

# Use of Gray Decoding for Implementation of Symmetric Functions

Osnat Keren<sup>1</sup> Ilya Levin<sup>2</sup> and Radomir S. Stankovic<sup>3</sup>

<sup>1</sup> Bar-Ilan University, Israel, [kereno@eng.biu.ac.il](mailto:kereno@eng.biu.ac.il)

<sup>2</sup> Tel-Aviv University, Israel, [ilia1@post.tau.ac.il](mailto:ilia1@post.tau.ac.il)

<sup>3</sup> Nis University, Serbia, [radomir.stankovic@gmail.com](mailto:radomir.stankovic@gmail.com)

**Abstract.** We study a problem of reduction of the number of product terms in representation of totally symmetric Boolean functions by Sum of Products (SOP) and Fixed Polarity Reed-Muller (FPRM) expansions. We propose a method, based on the Gray decoding, for reduction of the number of product terms, and, consequently, the implementation cost of the symmetric functions. The method is founded on the principles of linear transformations of the input variables of an initial function. It provides significant simplification both of the SOPs and the FPRMs representations of the functions. Mathematical analysis as well as experimental results demonstrate the efficiency of the proposed method.

## 1 Introduction

Linearization of switching functions based on linear transformation of variables is a classical method of optimization in circuit synthesis originating already in 1958 [22]. It has been recently efficiently exploited by several authors and discussed for different aspects due to its:

1. *Effectiveness.* The method provides considerable simplification of the representation of functions with respect to different optimization criteria.
2. *Simplicity of the implementation.* The overhead comprises EXOR circuits required to perform the selected linear combination of variables. The overhead is usually quite negligible compared to the overall complexity of the implementation [12].

The linearization can be performed over different data structures used to represent functions. For example, it has been performed over Sum-of-Product (SOP) expressions [10, 13, 15, 29], AND-EXOR expressions [5], word-level expressions [28] as well as decision diagrams [7, 14, 18].

In spectral techniques, the linearization is studied as a mean to reduce the number of non-zero coefficients in spectral expressions for discrete functions [8], [12]. In [12, 20], and [21] the extensions to multiple-valued logic functions are discussed. The complexity of determining an optimal non-singular binary matrix that defines the optimal linear transformation of variables is NP-complete. For this reason a number of alternative strategies have been suggested in exploiting this method.

For achieving the exact optimum, some restrictions should to be made on the number of variables in functions processed. For example, it has been reported in [7] that the complete search over all possible linear transformations is feasible for functions up to seven variables within reasonable space and time resources. Another strategy in using the linearization is to apply the method for particular classes of functions. For instance, in [10, 28] a method has been used for specific circuits, such as  $n$ -bit adders. The optimal linear transform has been found for this adder.

Sometimes, nearly optimal solutions can be provided by deterministic algorithms if analysis of additional information about the functions is available ([8, 12] and [14] and references therein).

In this paper, we develop a compromising approach. We deal with symmetric Boolean functions and demonstrate that for such functions an efficient linear transformation of variables can be determined analytically without intense computations. In particular, we show that a Gray decoding of the input variables provides a significant reduction in the number of product terms.

Symmetric Boolean functions represent an important fraction of Boolean functions. They are characterized by the fact that their outputs only depend on the Hamming weights of their inputs. These functions can be represented in a compact way both for their algebraic normal forms and for their value vectors. As symmetric functions are the only functions having a known implementation with a number of gates which is linear in the number of input variables [1], they might be good candidates in term of implementation complexity. There are efficient circuit-based methods and complete BDD-based methods for identifying symmetries of completely and incompletely specified functions [11, 17, 19, 23, 30, 33].

In last several years, symmetric functions have been studied from different aspects. Optimal Fixed Polarity Reed-Muller (FPRM) expansions for totally symmetric functions are discussed in [4, 32] and references therein. A lower bound on the number of gates in conjunctive (disjunctive) normal form representation of symmetric Boolean functions is given in [31] and a method for generating a minimal SOP cover is presented in [3]. A multilevel synthesis of symmetric functions which exploits the disjoint decomposability and weight dependency of the functions is presented in [16] and a mapping of symmetric and partially symmetric functions to the CA-type FPGAs was suggested in [2]. A new expansion of symmetric functions and their application to non-disjoint functional decompositions for LUT-type FPGAs is presented in [26].

In this paper we study a specific case of symmetric functions, and show that for these functions, a linear transformation based on Gray decoding of the input variables is very effective. We show that the Gray decoding almost always reduces the complexity in terms of the following three criteria: the number of gates in two-level realization, the number of FPRM terms and the number of FPGA LUTs. Additionally we show that not only symmetric functions but also partially symmetric functions may be efficiently implemented by using the proposed technique. This technique can be considered as a combination of a well-

known classical decomposition technique of the partially symmetric functions ([24]) with the proposed linear transformation.

The paper is organized as follows. Section 2 gives basic definitions of symmetric Boolean functions and Gray codes. Section 3 presents the implementation of a symmetric function as a superposition of a Gray decoder and a non-linear function. Section 4 presents an illustrative example discussing in detail application of the proposed method. In Section 5 we discuss features of the proposed method and prove that the solutions produced can never increase complexity of representation of SOPs compared to the given initial representations. In Section 6 we discuss the use of the Gray decoding for partially symmetric functions. Section 7 contains experimental results and Section 8 concludes the paper.

