

# Former and Recent Work in Classification of Switching Functions

Radomir S. Stanković  
Dept. of Computer Science  
Faculty of Electronics  
18 000 Niš  
Serbia

Jaakko Astola  
Department of Signal Processing  
Tampere University of Technology  
FI-33101 Tampere  
Finland

Bernd Steinbach  
Institute of Computer Science  
Freiberg University of Mining and Technology  
D-09596 Freiberg  
Germany

## Abstract

This paper is a tutorial review of some former and recent work in classification of switching functions. Considered are different approaches to the classification and discussed features of resulting classification methods.

## 1 Introduction

Classification of switching functions is among the oldest and most important problems in switching theory, since is closely related to realization of switching functions by exploiting a small number of different circuit modules.

In switching theory, the term class of switching functions is often used in two slightly different although closely related meanings. First, study of functions that share some common property and in this way belong to the same class of functions is always useful since, in general, functions with some peculiar properties are easier to be analyzed and realized than other Boolean functions. Examples are classes of linear, self-dual, monotone, threshold functions, etc. Exploiting such properties in representations and realizations usually provides advantages compared to the representation and realization of arbitrary switching functions without particular properties. For instance, threshold logic synthesis is a good example supporting this statement, see for instance [1], [2], [28].

In another interpretation, the term class of switching functions is used to denote a subset of the set of all  $2^{2^n}$  switching functions of a given number of variables  $n$  that can be derived from each other by applying some precisely defined and possibly simple classification rules. In this paper, we use the term class of switching functions in the later meaning. The continuous interest in this approach to the classification is driven by realization problems and nowadays applications of these basic ideas and motivation for their further developments can be explained by the following considerations.

Computer aided design of digital systems incorporates logic synthesis as an essential part. There are basically two major steps,

1. First, for a given switching function a multi-level network is determined, and then optimized with respect to various criteria as, for instance, the number of gates and testability. This is a technology independent step in the design procedure.
2. The produced logic network is mapped to the targeted technology usually dealing with a library of available physical cells or logic blocks. In a library, usually there are several physical cells capable of realizing the same function but with different area/speed characteristics. Multi-criterion optimization is performed again with respect to the area, speed, testability, dissipated power, and power consumption, etc.

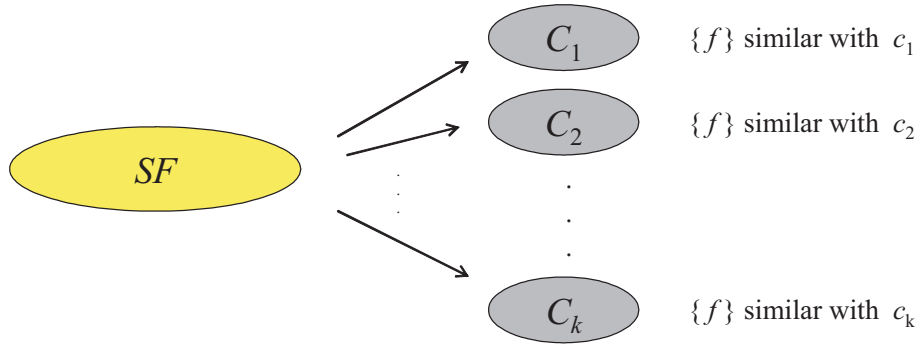


Figure 1: Classification of switching functions.

Technology independent network optimization, technology mapping, and cell library selection exploit in a direct or indirect way functional properties of functions to be realized. Efficient identification of functionally equivalent logic is essential in these areas.

The goal of the classification is to find functions similar to each other in some sense. The notion of similarity is defined in different ways depending on the classification rules selected.

For instance, a classification rule can be taking logic complement of function values, and in this case, two functions are similar if they are logic complement of each other. It follows that a function  $f$  and the logic complement of it  $\bar{f}$ , can be realized by the same network by adding an inverter at the output of the network that realizes  $f$ .

The principle expressed in this example can be generalized. Two functions which belong to the same class can be realized by similar networks. These networks are derived by a simple modification of the basic module realizing the function selected to be the representative of the considered class. The required modifications correspond to classification rules that are applied to demonstrate the classification equivalence of the functions, however, the rules are applied in the opposite order.

Different classifications have been defined by selecting various particular classification rules. In this paper, we will briefly present the *NPN*-classification and *LP*-classification as two examples of classification procedures based on different transformations of variables and function values. Then, we will consider classification in the spectral domain in terms of the Reed-Muller coefficients, Walsh coefficients, and autocorrelation functions. Finally, we will provide some basic concepts about a particular approach to the classification in terms of the decision diagrams to represent switching functions.

## 2 Classification of Switching Functions

Fig. 1 explains that classification task consists of the partition of the set  $SF$  of all switching functions of a given number of variables  $n$  into classes  $C_i$  of functions mutually similar with respect to some appropriately formulated classification criteria. Each class is represented by a *representative function*  $c_i$ , the *representative* of the class  $C_i$ . Functions that belong to the same class can be reduced to each other by applying the operations performed in the classification. These operations are usually denoted as the *classification rules*. In particular, a function  $f \in C_i$  can be reduced to the representative function  $c_i$  by the application of the classification rules. It follows that  $f$  can be realized by the same network as  $c_i$  modified as determined by the classification rules applied in order reverse to that used in the classification. It should be noticed that classes are usually but not necessarily disjunctive in all classification approaches.

