A CANONICAL AND/EXOR FORM THAT INCLUDES BOTH THE GENERALIZED REED-MULLER FORMS AND KRONECKER FORMS

Marek Perkowski, Lech Jóźwiak †, Rolf Drechsler +

Portland State University, Dept. of Electr. Engn., Portland, Oregon 97207,
Tel: 503-725-5411, Fax: 503-725-4882, *mperkows@ee.pdx.edu*
† Faculty of Electrical Engineering, Eindhoven University of Technology,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands, *lech@eb.ele.tue.nl*
+ Institute of Computer Science, Albert-Ludwigs-University,
79110 Freiburg in Breisgau, Germany, *drechsle@informatik.uni-freiburg.de*

Abstract— The paper gives a positive answer to an open problem - whether there exists some family of canonical forms that would include both the Generalized Reed-Muller Forms and the Kronecker Forms as its special cases. We call the new forms Generalized Kronecker Forms. We present also the corresponding decision diagrams. They can be used for representation of large functions and also for optimal synthesis of highly testable two-level and multilevel circuits in several technologies, including Field Programmable Gate Arrays (FPGAs).

I. INTRODUCTION

Several canonical families of AND/EXOR forms have been investigated, many of them (but not all) can be derived from binary AND/EXOR decision diagrams. All the circuit structures that use exclusively AND and EXOR gates and that were developed based on the AND/EXOR representations, belong to the highest testable circuits ever invented [18, 17]. Since EXOR gates are quite expensive in many technologies, AND/EXOR representations are not yet broadly used in custom VLSI design. We stress, however, that the new representations introduced here can be also used for synthesis with arbitrary gates. In the first phase of the entire synthesis process the AND/EXOR circuit is created. Then, it is mapped to an arbitrary technology that may involve AND, OR, EXOR and other gates. Circuits obtained in such a way continue to be highly testable [9]. In the mapping phase, the trade-offs between the implementation cost and the testability are investigated and utilised to satisfy various requirements for a given technology.

Families of canonical AND/EXOR forms, subsets of general ESOPs expressions, have been studied by many authors, because they display interesting trade-offs between testability, number of terms, area, and speed [21]. An open problem was posed by Tsutomu Sasao - *"whether there exists some family, or families, of canonical forms that ia a superset of both KRO and GRM families?"*. Such family would be interesting because presumably it would be close in cost to ESOP and still have good testability properties similar to GRM. In this paper we will answer positively to this questions and we will develop a new representation that can be used in the first stage of the proposed logic synthesis process - the "technology independent, EXOR synthesis" phase, which will be followed by the "EXOR-related technology mapping" [25, 9, 19, 26], not discussed here. Because the class of circuits covered by the new representation includes all the AND/EXOR circuits obtained from the previously known canonical representations, the new representation should lead to superior results, in terms of: the two-level realization areas for forms, the diagrams complexity, and the complexity of multilevel circuits corresponding to the diagrams after technology mapping.

Section II will give the preliminaries on EXOR type decision trees and corresponding canonical forms. In section III, on Canonical Expansions for Many Variables, the insufficiency of the **binary tree model** (i.e. trees with single variable expansions in nodes) as a base of high quality AND/EXOR representations will be explained. First we present a GRM as an example of a non-decomposable expansion for many variables that corresponds to a node in our new representation. GRM expansion is *non-decomposable,* because it cannot be presented as a composition of canonical single-variable expansions. In section IV we introduce the GRM Universal Logic Module (node) and show how to compute the expansion data functions for such multi-variable nodes in the process of tree creation. Section V will introduce Generalized Kronecker Tree Decision Diagrams and Forms. Section VI shows that the new forms are essentially better than all known canonical forms and section VII presents how these results can be applied to exact ESOP minimization. Section VIII concludes the paper.

II. AND/EXOR CANONICAL TREES WITH SINGLE-VARIABLE EXPANSION NODES.

The first fundamental idea of generalizing **the functions** realised by the nodes of the tree comes from Davio [1], who introduced the following three fundamental expansions realised by nodes:

$$
\begin{aligned}
f(x_1,x_2,...,x_n) &= \overline{x_1} f_0(x_2,...,x_n) \oplus x_1 f_1(x_2,...,x_n) \\
\text{in short} \quad f &= \overline{x_1} f_0 \oplus x_1 f_1 && (2.1) \\
f(x_1,x_2,...,x_n) &= 1 \cdot f_0(x_2,...,x_n) \oplus x_1 f_2(x_2,...,x_n) \\
\text{in short} \quad f &= f_0 \oplus x_1 f_2 && (2.2) \\
f(x_1,x_2,...,x_n) &= 1 \cdot f_1(x_2,...,x_n) \oplus \overline{x_1} f_2(x_2,...,x_n) \\
\text{in short} \quad f &= f_1 \oplus \overline{x_1} f_2 && (2.3)
\end{aligned}
$$

where f_0 is f with x_1 replaced by 0 (negative cofactor of variable x_1), f_1 is f with x_1 replaced by 1 (positive cofactor of variable x_1), and $f_2 = f_0 \oplus f_1$. All of them generalize the Shannon expansion. Equation (2.1) is the Shannon Expansion with inclusive OR operator replaced with EXOR operator. This can be done because functions $\overline{x_1} f_0$ and $x_1 f_1$ are disjoint. We will denote a node corresponding to the Shannon Expansion by S. Equation (2.2) represents the Positive Davio Expansion, pD for short, and (2.3) represents the negative Davio Expansion, nD, for short.