## 2 Preliminaries

### 2.1 Totally symmetric functions

Let  $f(x) = f(x_{n-1}, \dots, x_0)$  a Boolean function of  $n \geq 2$  inputs and a single output. The function  $f$  is *symmetric* in  $x_i$  and  $x_j$  iff

$$f(x_{n-1} \dots x_i \dots x_j \dots x_0) = f(x_{n-1} \dots x_j \dots x_i \dots x_0). \quad (1)$$

The function  $f$  is *totally* symmetric iff it is symmetric in all pairs of its variables.

A function  $f(x) = S_i(x)$  is called an *elementary* symmetric function with working parameter  $i$  iff

$$S_i(x) = \begin{cases} 1 & \|x\| = i \\ 0 & \text{otherwise} \end{cases}$$

where  $\|x\|$  is the Hamming weight of  $x$ . There are  $n + 1$  elementary symmetric functions satisfying

$$\sum_x S_i(x) S_j(x) = \begin{cases} \binom{n}{i} & i = j \\ 0 & \text{otherwise} \end{cases}.$$

Any symmetric function can be represented as a linear combination of elementary symmetric functions, i.e.  $f(x) = \bigoplus_{i=0}^n a_i S_i(x)$  where  $a_i \in \{0, 1\}$ . Hence, there are  $2^{n+1}$  symmetric functions out of  $2^{2^n}$  functions.

*Example 1.* Consider an elementary 5-inputs symmetric function  $f(x) = S_3(x)$ . The K-map of the function is given in Table 3. The minimal SOP representation of the function consists of 10 minterms of 5 literals.

A Fixed Polarity Reed-Muller (FPRM) expansion is an EXOR of product terms, where no two products consists of the same variables and each variable appears in complemented or un-complemented form, but not in both [25]. In matrix notation [1], the FPRM expansion of a function  $f(x_{n-1}, \dots, x_0)$  with a given polarity vector  $h = (h_{n-1}, \dots, h_1, h_0)$ , is defined as

$$f(x_{n-1}, \dots, x_0) = \left( \bigotimes_{i=0}^{n-1} [1, x_{n-1-i}^{h_{n-1-i}}] \right) \left( \bigotimes_{i=0}^{n-1} R^{h_{n-1-i}}(1) \right) F$$

where  $\otimes$  is a Kronecker product,

$$x_i^{h_i} = \begin{cases} x_i & \text{if } h_i = 0 \\ x'_i & \text{otherwise} \end{cases}$$

and

$$R^{h_i}(1) = \begin{cases} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} & \text{if } h_i = 0 \\ \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} & \text{otherwise} \end{cases}$$

and  $F$  is the truth vector. The number of product terms in the FPRM depends on the polarity vector.

*Example 2.* The FPRM expansion of the 3-out-of-5 function in Example 1 with a positive polarity ( $h = 0$ ) comprises 10 terms,

$$\begin{aligned} f = & x_4x_3x_2 \oplus x_4x_3x_1 \oplus x_4x_2x_1 \oplus x_3x_2x_1 \oplus x_4x_3x_0 \\ & \oplus x_4x_2x_0 \oplus x_3x_2x_0 \oplus x_4x_1x_0 \oplus x_3x_1x_0 \oplus x_2x_1x_0. \end{aligned}$$

The positive polarity produces the minimal number of terms, all the other 31 polarity vectors produces FPRM expansions of at least 16 product terms.

## 2.2 Gray code

The reflected binary code, also known as Gray code after Frank Gray [6], is used for listing  $n$ -bit binary numbers so that successive numbers differ in exactly one bit position. The definition of the Gray encoding and decoding is the following: Elements of a binary vector of length  $n$ ,  $z = (z_{n-1}, \dots, z_0)$  and the vector  $x = (x_{n-1}, \dots, x_0)$  derived by Gray encoding are related as

$$x_i = \begin{cases} z_i & i = n - 1 \\ z_i \oplus z_{i+1} & \text{otherwise} \end{cases}$$

and

$$z_i = \begin{cases} x_i & i = n - 1 \\ x_i \oplus z_{i+1} & \text{otherwise} \end{cases}.$$

This relation can be written using matrix notation as  $x = G_E z$  and  $z = G_D x$  where  $G_E = (\tau_{n-1}, \dots, \tau_1, \tau_0)$  is a non-singular matrix of the form

$$G_E = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & 1 \end{pmatrix}. \quad (2)$$

and  $G_D = G_E^{-1}$ . The matrices  $G_E$  and  $G_D$  are called the Gray encoding and the Gray decoding matrices, respectively. The implementation of the Gray encoder (decoder) requires  $n - 1$  two-input EXOR gates.