There are several applications of the classification, and we point out two of them

1. Realization by prototypes, which assumes design of similar circuits for functions within the same class,
2. Standardization of methods for testing logic networks.

Traditionally, classification of switching functions has as one goal the reduction of the number of different logic networks to realize all the functions of a given number of variables. Further development of this idea leads to the *universal logic modules (ULMs)* defined as logic networks capable of realizing any of the  $2^{2^n}$  logic functions of  $n$  variables. It is assumed that constants logic 0 and 1 and switching

variables in positive and negative polarity are available at the inputs of ULMs. A ULM must have some control inputs to select which particular function the ULM realizes. The increased number of inputs, compared to the number of variables, is the price for the universality of the module. For example, a ULM for functions of three variables, has five inputs, two of which are the control inputs.

Nowadays, the classification is closely related to problems in technology mapping as for instance cell library binding [8]. In this case, a given logic circuit is mapped into a bound logic circuit consisting of modules taken from a specified library of circuits. In many such techniques, some form of classification is used to efficiently match a given circuit into modules specified in the library.

In FPGA synthesis, a function of  $n$  variables is often realized by a logic block of  $k > n$  inputs. A way to perform this is to set some of the inputs to constants 0 or 1, or connect some inputs together. It is said that the function is degenerated to the given target function, however, this can not be done for every function which leads to the problem of specifying the so-called *module functions* for logic blocks in FPGAs. The answers can be found by looking into representative functions for different classes of switching functions.

### 3 NPN-classification

Probably the most widely used is the *NPN-classification* due to the simplicity of the classification rules, which means modification in the networks for representative function required to realize any function in the class.

In *NPN*-classification, the classification rules are

1. Negation of input variables ( $N$ )  $\bar{x}_i \leftrightarrow x_i$ ,
2. Permutation of input variables ( $P$ )  $x_i \leftrightarrow x_j$ ,
3. Negation of the output ( $N$ )  $f \rightarrow \bar{f}$ .

Depending on the allowed classification operations, it may be distinguished the classification

1.  $N$  (rule 1),
2.  $P$  (rule 2),
3.  $NP$  (rules 1 and 2),
4.  $NPN$  (rules 1, 2 and 3),

with  $NP$  and  $NPN$ -classification the most often used in practice. Classifications with larger number of classification rules produce fewer number of classes, and due to that are considered as stronger classifications, since each representative function covers a subset of functions.

We say that two functions  $f_1$  and  $f_2$  of the same number of variables  $n$  belong to the same class, or are equivalent, if can be reduced each to other by the classification rules allowed in the classification considered.

For example, if  $n = 3$ , there are 12, 6, and 4 representative functions in  $P$ ,  $NP$ , and  $NPN$  classes, respectively.

Table 1 compares the number of representative functions in different classes of functions. It shows the number of function of  $n$  variables ( $\#f$ ), functions that essentially depend on all  $n$  variables ( $\#f(n)$ ), and representative functions in  $C_P$ ,  $C_{NP}$  and  $C_{NPN}$  classes.

Table 2 shows the asymptotic number of functions per classes when  $n$  is sufficiently large.

The  $NP$ -classification has been considered by S.W. Golomb already in 1959, [12], and latter explored by many authors. Discussions of this topics can be found in [3], [13], [14], [24], [28].

$NPN$ -classification partitions the set of all switching functions of a given number  $n$  of variables in disjoint classes, since each function is covered by a single representative function which implies that a function cannot belong to several classes. In particular, it can be shown that  $NPN$ -equivalence is an equivalence relation, see for instance [3], [24], [28].

Table 1: Number of functions of  $n$  variables ( $\#f$ ), functions dependent on all  $n$  variables ( $\#f(n)$ ), and representative functions in  $C_P$ ,  $C_{NP}$  and  $C_{NPN}$  classes.

	n					
	1	2	3	4	5	6
$\#f$	4	16	256	65536	$4.3 \times 10^9$	$1.8 \times 10^{19}$
$\#f(n)$	2	10	128	64594	$4.3 \times 10^9$	$1.8 \times 10^{19}$
$ C_P $	4	12	80	3984	$3.7 \times 10^7$	-
$ C_{NP} $	3	6	22	402	1228158	$4.0 \times 10^{14}$
$ C_{NPN} $	2	4	14	222	616126	$2.0 \times 10^{14}$

Table 2: Asymptotic number of functions per classes for  $n$  sufficiently large.

$P$	$NP$	$NPN$
$\frac{2^{2^n}}{n!}$	$\frac{2^{2^n}}{2^n n!}$	$\frac{2^{2^n}}{2^{n-1} n!}$

### 3.1 $NPN$ -classification in terms of Reed-Muller expressions

The Reed-Muller expressions are a class of functional expressions to represent switching functions [3], [28]. They can be viewed as the decomposition of a given function  $f(x_1, \dots, x_n)$  into a linear combination of a set of basis functions defined by the elementary products of variables (the set of all possible products of switching variables  $x_i$ ,  $i = 1, \dots, n$ ). The literals for the variables can be selected as either positive or negative, but not both at the same time in the same expression for  $f$ . This leads to the Fixed-polarity Reed-Muller expressions, with the polarities for variables usually conveniently expressed by polarity vectors defined as  $n$ -bit vectors whose binary entries specify selection of the positive and the negative literals for each of  $n$  variables. In this theory, the complexity of a Reed-Muller expressions is often estimated in terms of two *weight vectors*.