Let us observe that S uses both a variable and its negation, pD uses only the positive literal of the variable, and nD uses only the negative literal of the variable. The circuit realisation of S is a multiplexer. Realisation of pD includes a two-input AND and a two-input EXOR gate and is called AND/EXOR gate. Realisation of nD includes a two-input AND gate with inverted input for the control variable x_1, called the inhibition gate, and the EXOR gate. It is called the Inhibition/EXOR gate. All such gates exist in Fine Grain FPGAs and in modern VLSI libraries.

An example of a generalized decision tree (we will call it the Generalized Kronecker Tree) is shown graphically in Figure 1. The S, pD and nD expansions are single-variable nodes: for variable x_1 in the first level, and for variable x_3 in the third level. The output function of a node is shown by it, the edges on the bottom of the node correspond to the two inputs of a node - these are the

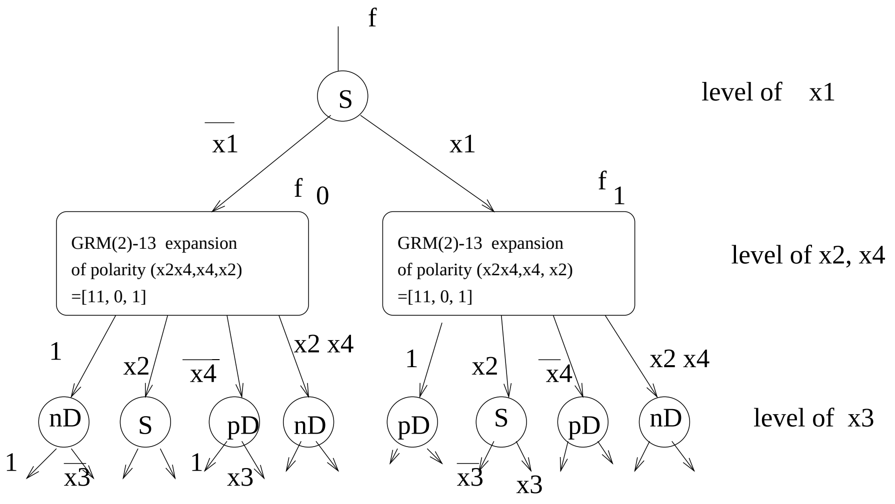

Figure 1: *Example of a Generalized Kronecker Tree*

(respective to the expansion type) functions from (2.1) - (2.3) and are taken from the set f_0, f_1, f_2. The expansion literals are shown near the input edge(s). Label 1 corresponds to function f_0 in pD and f_1 in nD. Label $\overline{x_1}$ to f_0 in S and f_2 in nD. Label x_1 to f_1 in S and f_2 in pD. By applying recursively expansions (2.1) - (2.3), or any subset of them, to the function, its cofactors with respect to the first level variable of the function, second level variable cofactors, third level cofactors, and so on, various types of binary decision trees can be created [24]. The meaning of multi-variable nodes GRM(2)-13 for variables x_2, x_4 will be explained in the next section.

III. CANONICAL EXPANSIONS FOR MANY VARIABLES

A. Insufficiency of the single-variable expansion tree models.

Three expansions, S, pD and nD, are the basis of all single-variable EXOR- based representations of logic functions, and have been therefore investigated in detail by many researchers. For instance, many canonical AND/EXOR based expansions for a function are obtained by repetition of some, or all, of these expansions for all input variables of this function. All these expressions, including minterm expansion, positive Davio expansion, Fixed Polarity Reed-Muller expansion (FPRM) and Kronecker expansion (KRO), are called Kronecker family of families of expansions (expressions) [7]. It can be observed, however, that there exists a peculiar canonical EXOR-based expression, called the Generalized Reed-Muller Expansion (GRM), that cannot be obtained by a composition of canonical expansions for single variables, but is obtained by certain simultaneous expansion for ALL variables [24, 2, 31, 27, 28]. This expansion does not belong to the Kronecker family of families of expansions. GRM expansions, as proved by Debnath and Sasao [2], are less complex on average and on the worst case than the Kronecker type of expansions, and are therefore of high practical interest in recent years. However, until now, they have not been linked to the general theory of tree expansions, their respective diagrams, and the flattened (two-level) expressions obtained from them. Below, the generalized canonical expansions will be presented, that will include these Generalized Reed-Muller Expansions as their special cases. Also, the new kinds of expansions applied in nodes of the decision trees will be introduced. These new *multi-variable node expansions* will generalize the three expansions (2.1) - (2.3) that have been used in the classical representations and in the based on them synthesis methods. The generalization of tree representation will be achieved by a simple conceptual extension to the known binary trees: in addition to single-variable nodes in the trees, the nodes with GRM expansions for several variables will be also allowed, see Figure 1.

B. Generalized Reed-Muller Expansion

An arbitrary n-variable function $f(x_1, x_2, ..., x_n)$ can be represented as

$$f(x_1, ..., x_n) \;=\; a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus ...\oplus a_nx_n \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus \\ ...\oplus a_{n-1,n}x_{n-1}x_n \oplus .... \oplus a_{12...n}x_1x_2x_3...x_n \quad (3.1)$$

where a_i's are either 0 or 1.

Formula (3.1) is a Positive Polarity Reed-Muller Expansion (PPRM). For a given function f, the coefficients a_i are uniquely determined. PPRM is thus a canonical representation. The number of products in (3.1) is 2^n, and all the literals are positive (uncomplemented). In (3.1), for each variable x_i ($i =$ 1,2,... n), if we use either a positive literal (x_i) throughout or a negative literal ($\overline{x_i}$) throughout, then we have a Fixed Polarity Reed-Muller Expansion (FPRM). For each variable x_i there are two ways of choosing the polarities: positive or negative. Therefore, 2^n different combinations of values, called *polarities*, exist for an n-variable function. Thus, every FPRM is a canonical representation and there are 2^n FPRMs for a function. Let us observe, that each of these FPRMs can be obtained by a step-by-step expansion of the function using expansions from set (2.1) - (2.3), one variable at a time.