*Example 3.* Let  $n = 4$  and  $z = (1, 1, 0, 1)$  then

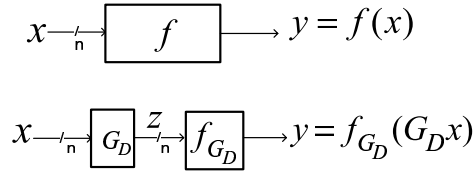
$$\begin{aligned} x_3 &= z_3 = 1 \\ x_2 &= z_3 \oplus z_2 = 0 \\ x_1 &= z_2 \oplus z_1 = 1 \\ x_0 &= z_1 \oplus z_0 = 1 \end{aligned}$$

or

$$x = (\tau_3, \tau_2, \tau_1, \tau_0)z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}.$$

### 3 Implementation of totally symmetric functions by Gray decoded inputs

In this paper we introduce an implementation of a symmetric function as a superposition of two functions: a Gray decoder defined by the matrix  $G_D$ , and the corresponding function  $f_{G_D}$  whereas  $f(x) = f_{G_D}(G_D x)$  (see Figure 3).



**Fig. 1.** Implementation of a Boolean function with a Gray decoding of the input variables

The main idea behind this approach is the following: A Boolean function maps elements of the vector space  $\{0, 1\}^n$  to  $\{0, 1\}$ . The vector space  $\{0, 1\}^n$  is spanned by  $n$  base vectors, usually the binary vectors  $\{\delta_i\}_{i=0}^{n-1}$  corresponding to the integer value  $2^i$  are used. The set of  $\delta_i$ 's is called the initial basis. This basis is used in definition of SOP expressions.

Any set of  $n$  linearly independent vectors forms a basis, and in particular, the columns  $\{\tau_i\}_{i=0}^{n-1}$  of the matrix  $G_E$ .

Since  $Ix = G_E z$ , the vector  $x$  can be interpreted as the coefficient vector that defines an element of  $\{0, 1\}^n$  using the initial basis, and  $z$  can be interpreted as the coefficient vector representing an element with the set of  $\tau$ 's. Thus, the matrices  $G_E$  and  $G_D$  define a linear transformation between the coefficient vectors.

*Example 4.* In Example 3, the element  $(1, 0, 1, 1) \in \{0, 1\}^4$  can be represented as a linear combination of the initial base vectors  $\delta_3 = (1, 0, 0, 0)$ ,  $\delta_2 = (0, 1, 0, 0)$ ,  $\delta_1 = (0, 0, 1, 0)$  and  $\delta_0 = (0, 0, 0, 1)$ , or as a linear combination of the columns of  $G_E$ . Namely,

$$(1, 0, 1, 1) = 1 \cdot \delta_3 + 0 \cdot \delta_2 + 1 \cdot \delta_1 + 1 \cdot \delta_0 = 1 \cdot \tau_3 + 1 \cdot \tau_2 + 0 \cdot \tau_1 + 1 \cdot \tau_0,$$

thus,  $x = (1011)$  and  $z = (1101)$ .

In theoretical considerations, complexity of circuit realization of a Boolean function is usually estimated without referring to a specific implementation technology. It is, therefore, often expressed in the number of two-input gates (AND/OR) that are required for the realization of the function considered. Formally, this criterion can be written in terms of a cost function [12, 27]

$$\mu(f) = |\{x|x, \tau \in \{0, 1\}^n, f(x) = f(x + \tau), \|\tau\| = 1\}|$$

where  $+$  stands for a bitwise EXOR of two binary vectors and  $\|\tau\|$  is the Hamming weight of a binary vector  $\tau$ . The autocorrelation function of  $f$ , is defined as  $R(\tau) = \sum_{x \in \{0, 1\}^n} f(x)f(x \oplus \tau)$ . For a given function  $f$ , the value of  $\mu$  can be related to the values of the autocorrelation function of  $f$ , at points corresponding to the base vectors,

$$\mu(f) = \sum_{i=0}^{n-1} R(\delta_i).$$

In the case of initial basis, these are points  $2^i$ , and linear transformation of variables performs the shift of these values.

There is a variety of minimization procedures that construct a linear transformation deterministically, see, for instance [14, 15] and [29] and references therein. It should be noticed that implementation of such procedures may be a space and time demanding task, and therefore, it is useful to take into considerations specific features of functions to be realized. In particular, we point out that for totally symmetric Boolean functions the linear transformation of variables derived from the Gray code almost always reduce the implementation cost. The same transformation often reduces the number of terms in Fixed polarity Reed-Muller expressions.

## 4 Motivation example

Consider the *3-out-of-5* function in Example 1. Let  $G_E$  and the  $G_D$  be the  $5 \times 5$  Gray encoding and decoding matrices. The columns of  $G_E$  are binary vectors of length 5 corresponding to the integer values 1, 3, 6, 12 and 24. Let  $z = G_D x$  be the Gray decoded inputs. Table 4 shows the K-map of  $f_{G_D}$ . The minimal SOP representation of  $f_{G_D}$  consists of 5 products,

$$\begin{aligned} f_{G_D}(z_4, z_3, z_2, z_1, z_0) = \\ z_3 z_2' z_0 + z_3 z_1' z_0 + z_4 z_2' z_0 + z_4' z_2 z_1' z_0 + z_4 z_3' z_1 z_0. \end{aligned}$$