The weight  $w_p$  of a Reed-Muller expression is defined as the number of product terms in the given expression for a specified polarity vector. Similarly, the weight  $w_l$  is the number of literals required in a Reed-Muller expression of a given polarity. The weight vector  $W_p$  ( $W_l$ ) is a vector of all  $2^n$  weights  $w_p$  ( $w_l$ ) arranged in ascending order of their magnitudes.

In [4], see also [5], it is shown that vectors  $W_p$  and  $W_l$  are invariant to  $NPN$ -classification operations.

## 4 Classification in terms of Reed-Muller Expressions

In [36], the switching functions are classified in 8 classes defined as follows. Denote by  $|f|$  the number of non-zero values in the truth-vector of a function  $f$ .

1. A switching function  $f$  is called *neutral* if  $|f| = |\bar{f}|$ . Neutral switching functions form the class  $NEU$ .
2. A switching function  $f$  is called *odd (even)* if  $|f|$  an odd (even) integer. Such functions form classes  $ODD$  and  $EVEN$ .
3. A switching function  $f$  is called *degenerated* if there exists a variable  $x_i$  such that  $f(x_i) = f(\bar{x}_i)$ , i.e.,  $x_i$  is not an essential variable for  $f$ . Such functions form the class  $DEG$ .
4. A switching function  $f$  is a *self-dual function* if  $f(x_1, \dots, x_n) = \bar{f}(\bar{x}_1, \dots, \bar{x}_n)$ . Self-dual functions form the  $SD$ -class.
5. A variable  $x_i$  in a switching function  $f$  is called *linear* if  $\bar{f}(x_i) = f(\bar{x}_i)$ . The set of functions that contain at least one linear variable form the  $LV$ -class.
6. A function  $f$  is *self-complementary* if both  $f$  and its complement  $\bar{f}$  belong to the same  $NP$ -class. All self-complementary functions for a given number of variables  $n$  form the  $SC$ -class.

Table 3: Distribution of functions for  $n = 4$  into 13 classes.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13
# of $f$	32768	19208	5856	5768	786	176	32	32	108	16	96	0	690

7. The *NSC* class is defined as the set difference of the *NEU*-class and the *SC*-class.

Splitting switching functions in exactly these classes is motivated by various problems in technology mapping. For instance, in both mask and field programmable technologies, the minimization of inverters is a problem [15]. Library cells with linear variables, or cells that are capable of realizing self-complementary functions are useful in reducing the number of inverters by allowing their shift between the inputs and outputs of the cell. To perform such an operation, the required property should be identified in advance.

Some of these classes can be specified using derivative operations of the Boolean Differential Calculus [19].

Let  $f(\mathbf{x}) = f(x_1, \dots, x_i, \dots, x_n)$  be a logic function of  $n$  variables, then

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = f(x_1, \dots, x_i, \dots, x_n) \oplus f(x_1, \dots, \bar{x}_i, \dots, x_n)$$

is the (simple) derivative of the logic function  $f(\mathbf{x})$  with regard to the variable  $x_i$ .

Using this definition of the simple derivative two of the above classes can be specified. A function  $f$  is in the class *DEG* with regard to  $x_i$  if

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0,$$

which means, the function is independent of  $x_i$ . A function  $f$  is in the class *LV* with regard to  $x_i$  if

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 1,$$

which means, the function is linear with regard of  $x_i$ .

The derivative can be calculated not only with regard to a single variable but with regard to several variables, too.

Let  $\mathbf{x}_0 = (x_1, x_2, \dots, x_k)$ ,  $\mathbf{x}_1 = (x_{k+1}, x_{k+2}, \dots, x_n)$  be two disjoint sets of Boolean variables, and  $f(\mathbf{x}_0, \mathbf{x}_1) = f(x_1, x_2, \dots, x_n) = f(\mathbf{x})$  a logic function of  $n$  variables, then

$$\frac{\partial f(\mathbf{x}_0, \mathbf{x}_1)}{\partial \mathbf{x}_0} = f(\mathbf{x}_0, \mathbf{x}_1) \oplus f(\bar{\mathbf{x}}_0, \mathbf{x}_1)$$

is the vectorial derivative of the logic function  $f(\mathbf{x}_0, \mathbf{x}_1)$  with regard to the variables of  $\mathbf{x}_0$ .

