to the memory requirements, the behavior of the relative error shown in Fig. 9, demonstrates almost an exponential characteristic except for n = 2. And, this exception for (n = 2) is because of that the implementing the Gray code generator using conventional approach requires significant more memory space than the proposed approach of generating the Gray codes for n = 2. The run time requirements comparison is demonstrated through Fig. 10.

7. Conclusions

A time space optimal algorithmic procedure to generate Gray code-words of any bit length n is presented through this paper. The comparative study reveals that the proposed approach is not only faster but also, requires about 25% less memory space on average while compared with the conventional method of Gray code-word generation technique. Since Gray code is widely used for on line system monitoring hooked with sensors and with on board systems and hence these two parameters (space and time requirements) are very critical in these applications. Therefore this proposed algorithmic procedure is more advantageous.

8. References

Alan, A., Bertossi, Alessandro, Mei., 2004, "Time and Work Optimal Simulation of Basic Reconfigurable Meshes on Hypercubes," J. of Parallel and Distributed Computing, Vol. 64(1), pp.173 - 180.

Bitner, J., Ehrlich, G. and Reingold, E., 1976, 'Efficient Generation of the Binary Reflected Gray Code and its Applications," Communications of the ACM, Vol. 19(9), pp. 517-521.

Black Paul, E., (2004), "Gray Code," NIST National Institute of Standards and Technology.

Conway, J., Sloane, N. and Wilks, A., 1989, "Gray Codes and Reflection Groups," Graphs and Combinatorial, Vol. 5, pp. 315-325.

Dominique Roelants van Baronaigien., 2000, "A Loopless Gray-Code Algorithm for Listing K-Array Trees," J. of Algorithms, Vol. 35(1), pp.100-107.

Er, M.C., 1984, "On Generating the N-Array Reflected Gray Codes," IEEE transactions on computers, Vol. 33, pp. 739-741.

Etzion, T., Paterson, K.G., 1996, "Near Optimal Single-Track Gray Codes," IEEE Trans. on Information Theory, Vol. 42(3), pp. 779-789.

Goddyn, L., Gvozdjak, P., 2003, "Binary Gray Codes with Long Bit Runs," Electron. J. Combin. Vol. 10, pp. 1-10.

Gray, F., 1953, "Pulse Code Communication," United States Patent Number 2, 632, 058.

Guan Dah-Jyu., 1998, "Generalized Gray Codes with Applications," Proceeding National Science Council of Republic of China, Vol. A(22), pp. 841 - 848.

Hiltgen, P., Paterson, K. G. and Brandestini, M., 1996, "Single-Track Gray Codes," IEEE Trans. on Information Theory, Vol. 42(5), pp. 1555-1561.

Jywe-Fei Fang , Kuan-Chou Lai, 2005, "Embedding the Incomplete Hypercube in Books," Information Processing Letters, Vol. 96(1), pp. 1-6.

Lassing, E. G. Strom, T., Ottosson. and Agrell, E., (2003), "Computation of the Exact Bit Error rate of Coherent M-array PSK with Gray Code Bit Mapping," IEEE Transactions on Communications, Vol. 51(11), pp. 1758-1760.

Lee, P. J., 1986, "Computation of the Bit Error Rate of Coherent M-array PSK with Gray Code Bit Mapping," IEEE Transactions on Communications, Vol. COM-34(5), pp. 488-491.

Ludman, J. E., 1981, "Gray Code Generation for MPSK Signals," IEEE Transactions on Communications, Vol. COM-29(10), pp. 1519-1522.

Moshe Schwartz and Tuvi Etzion, 1999, "The Structure of Single-Track Gray Codes," IEEE Transactions on Information Theory, Vol. 45(7), pp. 2383 - 2396.

Press, W.H., Flannery, B.P., Teukolsky, S. A. and Vetterling, W.T., 1992, "Gray Codes, Numerical Recipes in Fortran," Cambridge, England: Cambridge University Press, pp. 886-888.

Proskurowski, A. and Ruskey, F., '1985, "Binary Tree Gray Codes," J. Algorithms 6, pp. 225-238.

Ruskey, F., 1997, "A Survey of Venn Diagrams," Elec. J. of Comb, a special issue.

Ruskey, F., 1993, "Simple Combinatorial Gray Codes Constructed by Reversing Sub-lists," Proceedings of the 4th International Symposium on Algorithms and Computation, p. 201-208, December 15-17.

Savage, Carla., 1997, "A Survey of Combinatorial Gray Codes," Society of Industrial and Applied Mathematics Review, Vol. 39, pp. 605-629.

Skiena, S., 1990, "Gray Code, Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica," Reading, MA: Addison-Wesley, pp. 42-43 and 149.

Sundberg, C. E., 1975, "Bit Error Probability Properties of Gray-coded M.P.S.K Signals," IEE Electronics Letters, Vol. 11(22), pp. 542-544.

Vajnovszki, V. and Walsh, T.R., 2006, "A Loop-free two-