Definition 1. The **Generalized Reed-Muller Expansion** (GRM) is obtained by freely choosing in (3.1) the polarity for each individual literal.

Thus, in GRM, and contrary to FPRMs, the same variable can stand in both positive and negative polarities. There are $n2^{n-1}$ literals in (3.1), so there are $2^{n2^{n-1}}$ polarities for n-variable function. Each of the polarities determines a unique set of coefficients, and thus a canonical representation of a logic function. Let us observe that there are many more GRMs than FPRMs for a function, thus the minimal GRM (among the GRMs for all the possible polarities) is not worse (and is usually much better) than the minimal FPRM (among all the respective polarities) of the same function. However, it is more difficult to find a good GRM. GRM cannot be found by steps executed for each variable separately, but for all the variables together.

Next section will explain how to compute these multi-variable GRM expansions, assuming however that coefficients a_i in (3.1) are no longer binary constants, but functions of some variables.

IV. COMPUTATION OF THE DATA FUNCTIONS FOR A GIVEN GRM POLARITY

The *GRM Universal module*, introduced here, computes the output function $f(x_1, ..., x_m, ..., x_n)$ from the values of expansion variables and subfunctions SF_i in the following GRM expansion formula with functional coefficients.

$$f(x_1, x_2, ..., x_m, ..., x_n) \;=\; SF_0(x_{m+1}, ..., x_n) \oplus \hat{x_1}SF_1(x_{m+1}, ...x_n) \oplus \hat{x_2}SF_2(x_{m+1}, ..., x_n) \oplus ... \\ \oplus\ \hat{x_m}SF_m(x_{m+1}, ..., x_n) \oplus \hat{x_1}\hat{x_2}SF_{12}(x_{m+1}, ..., x_n) \oplus \hat{x_1}\hat{x_3}SF_{13}(x_{m+1}, ..., x_n) \oplus \\ ...\hat{x_{m-1}}\hat{x_m} \oplus SF_{m-1,m}(x_{m+1}, ..., x_n) \oplus .... \oplus \hat{x_1}\hat{x_2}\hat{x_3}...\hat{x_m}SF_{12...n}(x_{m+1}, ..., x_n) \quad (4.1)$$

where $\hat{x_i}$ means variable x_i with negation or not (i.e. variable x_i in arbitrary polarity).

Let us observe that formula (4.1) for function $f(x_1, x_2, ..., x_m, ..., x_n)$ is similar to (3.1) with the difference that now: (1) the expansion is with respect to variables $x_1, ..., x_m$, (2) polarities of all

literals in product terms are arbitrary, and (3) coefficients a_i are now replaced with subfunctions SF_i of remaining variables $x_{m+1}, ...x_n$. Observe also that functions $SF_i(x_{m+1}, ..., x_m)$ are **not** the respective cofactors of function f, as it would be in case of variable-decomposable expansions, FPRM or KRM, but are **computed together** for every GRM polarity.
In this section we will explain how GRM expansion is calculated from given function $f(x_1,..., x_m,...,x_n)$, for only a subset of its variables $(x_1,...,x_m)$. This calculation is crucial for the method introduced below, to expand efficiently any function $f(x_1, ..., x_m, ..., x_n)$ with respect to given product terms $\hat{x_1}, ..., \hat{x_m}$ of arbitrary GRM on the subset of the function's variables. As far as we know, this generalization of GRM expansion has been yet not discussed in the literature.

We will use methods of Linearly Independent logic [12, 13, 14] to uniquely derive functions SF_i from the original function $f(x_1, x_2, ..., x_n)$, assuming given sets of products of literals $\hat{x_1}, ..., \hat{x_m}$. These methods are very powerful and generic: they can be applied to **every** field extension.

We create a $2^m \times 2^m$ matrix M with rows corresponding to minterms (for a function with m variables we have 2^m columns). The columns correspond then to some Boolean functions of m variables. A 1 in the intersection of a column "i" and row "j" means that minterm "j" is in function "i". The set of columns can be linearly independent with respect to EXOR operation (i.e. columns are bit-by-bit exored). If a set of 2^m columns is linearly independent then there is one and only one matrix M^{-1}, inverse to M with respect to exoring operation. In such case, the family of Boolean functions corresponding to columns will be called the "linearly independent family of Boolean functions" (or set of LI Boolean functions, or LI set), and the matrix will be called a "nonsingular matrix".

Let us denote the vector of minterms by FV. CV denotes the vector of coefficients for some given canonical form represented by nonsingular M. Given is an arbitrary linearly independent (LI) set of 2^m Boolean functions f_i of m variables. This set can be represented as a $2^m \times 2^m$ nonsingular matrix M with functions f_i as columns, $i = 0, ..., 2^m - 1$.

Example 4.1. Let us assume that for $f(A, B)$ the LI functions are AB, $\overline{B}$, $\overline{A}$, and 1. Because m=2, the minterms are in 2^m-dimensional space, $2^m = 4$. The rows of matrix M from top to bottom correspond to the following minterms in m = 2-dimensional space: $\overline{A}\ \overline{B}$, $\overline{A}\ B$, $A\ \overline{B}$, and $A\ B$, respectively. Thus, the $2^m \times 2^m$ matrix M is as follows:

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

The first column corresponds to function AB, the second column to function $\overline{B}$, the third column to function $\overline{A}$, and the last column to function 1. It is easy to check that M is nonsingular with respect to EXOR-ing operation, and that it has exactly one inverse matrix M^{-1}.