Using this definition of the vectorial derivative with regard to all variables it can be detected whether a function belongs to the class *SD*. A function  $f$  is in the class *SD* if

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 1.$$

In [36], the above mentioned classes, *ODD*, *EVEN*, *NEU*, *DEG*, *NSC*, *SC*, *SD*, and *LV* are determined and characterized in terms of *Fixed-polarity Reed-Muller expressions*. The representative functions of classes have been expressed by the Reed-Muller expressions in fixed polarity, i.e., with all the variables represented by either positive or negative literals. The results in [36], have been corrected in [7] by showing that  $(SD \cap DEG) \setminus LV$ ,  $(LV \cap DEG) \setminus SD$  and  $DEG \cap SD \cap LV$  are non-empty sets. Due to these considerations, instead into 8 classes as proposed in [36], switching functions should be classified into 13 classes as described in Fig. 2, where the above mentioned non-empty sets are denoted by 8, 9, and 10, respectively. The distribution of functions per these classes for  $n = 4$  is as in Table 3 [7]. Interestingly, for  $n = 4$ , the class 12 is empty, but it is not for  $n > 4$  as shown by an example [7].

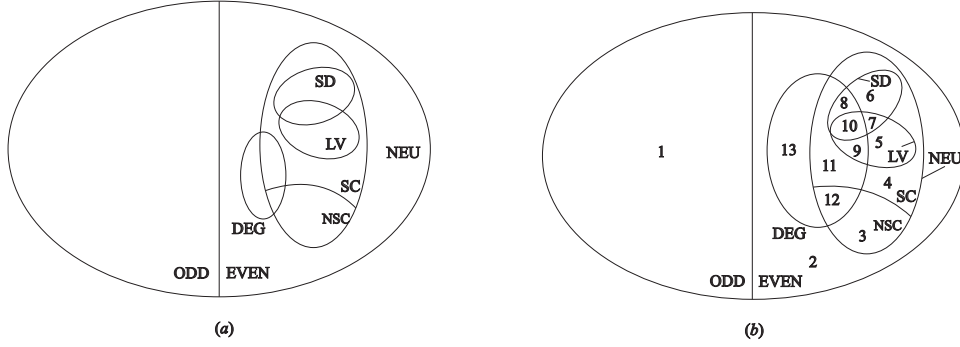


Figure 2: Classification in terms of Reed-Muller expressions (a) as in [36], (b) as in [7].

## 5 *SD*-classification

A classification stronger than *NPN*-classification is defined in terms of *self-dual functions* and is, therefore, called the *Self-dual (SD) classification* of switching functions. It is based upon the following representation of switching functions.

Consider a function  $f(x_1, \dots, x_n)$  and its dual function  $f^d(x_1, \dots, x_n)$  defined by

$$f^d(x_1, \dots, x_n) = \bar{f}(\bar{x}_1, \dots, \bar{x}_n).$$

The function

$$f^{sd}(x_1, \dots, x_n, x_{n+1}) = x_{n+1}f(x_1, \dots, x_n) + \bar{x}_{n+1}f^d(x_1, \dots, x_n),$$

is called its self-dualization.

Switching functions that become identical by self-dualization, nonself-dualization, negation of variables or the function, or permutation of variables are called *SD*-equivalent. It is an equivalence relation where the equivalence classes contain nonself-dual functions of  $n$  variables and self-dual function of  $n + 1$  variables.

This classification splits the set of all  $2^{2^n}$  switching functions into non-disjoint classes, which means that a given function can be realized by more than one representative functions. For example, if  $n = 3$ , there are 7 classes realizing 4, 31, 8, 128, 64, 96, and 128 functions respectively. It should be noticed that the sum of the number of functions per classes is larger than  $2^{2^3} = 256$ , which is the total number of functions for  $n = 3$ .

For more information, see for example [3], [28].

## 6 *LP*-classification

*NPN* and *SD*-classifications are intended for AND-OR synthesis, which means the representation of functions by AND-OR expressions. From 1990, there is apparent an increasing interest in AND-EXOR synthesis, mainly after publication of [29], due to the feature that AND-EXOR expressions require on the average fewer product, accompanied by the technology advent which provided EXOR circuits with the same propagation delay and at about the same price as classical OR circuits with the same number of inputs, as it was documented in 1989 by the example of EXOR circuits with four inputs.

**Example 1** Table 4 shows the number of functions of four variables that can be realized with  $t$  products by AND-OR and AND-EXOR expressions, represented by SOPs and ESOPs, respectively. Notice that AND-OR expressions require on the average 4.13 compared with 3.66 products in AND-EXOR expressions.

This consideration was a motive to introduce *LP*-classification adapted by the allowed classification rules to the AND-EXOR synthesis [18], [20]. In this classification, unlike *NPN* and *SD*-classification, transformations over constants, besides over variables, are also allowed.

*LP*-classification has been introduced by the following considerations.

Consider the following transformations that change a function  $f$  to (possibly) another function  $g$

Table 4: Number of functions realizable by  $t$  products.

$t$	AND-OR	AND-EXOR
0	1	1
1	81	81
2	1804	2268
3	13472	21744
4	28904	37530
5	17032	3888
6	3704	24
7	512	0
8	26	0
av.	4.13	3.66

1. For  $i \in \{1, \dots, n\}$  and  $f(x_1, \dots, x_n)$  written in the Shannon expansion with respect to the variable  $x_i$

$$f = \bar{x}_i f_0 \oplus x_i f_1.$$

the function  $g$  is in the form

$$g = \begin{bmatrix} g_0 \\ g_1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \end{bmatrix},$$

i.e.,

$$g = \bar{x}_i g_0 \oplus x_i g_1 = \bar{x}_i (a f_0 \oplus b f_1) \oplus x_i (c f_0 \oplus d f_1),$$

where  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$  is a nonsingular matrix over  $GF(2)$ .