The FPRM expansion of  $f_{G_D}$  with a polarity vector  $h = (11000)$  is

$$f_{G_D}(z_4, z_3, z_2, z_1, z_0) = z_0 \oplus z_2 z_1 z_0 \oplus z_3' z_2 z_0 \oplus z_4' z_3' z_0.$$

**Table 1.** K-map of a 3-out-of-5 function

$x_4 x_3 x_2$	000	001	011	010	110	111	101	100
$x_1 x_0$								
00						1		
01			1		1		1	
11		1		1				1
10			1		1		1	

**Table 2.** K-map of Gray coded 3-out-of-5 function

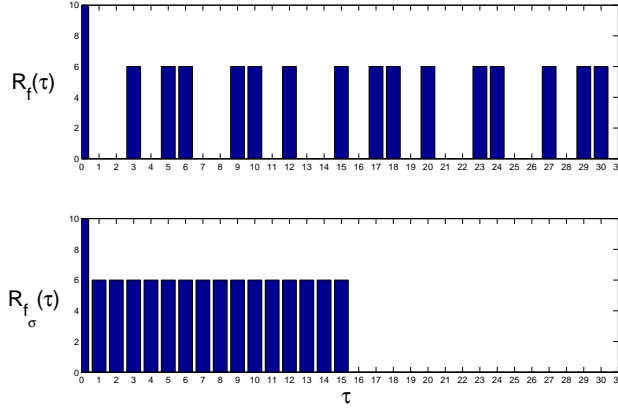
$z_4 z_3 z_2$	000	001	011	010	110	111	101	100
$z_1 z_0$								
00								
01		1	1	1	1	1		1
11				1	1		1	1
10								

The values of the autocorrelation function of the original 3-out-of-5 function are shown in Figure 2 (top figure). The values of  $R(\tau)$  at positions  $\tau = 1, 2, 4, 8$  and 16 corresponding to the initial base vectors are all zero, thus, the minimal SOP comprises 10 minterms. The autocorrelation values at positions  $\tau = 1, 3, 6, 12$  corresponding to the new base vectors ( $\tau_0, \tau_1, \tau_2$  and  $\tau_3$ ) are equal to 6.

Applying the Gray decoding on the inputs is equivalent to permuting the autocorrelation values so that high autocorrelation values are now placed at positions  $2^i$ . The autocorrelation function of  $f_{G_D}$  is shown at the bottom of Figure 2. The sum of the autocorrelation values of  $f_{G_D}$  at positions  $2^i, i = 0, \dots, 4$  is  $4 \cdot 6 + 0$ , therefore, the number of pairs in the first merging step of the Quine-McClusky minimization algorithm is now 12 which leads to a minimal SOP representation.

## 5 Analysis

Let  $f(x) = f(x_{n-1}, \dots, x_0) \sum_{i=0}^n a_i S_i(x)$ ,  $a_i \in \{0, 1\}$ , a totally symmetric Boolean function of  $n$  variables and a single output. The autocorrelation function of  $S_i(x)$



**Fig. 2.** Autocorrelation function values of the original *3-out-of-5* symmetric function  $f$  (top) and the values of the autocorrelation function corresponding to  $f_{G_D}$  with the Gray decoded inputs (bottom).

is [12]

$$\begin{aligned}
 R_{S_i}(\tau) &= \sum_{x \in \{0,1\}^n} S_i(x) S_i(x \oplus \tau) \\
 &= \begin{cases} \binom{n-|\tau|}{i-|\tau|/2} \binom{|\tau|}{|\tau|/2} & |\tau| \text{ is even} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

where  $\binom{a}{b} = 0$  for  $b < 0$ .

The cross correlation between  $S_i(x)$  and  $S_j(x)$  is

$$\begin{aligned}
 R_{S_i, S_j}(\tau) &= \sum_{x \in \{0,1\}^n} S_i(x) S_j(x \oplus \tau) \\
 &= \begin{cases} \binom{n-|\tau|}{i-w} \binom{|\tau|}{w} & i - j + |\tau| \text{ is even} \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

where  $w = (i - j + |\tau|)/2$ .

The autocorrelation function of  $f$  is

$$\begin{aligned}
 R_f(\tau) &= \sum_{x \in \{0,1\}^n} f(x) f(x \oplus \tau) \\
 &= \sum_{i=0}^n a_i R_{S_i}(\tau) + \sum_{\substack{i, j=0 \\ i \neq j}}^n a_i a_j R_{S_i, S_j}(\tau). \tag{3}
 \end{aligned}$$



Therefore, the autocorrelation values in positions corresponding the the initial set of base vectors  $\{\delta_i\}_{i=0}^{n-1}$  is

$$\begin{aligned} R_f(\delta_i) &= 2 \sum_{k=1}^{n-1} a_k a_{k+1} R_{S_k, S_{k+1}}(\tau) \\ &= 2 \sum_{k=1}^{n-1} a_k a_{k+1} \binom{n-1}{k}. \end{aligned} \quad (4)$$

On the other hand, the autocorrelation values at positions corresponding to the base vectors  $\tau_i = \delta_i + \delta_{i+1}$ ,  $i = 0, \dots, n-2$ , defined by the columns of the Gray encoding matrix  $G_E$ , are

$$R_f(\tau_i) = 2 \sum_{k=0}^n a_k \binom{n-2}{k-1} + 2 \sum_{k=1}^{n-2} a_k a_{k+2} \binom{n-2}{k} \quad (5)$$

The following Theorem states that the realization cost of  $f_{G_D}$  with the Gray decoded inputs is less or equal to the realization cost of  $f$  for any totally symmetric function.