Theorem 1. Given is a function F$(x_1, ..., x_m, ..., x_n)$ such that the set of input variables $\{x_1, ..., x_n\}$ includes properly the set $\{x_1, ..., x_m\}$. There exists an unique expansion (called an LI expansion):

$$F(x_1, ..., x_n) = f_0(x_1, ..., x_m)SF_0(x_{m+1}, ..., x_n) \oplus f_1(x_1, ..., x_m)SF_1(x_{m+1}, ..., x_n) \oplus .... \\ ... \oplus f_{2^n-1}(x_1, ..., x_m)SF_{2^n-1}(x_{m+1}, ..., x_n), \quad (4.2)$$

where functions f_i are the given LI functions of m variables, and the coefficient functions (called also the "data input functions") SF_i of the remaining input variables are determined from the coefficient vector CV = $M^{-1} \times$ FV, where $FV(x_{m+1}, ..., x_n)$ is a vector of all 2^m cofactors of F with respect to variables from the set $\{x_1, ..., x_m\}$. In general, M is the matrix of 2^m cofactors of F with respect to variables from the set $\{x_1, ..., x_m\}$. Thus, when $m = n$, the cofactors with respect to variables

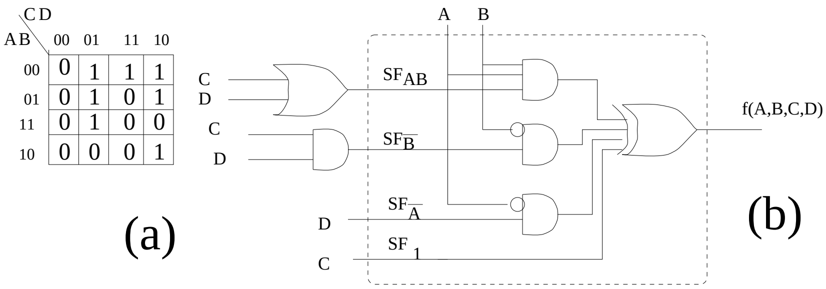

Figure 2: *(a) Function $f(A,B,C,D)$ to Example 4.2, (b) Universal GRM expansion module to Example 4.1.*

$x_1, ..., x_m$ become minterms on these variables, as in Example 4.1 above.

Proof. Proof is a generalization of the method of solving EXOR logic equations from Example 4.2 below. [1]

It can be observed that, because the set of product terms $\hat{x_1}\hat{x_2}\hat{x_3}...\hat{x_m}$ is a linearly independent family of functions, the GRM expansion with functional coefficients (4.1) is a special case of the LI expansion with functional coefficients (4.2). This is an unique expansion for the set of variables $x_1, ..., x_m$ and the linearly independent functions on variables $x_1, ..., x_m$. Thus, the data functions SF_i are also unique.

Example 4.2. Given is function $f(A,B,C,D)$ from Fig. 2a. Let us assume that the GRM expansion is with respect to variables $\{A,B\}$. In order to calculate functions $SF_i(C,D)$ we will expand function $f(A,B,C,D)$ with respect to these variables twice. The first expansion will use the computable from function $f(A,B,C,D)$ standard cofactors: $f_{\overline{A}\ \overline{B}}(C,D)$, $f_{\overline{A}\ B}(C,D)$, $f_{A\ \overline{B}}(C,D)$, $f_{AB}(C,D)$. The second expansion will use unknown functions $SF_i(C,D)$. This will lead to a set of linear logic equations, which after solving will give the values to the unknown functions $SF_i(C,D)$. The basis of LI functions of a GRM expansion is here arbitrarily selected as: $f_{AB} = AB$, $f_{\overline{B}} = \overline{B}$, and $f_{\overline{A}} = \overline{A}$, and $f_1 = 1$.

Thus

$$\overline{A}\ \overline{B}\ f_{\overline{A}\ \overline{B}}(C,D) \oplus\ \overline{A}\ B f_{\overline{A}\ B}(C,D) \oplus\ A\ \overline{B} f_{A\ \overline{B}}(C,D) \oplus\ A\ B\ f_{AB}(C,D)$$
$$= \overline{A}\ \overline{B}\ f(A,B,C,D) \mid_{A=0,B=0} \oplus\ \overline{A}\ B\ f(A,B,C,D) \mid_{A=0,B=1} \oplus\ A\ \overline{B}\ f(A,B,C,D) \mid_{A=1,B=0} \oplus$$
$$A\ B\ \ f(A,B,C,D) \mid_{A=1,B=1} = \overline{A}\ \overline{B}\ (C+D) \oplus\ \overline{A}\ B(D \oplus C) \oplus\ A\ \overline{B}(C\overline{D}) \oplus\ \ A\ B\ (D\overline{C})$$
$$= AB\ SF_{AB}(C,D) \oplus\ \overline{B}\ SF_{\overline{B}}(C,D) \oplus\ \overline{A}\ SF_{\overline{A}}(C,D) \oplus\ SF_1(C,D)$$

By substituting in the above equation A = 0, B = 0, we get

$SF_{\overline{B}} \oplus SF_{\overline{A}} \oplus SF_1 = f_{\overline{A}\ \overline{B}}(C,D) = (C+D)$. By substituting A = 0, B = 1, we get $SF_{\overline{A}} \oplus SF_1 = f_{\overline{A}\ B}(C,D) = (C \oplus D)$. By substituting A = 1, B = 0, we get $SF_{\overline{B}} \oplus SF_1 = f_{A\ \overline{B}}(C,D) = (C\overline{D})$. By substituting A = 1, B = 1, we get $SF_{AB} \oplus SF_1 = f_{A\ B}(C,D) = (\overline{C}D)$.