2.  $g$  is obtained from  $f$  by permuting variables.

Functions  $f$  and  $g$  are called *LP-equivalent* if  $g$  is obtained from  $f$  by a sequence of (operations) transformations (1) and (2) above [20].

It is clear that the *LP-equivalence* is an equivalence relation.

**Example 2** Consider the transformation by the matrix  $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ . Then,

$$g = \begin{bmatrix} g_0 \\ g_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_0 \oplus f_1 \end{bmatrix},$$

and equivalently

$$g = \bar{x}_i g_0 \oplus x_i g_1 = \bar{x}_i f_0 \oplus x_i (f_0 \oplus f_1) = (\bar{x}_i \oplus x_i) f_0 \oplus x_i f_1 = 1 \cdot f_0 \oplus x_i f_1.$$

Thus,  $g$  is obtained by the substitution  $\bar{x}_i \rightarrow 1$  in the expression of  $f$ .

Similarly, the matrix  $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$  corresponds to the substitution  $x_i \rightarrow 1$ .

Usually, the six transformations corresponding to the six different nonsingular matrices over  $GF(2)$  are expressed in this substitution notation as shown in Table 5.

It is worth noticing that if  $f$  is represented by an ESOP with  $|f|$  products, then  $|g| = |f|$  [20].

For functions of  $n = 3$  variables, there are 6 *LP-representative* functions

$$\begin{array}{ll} 0, & \bar{x} \cdot \bar{y} \cdot \bar{z}, \\ \bar{x} \cdot y \oplus \bar{x} \cdot z, & \bar{x} \oplus \bar{y} \cdot z \oplus \bar{x} \cdot y \cdot z, \\ x \cdot \bar{y} \cdot \bar{z} \oplus \bar{x} \cdot y \cdot z, & \bar{x} \oplus y \cdot \bar{z} \oplus x \cdot \bar{y} \cdot z. \end{array}$$

Table 5:  $LP$ -classification rules.

1.	$LP_0(f) = \bar{x}_i f_0 \oplus x_i f_1$	Identical mapping	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
2.	$LP_1(f) = \bar{x}_i f_0 \oplus 1 \cdot f_1$	$x_i \leftrightarrow 1$	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
3.	$LP_2(f) = 1 \cdot f_0 \oplus x_i f_1$	$x_i \leftrightarrow 1$	$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$
4.	$LP_3(f) = x_i f_0 \oplus \bar{x}_i f_1$	$x_i \leftrightarrow \bar{x}_i$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
5.	$LP_4(f) = x_i f_0 \oplus 1 \cdot f_0$	$\bar{x}_i \rightarrow 1$ , and $\bar{x}_i \rightarrow x_i$	$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
6.	$LP_6(f) = 1 \cdot f_0 \oplus \bar{x}_i f_1$	$\bar{x}_1 \rightarrow 1$ , and $x_i \rightarrow \bar{x}_i$	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

Table 6:  $LP$ -representative functions for  $n \leq 4$ .

$n$	2	3	4		
0	00	0000	016a	0678	
1	01	0001	0180	06b0	
6	06	0006	0182	06b1	
	16	0016	0186	1668	
	18	0018	0196	1669	
	6b	0066	0660	1681	
			0116	0661	1683
			0118	0662	168b
			012c	066b	18ef
			0168	0672	6bbd

Table 7: Number of  $LP$ -classes and its medium sizes.

$n$	1	2	3	4	5	6
$ C_{LP} $	2	3	6	30	6936	$> 5.5 \times 10^{11}$
$ f $	4	16	256	65536	$4.3 \times 10^9$	$1.8 \times 10^{19}$
$ f / C_{LP} $	2	5.3	42.6	2,184.5	$6.2 \times 10^5$	$3.3 \times 10^7$

Table 6 shows the truth-vectors of  $LP$ -representative functions for  $n = 2, 3, 4$  in hexadecimal encoding meaning that each character in the truth-vectors should be replaced by its binary representation as a sequence of four binary values.

Table 7 shows the number of  $LP$ -representative functions for up to 6 variables, which can be compared to data in Table 1. Since in any class there are  $n!6^n$  functions, it follows that for  $n$  sufficiently large, the number of  $LP$ -classes approximates to  $2^{2^n}/n!6^n$ . It should be noticed that  $LP$ -classification is the stronger than  $NPN$  and  $SD$  classifications, since considerably reduces the number of different classes.

## 7 Classification by Walsh Coefficients

In 70's, synthesis with threshold logic elements has been widely exploited, since a relatively large class of functions (threshold functions) can be realized by a single threshold element. In connection with this, classification in terms of *Walsh spectral coefficients* has been considered, since threshold functions of  $n$  variables can be characterized by  $(n + 1)$  out of the total of  $2^n$  Walsh coefficients [9], [13], [14]. These coefficients can be identified with Chow parameters, although these parameters were introduced independently without relating them to the Walsh transform [6].