**Theorem 1.** *Let  $f(x) = \sum_{i=1}^n a_i S_i(x)$ ,  $a_i \in \{0, 1\}$  a totally symmetric function, and let  $f_{G_D}$  the corresponding function with the Gray decoded inputs, i.e.  $f(x) = f_{G_D}(G_D x)$ . Then,*

$$\mu_f \leq \mu_{f_{G_D}}.$$

*Proof.* The proof is based the fact that  $R_{f_{G_D}}(\delta_i) = R_f(G_D^{-1} \delta_i) = R_f(\tau_i)$ . Let  $\Delta_i = R_f(\tau_i) - R_f(\delta_i)$ , clearly,  $\Delta_{n-1} = 0$  and for  $0 \leq i < n-1$ ,  $\Delta_i = 2 \sum_{k=0}^n d_k$  where

$$d_k = a_k \left( \binom{n-2}{k-1} - a_{k+1} \binom{n-1}{k} + a_{k+2} \binom{n-2}{k} \right). \quad (6)$$

We now show that  $\Delta_i \geq 0$  for all  $i$ . From 6, if  $a_k = 0$  than  $d_k = 0$ , otherwise, there are four possible cases:

1. If  $a_{k+1} = a_{k+2} = 0$  than  $d_k > 0$ .
2. If  $a_{k+1} = a_{k+2} = 1$  than  $d_k = 0$  since

$$\binom{a}{b} = \binom{a-1}{b} + \binom{a-1}{b-1}.$$

3. If  $a_{k+1} = 0$  and  $a_{k+2} = 1$  than  $d_k > 0$ .
4. If  $a_{k+1} = 1$  and  $a_{k+2} = 0$  than we may consider the sum  $d_k + d_{k+1}$  and get

$$\binom{n-2}{k-1} - \binom{n-1}{k} + \binom{n-2}{k} + a_{k+2} \binom{n-2}{k} \geq 0 \quad (7)$$

Therefore,  $R_{f_{G_D}}(\delta_i) = R_f(\tau_i) \geq R_f(\delta_i)$ . From [12], the cost function  $\mu_f$  of a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  equals to  $\mu_f = 2^n - 2R_f(0) + 2 \sum_{i=0}^{n-1} R_f(\delta_i)$ , and thus  $\mu_{f_{G_D}} \geq \mu_f$ .  $\square$

## 6 Implementation of partially symmetric functions by Gray decoded inputs

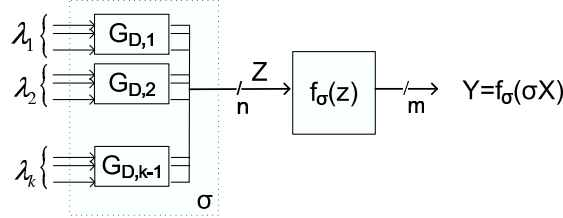
In this section we extend the use of the Gray decoder to the case of partially symmetric functions. We start with the definition of partially symmetric functions, and then we present a three-level implementation of partially symmetric functions which is based on a set of Gray decoders.

A function is said to be symmetric with respect to a set  $\lambda$ ,  $\lambda \subseteq \{x_{n-1} \dots x_0\}$ , if it is invariant under all the permutations of the variables in  $\lambda$ . The variables of the function can be partitioned into disjoint symmetry sets  $\lambda_1, \dots, \lambda_k$ . A function is called *partially symmetric* if it has at least one symmetry set  $\lambda_i$  of size  $|\lambda_i| > 1$ . Without loss of generality, we assume that the input variables are ordered, i.e. the first  $|\lambda_1|$  variables are elements of the first symmetry set  $\lambda_1$ , the next  $|\lambda_2|$  variables are elements of the second symmetry set  $\lambda_2$  etc.

A partially symmetric function  $f$  may be represented as a superposition of two functions: a linear function defined by a matrix  $\sigma$ , and a function  $f_\sigma$  whereas  $f(x) = f_\sigma(z)$  and  $z = \sigma x$ . The linear transformation matrix  $\sigma$  represents  $k$  Gray decoders that work on each symmetric set separately (see Figure 3). The matrix  $\sigma$  is defined through its inverse matrix  $T$  as follows,

$$T = \begin{pmatrix} G_{D,k} & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & G_{D,2} & 0 \\ 0 & \cdots & 0 & G_{D,1} \end{pmatrix} \quad (8)$$

where  $G_{D,i}$  is a Gray decoding matrix corresponding to  $|\lambda_i|$  variables. The overall implementation cost of the Gray decoders is  $(n - k)$  XOR gates.



**Fig. 3.** The original function (top) and its linear decomposition with a set of Gray decoders (bottom)

Recall that we are not interested in the optimal solution to the general minimization problem of partially symmetric functions. Rather, we suggest a specific linear transformation matrix based on the Gray decoder which is suitable for partially symmetric functions. The following example clarifies this point.