Hence we obtain the following equation for cofactors $f_{A^i\ B^j}(C,D)$: $FV = M \times CV =$

$$= \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} SF_{AB}(C,D) \\ SF_{\overline{B}}(C,D) \\ SF_{\overline{A}}(C,D) \\ SF_1(C,D) \end{bmatrix} = \begin{bmatrix} f_{\overline{A}\ \overline{B}}(C,D) \\ f_{\overline{A}\ B}(C,D) \\ f_{A\ \overline{B}}(C,D) \\ f_{A\ B}(C,D) \end{bmatrix}$$

[1]This Theorem can be expanded for arbitrary field and thus create a basis of constructive logic minimization for logic algebra over this field.

Therefore CV $= M^{-1} \times FV =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} f_{\overline{A}\ \overline{B}}(C,D) \\ f_{\overline{A}\ B}(C,D) \\ f_{A\ \overline{B}}(C,D) \\ f_{A\ B}(C,D) \end{bmatrix} = \begin{bmatrix} SF_{AB}(C,D) \\ SF_{\overline{B}}(C,D) \\ SF_{\overline{A}}(C,D) \\ SF_1(C,D) \end{bmatrix}$$

From Theorem 1, the corresponding GRM expansion for ULM is:

$$f = AB\ SF_{AB}(C,D) \oplus \overline{B}\ SF_{\overline{B}}(C,D) \oplus \overline{A}\ SF_{\overline{A}}(C,D) \oplus SF_1(C,D) \quad (4.3)$$

and the coefficients $SF_i(C,D)$ are taken from the above vector CV.

Example 4.3. Let the function F be represented by a vector $FV^T = [(C+D) \quad (C \oplus D) \quad (C\overline{D}) \quad (\overline{C}D)\ ]$.

$$\text{CV} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} (C+D) \\ (C \oplus D) \\ (C\overline{D}) \\ (\overline{C}D) \end{bmatrix} = \begin{bmatrix} S_{AB} \\ S_{\overline{B}} \\ S_{\overline{A}} \\ S_1 \end{bmatrix} = \begin{bmatrix} (C+D) \\ (C+D) \oplus (C \oplus D) \\ (C+D) \oplus (C \oplus D) \oplus (\overline{C}D) \\ (C+D) \oplus (C \oplus D) \oplus (C\overline{D}) \end{bmatrix} = \begin{bmatrix} (C+D) \\ (CD) \\ (D) \\ (C) \end{bmatrix}$$

Then $f(A,B,C,D) = (AB)(C+D) \oplus \overline{B}(CD) \oplus \overline{A}(D) \oplus (C)$ (4.4)

Observe, that to calculate new GRM expansion with functional coefficients from M^{-1} and FV^T, one has always to execute certain EXOR-ing on a subset of the set of all cofactors from FV^T. These cofactors can be functions of many variables, so it is important to execute this operation efficiently.

In a circuit the GRM expansion with functional coefficients is realized using universal logic module for GRM expansion with respect to variables A, B (illustrated in Fig. 2b). This way, for the set of LI functions (products of literals in this case), $\{\overline{A}, \overline{B}, AB, 1\}$, there exists **only one** GRM expansion specified by its matrix M^{-1}. From this expansion the following ESOP can be found:

$$AB[C \oplus \overline{C}D] \oplus \overline{B}CD \oplus \overline{A}D \oplus C = ABC \oplus AB\overline{C}D \oplus \overline{B}CD \oplus \overline{A}D \oplus C$$

Let us observe that formula (4.3) describes **only one** of 16 GRM expansions for variables A, B, and that 16 possible GRM universal modules exist for two variables. In addition, any set of two or three variables out of set $\{A, B, C, D\}$ can be selected for the first level GRM expansion. So, there are many different trees representing successive expansions. Observe also that matrices are used here only for dydactic purpose, and expansions can be executed directly on any representation of functions, including ESOP cubes and BDDs, because all what is needed is: **cofactoring, exoring** and **solving linear equations**.

V. GENERALIZED KRONECKER TREES, DECISION DIAGRAMS AND FORMS.

The Kronecker Tree has levels that correspond to single input variables and only one of three types of binary expansions is used in every level [23]. However, such trees cannot generate the Generalized Reed-Muller expressions, and thus both the two-level and multi-level circuits corresponding to Kronecker Trees can be far from the minimum. Similarly, the decision diagrams that are created by applying reductions to nodes of such trees have also non-minimum numbers of nodes. By allowing to have nodes in the trees for **sets** of variables, instead for single variables only, the concept of the tree is now generalized, and the tree is no longer a binary tree. This new type of tree will be called the Generalized Kronecker Tree, (GKT), to reflect its properties.

Definition 2. The **Generalized Kronecker Tree** is a multi-variable expansion node tree created as follows:

1) The set of all n input variables is partitioned into disjoint and nonempty subsets S_j, called *blocks*, such that the union of all these subsets forms the initial set. The blocks are disjoint and non-empty. If each block includes just a single variable, the tree reduces to the special case of a KRO Tree. If there is only one block that includes all variables, the tree reduces to the special case of a GRM.

2) The blocks are ordered, each of them corresponds to a level of the tree.

3) For every level, if the block involves a single variable, S, nD, or pD is selected for its nodes. If the block is multi-variable, one GRM expansion polarity is selected for nodes.