In Walsh analysis the following operations are called the *translation invariant spectral operations*

1. Permutation of variables  $x_i$  by  $x_k, i \neq k \neq 0$ ,
2. Replacement of a variable by its complement  $x_i \rightarrow \bar{x}_i$ ,
3. Replacement of a function  $f$  by its complement  $f'$ ,
4. Replacement of a variable  $x_i$  by a linear combination of variables  $x_i \oplus x_k$ , or in more general case, replacement of  $x_i$  by  $(x_a \oplus x_b \oplus \dots \oplus x_h) \oplus x_i$ , which is called *spectral translation*,
5. Replacement of a function  $f(x_1, \dots, x_n)$  by the function  $x_i \oplus f'(x_1, \dots, x_n)$  where  $f'$  is uniquely defined by a particular transformation over indices of Walsh spectral coefficients for  $f$ . This operation is called the *disjoint spectral translation*.

In other words, spectral translation states that given a function and its spectrum, linear (mod 2) transformation of arguments corresponds to a particular permutation of spectral coefficients. The disjoint spectral translation states that given a function  $f$  and its Walsh spectrum  $S_f$ , the spectrum of a new function  $f'$  obtained by modulo 2 addition of the original function and some of its arguments, corresponds also to a permutation in the Walsh spectrum  $S_f$ . A generalization of the disjoint spectral translation to multiple-valued logic functions has been presented in [23].

The above operations have been used to define classes of *disjointly translationally equivalent* functions, as functions which can be derived from each other by permutation of variables, logic negation of variables, negation of function values, application of spectral translation and disjoint spectral translation. It is assumed that these operations can be applied simultaneously and repeatedly. It has been shown that for  $n \leq 4$ , there are 8 classes of functions, and the same number of representative functions, out of which 7 are threshold functions. Thus, they can be realized by a single threshold logic element [9] providing a sufficient number of inputs, or by adding some network of EXOR circuits.

It should be noticed that the classification of switching functions by disjoint spectral translation corresponds to the classification defined in [22] in terms of group theory and Fourier transforms on finite Abelian groups.

It has been shown that  $n + 1$  Walsh coefficients are necessary to characterize all switching functions which can be derived from a threshold logic function by pre- and postlinear translation [25].

In [38], the method of synthesis of any switching function by a threshold logic gate and a set of EXOR gates has been related to the former work in references [16], [17], [21], see also the replay of Edwards to the comment by C.K. Yuen [10]. This method is related to the approximation of switching functions by Walsh series with less than  $2^n$  coefficients, actually  $n + m + 1$  coefficients, with  $m$  a small integer, (called Chebyshev approximation in [16]).

## 8 Classification by Autocorrelation Functions

Classification by autocorrelation functions [26], [27] is closely related to the classification in terms for Walsh coefficients in the sense that exploits spectral translation, i.e., replacement of variables by EXOR sum of pairs of variables. Recall that for a switching function  $f$ , the autocorrelation function is defined as

$$B_f(\tau) = \sum_{i=0}^{2^n-1} f(x)f(x \oplus \tau).$$

In this classification, the set of *NPN*-classification rules is extended by allowing replacement of a variable  $x_i$  by  $x_i \oplus x_j$ ,  $i, j \in \{1, \dots, n\}$ ,  $i \neq j$  and replacing a function  $f$  by  $f \oplus x_i$ ,  $i = 1, \dots, n$ .

It has been shown that the autocorrelation functions in  $\{0, 1\} \rightarrow \{1, -1\}$  encoding are invariant to these classification operations, which permits to correspondingly define the function classes. In this classification, there are 18 equivalence classes for  $n \leq 4$ . Each class is uniquely characterized by an autocorrelation function [27].

Binary decision diagrams (BDDs) have been used for fast checking if two functions belong to the same autocorrelation class. The method exploits property that following performances of a decision diagram are not affected by the autocorrelation classification rules

1. The number of true minterms,
2. The number of essential variables,
3. The shortest path length of the BDD.

It is stated that two functions which belong to the same autocorrelation class have identical these parameters [26].

## 9 Classification by Decision Diagrams

First ideas about classification of switching functions by referring to the *shape of decision diagrams* have been presented in [34]. These considerations were motivated by the feature that logic functions when represented by decision diagrams are easily mapped to technology, with the layout of the circuit directly determined by the shape of the decision diagram [8], [37].

Recall that the shape of a decision diagram is specified by the distribution of nodes per levels and the interconnections of nodes in the diagram [31], [37]. The distribution of nodes can be specified by a vector  $\mathbf{D} = [k_1, \dots, k_{n+1}]$ , where  $k_i$ ,  $i = 1, \dots, n$ , is the number of nodes at the level  $i$  by starting from the root node at the level  $i = 1$ . Thus,  $k_1 = 1$  for any diagram. Interconnections in a diagram are specified as a list of nodes to which the given node is connected for each node.

The basic idea in exploiting decision diagrams in classification of switching functions expressed in [34], consists in the representation of functions representing *LP*-classes by *Binary Decision Diagrams* (BDDs) and *Walsh decision diagrams* (WDDs) [35] in which case the  $\{0, 1\} \rightarrow \{1, -1\}$  encoding of logic values is used. For  $n = 4$ , 15 WDDs of different shape are sufficient to represent 30 *LP*-representative functions, with four constant nodes. If different number of constant nodes is allowed, then 11 WDDs of different shape are sufficient.