*Example 5.* Consider a single-output partially symmetric function  $f(x_4, x_3, x_2, x_1, x_0)$  that has two symmetry sets  $\lambda_1 = \{x_4, x_3, x_2\}$  and  $\lambda_2 = \{x_1, x_0\}$ . The K-map of the function is given in Table 3. The SOP representation of the function consists of 13 product terms and 62 literals.

Let  $T$  be the Gray encoding matrix as defined in Eq. 8 and  $\sigma = T^{-1}$  the Gray decoding matrix. The columns of  $T$  are binary vectors of length 5 corresponding to the integer values 1, 3, 4, 12 and 24, that is,

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Let  $z = \sigma x$  be the Gray decoded inputs. The K-map of  $f_\sigma(z)$  is given in Table 4. The minimal SOP representation of  $f_\sigma$  consists of 5 products and only 16 literals.

Note that the values of  $R(\tau)$  at positions  $\tau = 1, 2, 4, 8$  and 16 corresponding to the initial base vectors, are 0, 0, 2, 2 and 2 respectively. However, the autocorrelation values at positions  $\tau = 1, 3, 4, 12$  and 24 corresponding to the new base vectors are equal to 0, 12, 2, 10 and 10. Therefore, the cost function of the original function is  $\mu(f) = 6$  and it is smaller than  $\mu(f_\sigma) = 34$ . Nevertheless, the linear transformation corresponding to the Gray decoder is not optimal. It is possible to choose a different set of base vectors, for example, 3, 30, 5, 4 and 8, that is,

$$\hat{T} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

for which the autocorrelation values are 12, 12, 10, 2 and 2, respectively. Thus, the matrix  $\hat{\sigma} = \hat{T}^{-1}$  defines a function  $f_{\hat{\sigma}}$  having  $\mu(f_{\hat{\sigma}}) = 36$ . Consequently, the minimal SOP representation of  $f_{\hat{\sigma}}$  comprises 4 product terms and 13 literals. Clearly, this additional reduction in the number of literals (from 16 to 13) is negligible.

**Table 3.** K-map of the original function in Example 5

$x_4x_3x_2$ $x_1x_0$	000	001	011	010	110	111	101	100
00	1		1		1		1	
01		1		1				1
11			1		1	1	1	
10		1		1				1

The following lemma states that the Gray decoding of the inputs of a partially symmetric function cannot increase the implementation cost.

**Table 4.** K-map of Gray coded function in Example 5

$z_4z_3z_2$	000	001	011	010	110	111	101	100
$z_1z_0$								
00	1			1	1			1
01		1	1			1		
11		1	1			1		
10				1	1		1	1

**Lemma 1.** *Let  $f$  be a partially symmetric function that has  $k \geq 1$  symmetric sets  $\lambda_1, \dots, \lambda_k$ . Let  $\sigma$  be the linear transformation matrix as defined in Eq. 8, and  $f_\sigma$  be the corresponding linearized function,  $f(x) = f_\sigma(\sigma x)$ . Then  $\mu(f_\sigma) \geq \mu(f)$ .*

The proof of the lemma is similar to the proof of Theorem 1.

## 7 Experimental results

In this section, we compare the implementation cost of the original and Gray-coded functions in terms of:

- The number of Look-Up-Tables (LUTs) required to implement the function by *SPARTAN3 xcs200ft256* as computed by LeonardoSpectrum.
- The number of literals ( $L$ ) in its minimal SOP representation as produced by ESPRESSO.
- The number of nonzero terms in the optimal Fixed-Polarity Reed-Muller (FPRM) expansion.

Tables 5 and 6 show the number LUTs for several totally symmetric functions of 8 and 12 input variables, the number of literals in the minimal SOP expression and the number of non-zero FPRM terms as computed with and without the Gray decoding. The improvement in those parameters is given in percentage. The symmetric functions  $f = \sum_i a_i S_i(x)$  are specified by a set  $I$ ,  $I = \{i | a_i \neq 0\}$ , of working parameters,  $I$  is written in the left column of Tables 5 and 6. The simulation results show an average reduction of 70% in the number of LUTs, an average reduction of 88% in the number of literals, and 68% in the number of non-zero FPRM coefficients.

Table 7 shows how the Gray decoding reduces the implementation cost of several totally symmetric LGSynth93 benchmark functions. Given a polarity vector, the number of non-zero FPRM terms of a  $k$ -output function is defined as the size of the union of the non-zero terms in the FPRM expansion of each one of the  $k$  single-output functions. For example, the original benchmark function *rd84* has four outputs, the number of non-zero FPRM terms of each is 28, 8, 1 and 70 and the size of the union of these terms is 107. The number of non-zero terms of the corresponding Gray coded single-output functions is 14, 4, 1 and 38 and the size of their union is 39.

Simulation results that demonstrate the efficiency of Gray decoding on a number of partially symmetric functions are shown in Table 8. The simulation

results show an average reduction of 86% in the number of literals and an average reduction of 60% in the number of non-zero FPRM coefficients. The first column in the table is the benchmark name, the second column shows the number of inputs, the third column specifies which single-output function is simulated, and the fourth column shows the number of symmetry sets in that function.