In the name "Generalized Kronecker Tree", the component "Kronecker" comes from the fact that, similarly to the Kronecker tree, the levels of GKTs correspond to variables, and at each level, the same expansion type is applied for the same set of variables. But like in the Generalized Reed-Muller Expressions, the expansion in every level can correspond to a **set** of variables with more than one member, and the expansions applied in these nodes are the Generalized Reed-Muller expansions, hence the component - "Generalized" in the name Generalized Kronecker Trees. In GKTs, the set of all input variables is thus partitioned to several blocks (disjoint subsets), each corresponding to a level of the tree. For every block with a single variable, the corresponding expansions are S, pD and nD. For a set with two variables there are $2^{n2^{n-1}}$ GRM expansions, i.e. 16 expansions. Therefore, for the two- variable nodes there are 16 types of nodes, called GRM(2) nodes (expansion types). They will be denoted by GRM(2)-0, GRM(2)-1,...,GRM(2)-15, or as their polarities: [00,0,0], [00,0,1], ...,[11, 1, 1], respectively. To make the above notation easier, we write all bits corresponding to literals in the same product term without separating them with commas; thus, for variables $\overline{x_2}$, $\overline{x_4}$, the part of the polarity 00 represents the term $\overline{x_2}\ \overline{x_4}$. In this way, the polarity GRM(2)-0 = [00,0,0] represents LI functions $\overline{x_2}\ \overline{x_4}, \overline{x_4}, \overline{x_2}$ and **1** of the polarity matrix M (the LI function **1** has no polarity since it has no literals). Polarity GRM(2)-1 = [00,0,1] represents LI functions $\overline{x_2}\ \overline{x_4}, \overline{x_4}$, x_2 and **1** of the polarity matrix M. And so on. It can be shown in a similar way that there are 2^{12} types of GRM(3) nodes for 3 variables.

Definition 3. **Generalized Kronecker Forms** (GK Forms) are obtained by flattening the GK Trees, i.e. finding all ordered product terms obtained by multiplication of paranthesized expressions corresponding to AND/EXOR trees, using recursively the rule $a(b \oplus c) = ab \oplus ac$.

Definition 4. **Generalized Kronecker Decision Diagrams** (GK DDs) are created by: (1) combining isomorphic nodes of any kind, (2) performing standard Ordered Kronecker Functional Decision Diagram (OKFDD) transformations [3] on S, pD and nD nodes, (3) performing generalizations of standard Ordered Kronecker Functional Decision Diagram (OKFDD) transformations [3] on multi-variable nodes. All these generalizations remove nodes that evaluate to their single argument.

Example 5.1. Figure 1 illustrates an example of the Generalized Kronecker Tree (GKT). The first level of the tree has Shannon expansion for variable x_1, the second level has GRM(2)-13 expansion for set of variables $\{ x_2, x_4 \}$ and the third level has S, pD, and nD expansions for variable x_3. As we can verify in Figure 1, the number 13 in expansion name GRM(2)-13 is a natural number corresponding to a binary number 1101, in which the rightmost 1 corresponds to the positive polarity of single variable x_2, 0 corresponds to the negative polarity of variable x_4, and the leftmost two ones correspond to the positive polarities of variables x_2 and x_4 in a two variable AND term for variables x_2 and x_4. The expansion of the node GRM(2)-13 is described by the following formula:

$$f_0(x_2, x_3, x_4) = SF(f_0)_1(x_3) \oplus x_2 SF(f_0)_{x_2}(x_3) \oplus \overline{x_4} SF(f_0)_{\overline{x_4}}(x_3) \oplus x_2 x_4 SF(f_0)_{x_2 x_4}(x_3)$$

where notation $SF(f)_i(X)$ denotes function SF_i, with arguments from the set X of variables, applied

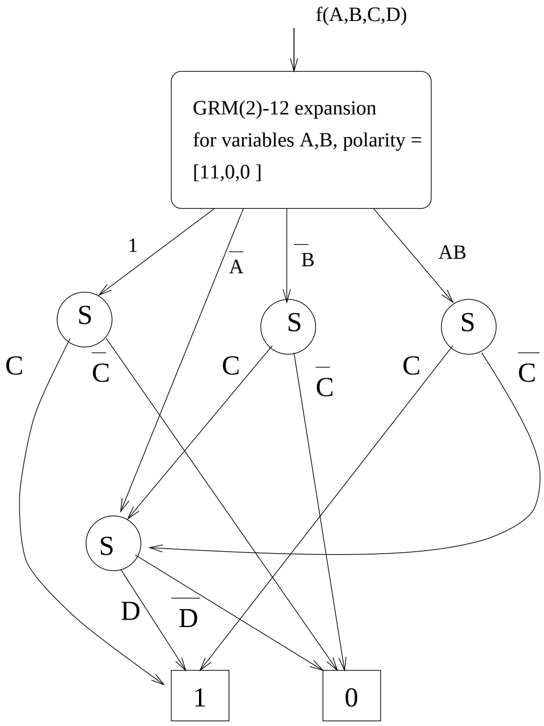

Figure 3: *A Generalized Kronecker Diagram for function from Example 4.1.*

to argument function f.

This formula is the specialization of GRM expansion (4.1) for $f_0(x_2, x_3, x_4)$ with expansion variables x_2, x_4, where subfunctions SFi of the remaining variables are all calculated on the cofactor function f_0 (these are functions: $SF(f_0)_1$,$SF(f_0)_{x_2}(x_3)$, $SF(f_0)_{\overline{x_4}}(x_3)$,$SF(f_0)_{x_2x_4}(x_3)$ in this particular GRM(2)-13 expansion).

We will say that the node evaluates to single argument if after substituting constants and a single variable value H_i to the function realized by this node, after propagation of constants the function evaluates to H_i. Observe that only multivariable nodes that have only logic constants and the same signal H_i as arguments should be evaluated.

For instance:

Formula $\overline{a}\ \overline{b}H_1 \oplus \overline{a}\ bH_1 \oplus a\ \overline{b}H_1 \oplus a\ bH_1$ evaluates to H_1.

Formula $\overline{a}\ b0 \oplus a\ b0 \oplus \overline{b}H_2 \oplus bH_2$ evaluates to H_2.

Formula $ab0 \oplus a0 \oplus b0 \oplus H_3$ evaluates to H_3.

This method is a generalization of simplifications of S, pD and nD nodes applied in OKFDDs.