It can be shown that few representative functions can be represented by decision diagrams of the same *shape* if permutation of variables, permutation of labels at the edges and change of values of constant nodes is allowed. For instance, in the case of six *LP*-representative functions for  $n = 3$ , five BDDs of different shape are sufficient. In the case of WDDs, three diagrams are sufficient, due to properties of the Walsh spectral coefficients of switching functions in the encoding used. For instance, if  $n = 4$ , These results were exploited in [33] is circuit synthesis, where *Haar Spectral Transform Decision Diagrams*, see for instance, [32] have been also considered.

Considerations related in some sense to this topic, but concerning the *NPN*-classification have been presented in [40]. It has been noticed that out of 14 representative functions for  $n = 3$ , 10 functions depend on all three variables. The BDDs for these functions can be combined in a *Super BDD* which when equipped with selection switches is capable of realizing all the functions for this number of variables. This circuit can be viewed as an *NPN Universal Logic Module* for  $n = 3$ . In the case of functions with four variables, 208 functions out of the total of 222 *NPN*-representative functions depend essentially on all four variables. Their BDDs can be combined in a *Super BDD* to realize *NPN*-functions. The main problem is that although the number of bits that is required to encode all functions realized by an *Super BDD* is smaller than encoding in realizations by Look-up-Tables (LUTs), the realization of the corresponding modules may be much larger and slower than that of LUTs [40]. The representation of *NPN*-representative functions by *Functional Decision Diagrams* (Reed-Muller decision diagrams) [30] requires smaller number of non-terminal nodes compared to BDDs, and thus a simplified *Super* decision diagram however, with Reed-Muller decomposition rule at the nodes [40].

## 10 Conclusions

Splitting and most often partitioning of the set of all switching functions of a given number of variables into classes of functions similar to each other with respect to some criteria of similarity proved useful in solving many problems related to the representations, realizations, and subsequently applications of switching functions. The chief background idea is that a class of functions is represented by a single representative function, from which all other functions in the same class can be derived by the application of classification rules used to define the classes of functions.

Classification rules are derived depending on the applications intended and they are determined by the contemporary technology. In relatively short history of switching theory classification rules have been formulated in different ways to serve some particular purposes as well as have been adapted to various representations of switching functions. Further requirements are that the number of different classes should be reasonably small with the classification performed by few possibly simple classification rules. Attempts to fulfill these opposite criteria provided different classification methods for switching functions.

In this paper, we reviewed few different former and recent classifications and believe that this review can be a good basis for a further study and research in this theory.

## Acknowledgments

The authors are grateful to the reviewers whose constructive comments were useful in improving the presentation in this paper.

This work was supported by the Academy of Finland, Finnish Center of Excellence Programme, Grant No. 213462.

## References

- [1] Aizenberg I.N., Aizenberg N.N., Vandewalle J. *Multi-Valued and Universal Binary Neurons - theory, Learning, Applications*, Kluwer Academic Publishers, Boston, Dordrecht, London, 2000.
- [2] Aizenberg, N.N., Ivaskiv, Yu.L., *Multivalued Threshold Logic* Naukova Dumka, Kiev, Ukraine, 1977, in Russian.
- [3] Astola, J.T., Stanković, R.S., *Fundamentals of Switching Theory and Logic Design*, Springer, 2006.
- [4] Chang, C.H., Falkowski, B.J., "NPN-classification using weight and literal vectors of Reed-Muller expansion", *Electronics Letters*, Vol. 35, No. 10. 1999, 798-799.
- [5] Cahng, C.H., Falkowski, B.J., "Reed-Muller weight and literal vectors for NPN-classification", *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '99*, May 30 - June 2, 1999, Vol. 1, 379-382.
- [6] Chow, C.K., "On the characterization of threshold functions", *Proc. IEEE Symp. Switching Theory and Logic Design*, 1961, 34-38.
- [7] Dautović, S., Novak, L., "A comment on 'Boolean classification via Fixed-polarity Reed-Muller Form'", *IEEE Trans. Computers*, Vol. C-55, No. 8, 2006, 1067-1069.
- [8] De Micheli, G., *Synthesis and Optimization of Digital Circuits*, McGraw Hill, 1994.
- [9] Edwards, C.R., "The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis", *IEEE Trans. Computers*, Vol. C-24, No. 1, 1975, 48-62.