**Table 5.** Totally symmetric functions of 8 inputs

<i>I</i>	<i>LUT</i>		%	<i>L</i>		%	<i>FPRM</i>		%
	<i>orig</i>	<i>Gray</i>		<i>orig</i>	<i>Gray</i>		<i>orig</i>	<i>Gray</i>	
3	12	7	41.7	448	92	79.5	64	24	62.5
4	13	9	30.8	560	106	81.1	107	15	86.0
3,4	18	13	27.8	490	185	62.2	96	31	67.7
3,5	15	8	46.7	896	45	95.0	104	17	83.6
3,4,5	18	15	16.7	336	123	63.4	162	49	69.7
2,3,5,7	19	10	47.4	904	74	91.8	36	40	-11.1
0,2,3,5,8	18	11	38.9	856	109	87.3	107	25	76.6

**Table 6.** Totally symmetric functions of 12 inputs

<i>I</i>	<i>LUT</i>		%	<i>L</i>		%	<i>FPRM</i>		%
	<i>orig</i>	<i>Gray</i>		<i>orig</i>	<i>Gray</i>		<i>orig</i>	<i>Gray</i>	
3	65	26	60.0	2640	470	82.2	232	200	13.8
4	32	41	-28.1	5940	800	86.5	794	166	79.1
3,4	143	71	50.3	5445	1225	77.5	562	306	45.6
3,5	37	57	-54.1	12144	584	95.2	1024	136	86.7
3,4,5	204	118	40.2	7920	1170	85.2	1354	356	73.7
0,2,3,5,8	217	101	53.5	17876	1582	91.1	738	328	55.6

**Table 7.** Totally symmetric benchmark functions

	in	out	<i>LUT</i>		<i>L</i>		<i>FPRM</i>	
			<i>orig</i>	<i>Gray</i>	<i>orig</i>	<i>Gray</i>	<i>orig</i>	<i>Gray</i>
rd53	5	3	6	4	140	35	20	12
rd73	7	3	24	8	756	141	63	24
rd84	8	4	51	13	1774	329	107	39
9sym	9	1	36	36	504	135	173	33
total			117	61	3174	640	363	108

**Table 8.** Partially symmetric benchmark functions

	in	out	sets	$L$	$L$	$FPRM$	$FPRM$
		func.		<i>orig</i>	<i>Gray</i>	<i>orig</i>	<i>Gray</i>
z4	7	1	2	12	1	3	1
z4	7	2	3	48	12	5	4
z4	7	3	3	136	22	9	6
radd	8	1	4	64	14	15	7
radd	8	2	4	184	27	9	6
radd	8	3	4	68	13	5	4
adr4	8	2	3	20	7	3	2
adr4	8	3	4	68	13	5	4
add6	12	1	6	63	6	63	11
add6	12	3	6	456	40	17	8
total				1119	155	134	53

## 8 Conclusions

The problem of linearization of logic functions may be considered as a determining a linear transform for variables in a given function, which produces a representation of the function appropriate for particular applications. However, it is not always necessary to determine the best possible linear transformation for a class of functions. For many practical applications it is sufficient to find a suitable transform producing acceptable solutions.

In this paper we consider the class of symmetric functions and point out a suitable linear transformation of variables resulting in considerably reduced number of product terms in AND-OR and Reed-Muller expressions.

We propose a method to represent a symmetric logic function as a superposition of a linear portion that realize the Gray decoding of input vectors and a non-linear portion. Being a particular case of the linear transformation, the described Gray decoding transform enables to achieve very compact implementations of the initial symmetric function.

We have shown that the use of the Gray transform improves the complexity of the initial function implementation in terms of a specific cost function. Experimental results show that for majority of benchmarks the proposed method improves also a LUT based implementation of the function. The suggested approach can be extended to partially symmetric functions. In addition, we have shown that it provides an average reduction of about 86% in the number of literals and an average reduction of about 60% in the number of non-zero FPRM coefficients.

## References

1. J.T. Astola and R.S. Stankovic, *Fundamentals of Switching Theory and Logic Design: A Hands on Approach*, Springer-Verlag New York, 2006.