Example 5.2. A Generalized Kronecker Decision Diagram created from a GKT corresponding to the expansion from Examples 4.1 - 4.3 is shown in Figure 3.

VI. THE QUALITY OF THE GK FORMS.

In this section we will investigate is there any real advantage in the new forms.

Example 5.3 An example of a GK form that is not a GRM and not a KRO is the following

$g(x,y,z) = \overline{x}(\overline{y}\ \overline{z})\ \oplus x(yz\ \oplus\ \overline{y}\ \oplus\ \overline{z}\ \oplus\ 1) = \overline{x}\ \overline{y}\ \overline{z}\ \oplus xyz \oplus x\overline{y} \oplus x\ \overline{z}\ \oplus\ x$

This is not a GRM form, because there are two terms that include the same set of variables $\{x, y, z\}$. The above form can be transformed to one of GRM forms as follows:

$g(x, y, z) = \overline{x}\ \overline{y}\ \overline{z} \oplus xyz \oplus x\overline{y} \oplus x\ \overline{z} \oplus x = (1 \oplus x)(1 \oplus y)(1 \oplus z) \oplus xyz \oplus x\overline{y} \oplus x\ \overline{z} \oplus x$
$= 1 \oplus x \oplus y \oplus z \oplus xy \oplus xz \oplus yz \oplus xyz \oplus xyz \oplus x\overline{y} \oplus x\ \overline{z} \oplus x = 1 \oplus y \oplus z \oplus x \oplus yz \oplus x = 1 \oplus y \oplus z \oplus yz$
$= 1 \oplus y \oplus z\overline{y}\ g1(x, y, z)$

As we see, in this case the GRM form $g1(x, y, z)$ is better than the GK form $g(x, y, z)$.

Form $g(x, y, z) =$ is also not a KRO, because $g(x, y, z) \mid_{x=1} = yz \oplus \overline{y} \oplus \overline{z} \oplus 1$ is not a KRO.

The above form can be transformed to one of KRO forms as follows:

$$\overline{x}(\overline{y}\ \overline{z}) \oplus x(yz \oplus \overline{y} \oplus \overline{z} \oplus 1) = \overline{x}(\overline{y}\ \overline{z}) \oplus x(yz \oplus y \oplus \overline{z}) = \overline{x}(\overline{y}\ \overline{z}) \oplus x(\overline{y}\ \overline{z}) = \overline{y}\ \overline{z} = g2(x, y, z)$$

As we see, in this case the KRO form $g2(x, y, z)$ is better than the GK form $g(x, y, z)$, and better than the GRM form $g1(x, y, z)$.

We demonstrated thus an example of a GK form that is not included in KRO or in GRM families, therefore we proved that GK forms include *properly* the KRO and GRM families.

The question remains, however, is the GK form potentially more minimal than GRM and KRO forms.

Theorem 2. Every AND/EXOR canonical form f can be obtained from any other canonical form g by substituting terms of f by EXOR operation performed on some subsets of product terms from g.
Proof. It is well known that in a non-singular matrix every column can be replaced with a linear combination of other columns. In AND/EXOR canonical expansions the columns correspond to product terms and the linear operation is and EXOR of columns. Thus, by EXOR-ing the subsets of product terms we obtain all possible Linearly Independent forms. Because the set of Linearly Independent forms includes all forms with columns corresponding to products of variables, it includes the set of all canonical AND/EXOR forms. Thus, every AND/EXOR canonical form can be obtained from some other AND/EXOR canonical form by EXOR-ing some of its terms.

Example 5.4. By EXOR-ing all terms of $g(x, y, z)$ we obtain $\overline{y}\ \overline{z} = g2(x, y, z)$

Theorem 3. Every AND/EXOR form that can be described by the following formula:

$$f(a, b, c, ..., k; x_1, x_2, ..., x_m) = f_1(a, b, c, ...k)x_1 \oplus f_2(a, b, c, ...k)x_2 \oplus ... f_m(a, b, c, ...k)x_m \tag{5.1}$$

where:
- sets of variables $\{a, b, c, ..., k\}$ and $\{ x_1, x_2, ..., x_m\}$ are disjoint,
- functions f_i are linearly independent,
- all variables x_i are different.

has the minimum number of terms.
Proof. According to Theorem 2, any canonical AND/EXOR form can be obtained by replacing terms from the form $f(a, b, c, ..., k; x_1, x_2, ..., x_m)$ by EXOR-ing some subsets of its terms, for instance:

$f(a, b, c, ..., k; x_1, x_2, ..., x_m) = f_1(a, b, c, ...k)x_1 \oplus f_2(a, b, c, ...k)x_2 \oplus ... f_m(a, b, c, ...k)x_m2$
$= h_1(a, b, c, ..., k; x_1, x_2, ..., x_m) \oplus h_2(a, b, c, ..., k; x_1, x_2, ..., x_m) \oplus h_3(a, b, c, ..., k; x_1, x_2, ..., x_m) =$
$[f_1(a, b, c, ...k)x_1 \oplus f_3(a, b, c, ...k)x_3] \oplus [f_2(a, b, c, ...k)x_2 \oplus f_4(a, b, c, ...k)x_4] \oplus$
$[f_5(a, b, c, ...k)x_1 \oplus f_6(a, b, c, ...k) \oplus f_7(a, b, c, ...k)x_7]$.

However, none of the expressions in brackets can be simplified to less terms, because they have *different variables* x_i. In the best case, when all functions f_i are the same, the factorization of f_i can be performed, but it also does not decrease the number of terms. So, any EXORs of terms will not decrease the number of terms. Thus, the form $f(a, b, c, ..., k; x_1, x_2, ..., x_m)$ is the one with the minimum number of terms among all canonical forms representing this function.