- [10] Edwards, C.R., "Author's Reply to the Comment by Yuen [39]", *IEEE Trans. Computers*, 1976, 767.
- [11] Hurst, S.L., "The application of Chow parameters and Rademacher-Walsh matrices in the synthesis of binary functions", *Comput. J.*, Vol. 16, No. 2, 1973.
- [12] Golomb, S.W., "On the classification of Boolean functions", *IRE Trans. Circuit Theory*, Vol. CT-6, 1959, 176-186.
- [13] Hurst, D.L., *Logical Processing of Digital Signals*, Crane Russak and Edward Arnold, London and Basel, 1978.
- [14] Hurst, S.L., Miller, D.M., Muzio, J.C., *Spectral Techniques in Digital Logic*, Academic Press, Bristol, 1985.
- [15] Jain, A., Bryant, R., "Inverter minimization in logic networks", *Proc. Int. Workshop on Logic Synthesis*, 1993, 9c1-9c6.
- [16] Kaplan, K.R., Winder, R.O., "Chebyshev approximation and threshold functions", *IEEE Trans. Electron. Comput.*, (Short Notes), Vol. EC-14, 1965, 250-252.
- [17] Kascerman, R., "A linear summation threshold device", *IEEE Trans. Electron. Comput.*, (Corresp.), Vol. EC-12, 1963, 914-915.
- [18] Koda, N., Sasao, T., "An upper bound on the number of products in minimum ESOPs", *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansions in Circuit Design (Reed-Muller 95)*, Makuhari, Japan, August 1995, 27-29.
- [19] Posthoff, Ch., Steinbach, B., "Logic Functions and Equations - Binary Models for Computer Science". Springer, Dordrecht, The Netherlands, 2004.
- [20] Koda, N., Sasao, T., "A method to simplify multiple-output AND-EXOR expressions,"(in Japanese), *Trans. IEICE*, Vol. J79-D-1, No. 2, 1996, 43-52.
- [21] Lawson, C.L., *Contributions to the Theory of Linear Least Maximum Approximation*, Ph.D. dissertation, Univ. California, Los Angeles, 1961.
- [22] Lechner, R.J., "Harmonic analysis of switching functions", in Mykhopadhyay, A., (ed.), *recent Developments in Switching Theory*, Academic Press, 1971, 122-228.
- [23] Moraga, C., "Introducing disjoint spectral translation in spectral multiple-valued logic design", *Electronics Letters*, Vol. 14, No. 8, 1978, 241-243.
- [24] Muorga, S., *Logic Design and Switching Theory*, Jon Wiley and Sons, 1979, Reprinted edition Krieger Publishing Company, Malabar, FL, 1990.
- [25] Muzio, J.C., Miller, D.M., Hurst, S.L., "Number of spectral coefficients necessary to identify a class of Boolean functions", *Electronics Letters*, Vol. 18, No. 13, 1982, 577-578.
- [26] Rice, J.E., "First thoughts on determining a method for fast autocorrelation classification", *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, August 24-26, 2005, 661-664.
- [27] Rice, J.E., Muzio, J.C., "Use of the autocorrelation in the classification of switching functions", *Euromicro Syp. on Digital System design, Architectures, Methods and Tools (DSD)*, 2002, 244-251.
- [28] Sasao, T., *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.
- [29] Sasao, T., Besslich, Ph.W., "On the complexity of mod 2 PLAs", *IEEE Trans. Comput.*, Vol. C-39, No. 2, 1991, 262-266.
- [30] Sasao, T., Fujita, M., (eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996.
- [31] Stanković, R.S., "Some remarks on basic characteristics of decision diagrams", *Proc. 4th Int. Workshop on Applications of Reed-Muller Expansion in Circuit Design*, August 20-21, 1999, Victoria, B.C., Canada, 139-146.
- [32] Stanković, R.S., Egiazarian, K., Astola, J.T., "Recent development in Haar wavelet transform for application to switching and multiple valued logic representations", in Astola, J., Yaroslavsky, L., (eds.), *Advances in Signal Transforms - Theory and Applications*, EURASIP Book Series on Signal processing and Communications, Hindawi Publishing Corporation, Japan, 2007, 57-92.
- [33] Stanković, R.S., Karpovsky, M.G., "Remarks on the number of logic networks with same complexity derived from spectral transform decision diagrams", *Proc. Int. TICSP Workshop on Spectral Methods and Multirate Signal Processing, SMMSP'02*, Toulouse, France, September 7-8, 2002, 163-170.
- [34] Stanković, R. S., Moraga, C., Sasao, T., "Spectral transform decision diagrams", *Proc. IFIP WG 10.5 Workshop on Application of the Reed-Muller Expansion in Circuit Design, Reed-Muller'95*, August 27-29, 1995, Makuhari, Chiba, Japan, 46-53.
- [35] Stanković, R.S., Sasao, T., Moraga, C., "Spectral transform decision diagrams" in [30], 55-92.
- [36] Tsai, Ch.-Ch., Marek-Sadowska, M., "Boolean function classification via Fixed-polarity Reed-Muller forms", *IEEE Trans. Computers*, Vol. C-46, No. 2, 1997, 173-186.
- [37] Yanushkevich, S.N., Miller, D.M., Shmerko, V.P., Stanković, R.S., *Decision Diagram Technique for Micro- and Nanoelectronic Design*, CRC Press, Taylor & Francis, Boca Raton, London, New York, 2006.
- [38] Yuen, C.K., "Function approximation by Walsh series", *IEEE Trans. Computers*, Vol. C-24, 1975, 590-598.
- [39] Yuen, C.K., "Comments on \*The application of the rademacher-Walsh transform of Boolean function classification and threshold logic synthesis\*", *IEEE Trans. Computers*, July 1976, 766-767.
- [40] Zilic, Z., Vranesic, Z., "Using decision diagrams to design ULMs for FPGAs", *IEEE Trans. Computers*, Vol. C-47, No. 9, 1998, 971-982.