2. M. Chrzanowska-Jeske and Z. Wang, "Mapping of symmetric and partially-symmetric functions to the CA-type FPGAs" *proc. of the 38th Midwest Symposium on Circuits and Systems*, vol. 1, pp. 290-293, Aug 1995.
3. D. L. Dietmeyer, "Generating minimal covers of symmetric functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 5, pp. 710-713, May 1993.
4. R. Drechsler, B. Becker, "Sympathy: fast exact minimization of fixed polarity Reed-Muller expressions for symmetric functions," *Proc. of the European Design and Test Conference*, pp. 91 - 97, March 1995.
5. R. Drechsler and B. Becker, "EXOR transforms of inputs to design efficient two-level AND-EXOR adders," *IEE Electronic Letters*, vol. 36, no. 3, pp. 201-202, Feb. 2000.
6. F. Gray, "Pulse code communication," March 17, 1953 (filed Nov. 1947). U.S. Patent 2,632,058.
7. W. Günther, R. Drechsler, "BDD minimization by linear transforms", *Advanced Computer Systems*, pp. 525-532, 1998.
8. S.L. Hurst, D.M. Miller, J.C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, Bristol, 1985.
9. J. Jain, D. Moundanos, J. Bitner, J.A. Abraham, D.S. Fussell and D.E. Ross, "Efficient variable ordering and partial representation algorithm," *Proc. of the 8th International Conference on VLSI Design*, pp. 81-86, Jan. 1995.
10. J. Jakob, P.S. Sivakumar, V.D. Agarwal, "Adder and comparator synthesis with exclusive-OR transform of inputs", *Proc. 1st Int. Conf. VLSI Design*, pp. 514-515, 1997.
11. S. Kannurao and B. J. Falkowski, "Identification of complement single variable symmetry in Boolean functions through Walsh transform," *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, 2002.
12. M.G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices*, John Wiley, 1976.
13. M.G. Karpovsky, E.S. Moskalev, "Utilization of autocorrelation characteristics for the realization of systems of logical functions," *Avtomatika i Telemekhanika*, No. 2, 1970, 83-90, English translation *Automatic and Remote Control*, Vol. 31, pp. 342-350, 1970.
14. M.G. Karpovsky, R.S. Stankovic and J.T. Astola, "Reduction of sizes of decision diagrams by autocorrelation functions," *IEEE Trans. on Computers*, vol. 52, no. 5, pp. 592-606, May 2003.
15. O. Keren, I. Levin and R.S. Stankovic, "Linearization of Functions Represented as a Set of Disjoint Cubes at the Autocorrelation Domain," *Proc. of the 7th International Workshop on Boolean Problems*, pp. 137-144, Sept. 2006.
16. B. G. Kim, D.L. Dietmeyer, "Multilevel logic synthesis of symmetric switching functions," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.10, No. 40, pp. 436-446, Apr. 1991.
17. E.J., Jr. McCluskey, "Detection of group invariance or total symmetry of a Boolean function," *Bell Systems Tech. Journal*, vol. 35, no. 6, pp. 1445-1453, 1956.
18. Ch. Meinel, F. Somenzi, T. Tehobald, "Linear sifting of decision diagrams and its application in synthesis," *IEEE Trans. CAD*, Vol. 19, No. 5, 2000, 521-533.
19. D. Moller, J. Mohnke, and M. Weber, "Detection of symmetry of Boolean functions represented by ROBDDs," *Proc. of the International Conference on Computer-Aided Design (ICCAD)*, 1993, pp. 680-684.
20. C. Moraga, "Introducing disjoint spectral translation in spectral multiple-valued logic design", *IEE Electronics Letters*, 1978, Vol. 14, No. 8, pp. 248-243, 1978.

21. C. Moraga, "On some applications of the Chrestenson functions in logic design and data processing", *Mathematic and Computers in Simulation*, Vol. 27, pp. 431-439, 1985.
22. E.I. Nechiporuk, "On the synthesis of networks using linear transformations of variables", *Dokl. AN SSSR*. Vol. 123, No. 4, pp. 610-612, Dec. 1958.
23. S. Panda, F. Somenzi, and B. Plessier, "Symmetry detection and dynamic variable ordering of decision diagrams," *Proc. of the International Conference on Computer-Aided Design (ICCAD)*, 1994.
24. T. Sasao, *Logic Synthesis and Optimization* , Springer, 1993.
25. T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, Feb. 1999
26. T. Sasao, "A new expansion of symmetric functions and their application to non-disjoint functional decompositions for LUT type FPGAs", *IEEE Int. Workshop on Logic Synthesis, IWLS-2000*, May 2000.
27. C. E. Shannon, "The Synthesis of Two-Terminal Switching Circuits," *Bell System Technical Journal*, Vol. 28, pp. 59-98, Jan. 1949.
28. R.S. Stankovic, J.T. Astola, "Some remarks on linear transform of variables in adders," *Proc. 5th Int. Workshop on Applications of Reed-Muller Expression in Circuit design*, Starkville, Mississippi, USA, Aug. 10-11, pp. 294-302, 2001.
29. D. Varma and E.A. Trachtenberg, "Design automation tools for efficient implementation of logic functions by decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 901-916, Aug. 1989.
30. K.H. Wang and J.H. Chen, "Symmetry Detection for Incompletely Specified Functions," *Proc. of the 41st Conference on Design Automation Conference, (DAC'04)*, pp. 434-437, 2004.
31. G. Wolfovitz "The complexity of depth-3 circuits computing symmetric Boolean functions," *Information Processing Letters*, vol. 100, No. 2, pp. 41 - 46, Oct. 2006.
32. S.N. Yanushkevich, J.T. Butler, G.W. Dueck, V.P. Shmerko, "Experiments on FPRM expressions for partially symmetric logic functions", *Proc. 30th Int. Symp. on Multiple-Valued Logic*, Portland, Oregon USA, 141-146, May 2000.
33. J. S. Zhang, A. Mishchenko, R. Brayton, M. Chrzanowska-Jeske, "Symmetry detection for large Boolean functions using circuit representation, simulation, and satisfiability", *Proceedings of the 43rd annual conference on Design automation* , pp. 510 - 515, 2006.