Theorem 4. In every minimal ESOP all product terms are linearly independent.
Proof. Let assume that $z(x_1, x_2, ...x_r)$ is the minimal ESOP. Thus, it has the minimum number of terms among all ESOP expressions equivalent to $z(x_1, x_2, ...x_r)$. Assume that all terms from ESOP $z(x_1, x_2, ...x_r)$ are different and are not linearly independent. Thus some term can be replaced by an EXOR of some subset with more than one element of them, leading to new equivalent ESOP expression $z'(x_1, x_2, ...x_r)$. This new expression would have less terms, which contradicts the assumption that the expression $z(x_1, x_2, ...x_r)$ had the minimum number of product terms.

Definition 5. By **ESOP-simplifiable expression** in AND/EXOR form we will understand one that applying any combination of rules from EXORCISM-MV-2, [29], an ESOP with smaller number of terms were created. Expression that is not ESOP-simplifiable, will be called **ESOP-nonsimplifiable**.
Theorem 5. Every AND/EXOR expression of the following form:

$$f(a, b, c, ..., k; X_1, X_2, ..., X_m) = f_1(a, b, c, ...k)f_{h1}(X_1)\oplus$$
$$f_2(a, b, c, ...k)f_{h2}(X_2) \oplus ... \oplus f_m(a, b, c, ...k)f_{hm}(X_m) \tag{5.2}$$

where:

- set of variables $\{a, b, c, ..., k\}$ and any set of variables $X_1, X_2, ..., X_m$ are disjoint,
- functions f_i are linearly independent or they are EXOR combinations of linearly independent functions that are ESOP-nonsimplifiable,
- variables in formula (5.2) x_i are all different,
- all functions f_{hi} are on disjoint sets X_i of variables.

has the minimum number of terms.

Theorem 6. The canonical form of the form (5.2) is the minimal ESOP.

Theorem 7. The minimal GK Form is better than the minimal KRO and the minimal GRM forms.
Proof. From Theorems 4 and 5, the form $g(x, y, z, u, v, r, s, t) = \overline{x}\ \overline{y}\ \overline{z}u \ \oplus xyzv \oplus x\overline{y}r \oplus x\ \overline{z}\ s \oplus\ xt$ has a smaller term cost than all KRO and GRM canonical forms.

Thus, Theorem 7 demonstrates the usefulness of the concept of GK forms in practical minimization algorithms for high-performance canonical forms (because their expected superior testability) and ESOPs. The form shown in Theorem 7 is also the exact minimum ESOP.

All above notions and methods can be generalized to functions with many outputs.

Definition 6. A **Single-Polarity GRM expansion** for a multi-output function is a vector of GRM expansions for its component single-output functions, all of them of the same polarity.

Definition 7. A **Multi-Polarity GRM expansion** for a multi-output function is a vector of GRM expansions for its component single-output functions, each of them can have different polarity.

Thus, for a three-output function, the polarity vector of a Single-Polarity GRM expansion has 4 bits, and for Multi-Polarity LI expansion has 3 * 4 = 12 bits. Let us observe, that Definition 6 of multi-output GRM expansions is in accordance with the definition from [31], while Definition 7 is in accordance with the definition from [2]. Obviously, the minimal DD obtained from Definition 7 is smaller than one obtained from Definition 6. There are, however, some advantages of considering representations created according to Definition 6; mostly simpler and more efficient algorithms, and better testability.

Concluding, the set of canonical forms that are created by flattening the GKTs properly includes both the Kronecker forms and the GRM forms. This way we give a positive answer to the question posed several years ago by Tsutomu Sasao - *"whether there exist any family of canonical forms that is a superset of both KRO and GRM families"*. Of course, as it results from counting the number of the GK Forms, in addition to expansions from KRO and GRM families, there are very many other new canonical expansions resulting from the flattening of the GK trees.

VII. APPLICATIONS TO EXACT ESOP MINIMIZATION

ESOP Circuits are the AND/EXOR two-level circuits with no any constraints imposed on the products of literals - they are thus the best AND/EXOR circuits but they are not canonical. Much research has been devoted to the synthesis of minimal ESOP circuits [6, 8, 15, 20, 21, 22, 29], but so far the exact solution can be obtained only for a very small number of input variables [22]. The quality of AND/EXOR circuits obtained from the expansions proposed here should be significantly better than those corresponding to GRMs or KROs because the search space of GKTs is much larger than the combined search space of GRM or KRO expansions. Therefore, such circuits will be on average very good approximations of exact ESOPs.

It is easy to improve every ESOP minimizer by using Theorem 4. It is done by adding a final check to it. For every solution it has to be checked if the terms are linearly independent. If not, the exoring of the terms that are not linearly independent should be executed to obtain a better expression. Some top ESOP minimizers have already the rules that satisfy the condition that all terms are linearly independent.

Theorems 5 and 6 can be also used to create better rule-based searching ESOP minimizers (such as EXORCISM-MV-2, [29], and especially EXORCISM-MV-3, [30]). This can be achieved by tree-structuring the transformation rules that search the solution space, so that the space is represented by a tree. New nodes are created by applying rules to parent nodes, but the rules are selected in such a way that the costs are non-decreasing along the branches, which allows to apply the branch-and-bound principle for cutting off branches with higher costs than some of the previously found solutions. Special rules and methods are used to nodes of expressions like (5.1) or like (5.1) but with repeated literals x_i. An expression with repeated literals is transformed to expression (5.2) in which ESOP-nonsimplifiable expressions stand by x_i. (For instance, the expressions such as $(abc \oplus \overline{a}\,\overline{b}\,\overline{c})x_i$ are ESOP-nonsimplifiable, even they have the same factorized variable x_i.) Concluding, we can say that the class of generalized expressions of form (5.2) is the generalization of sparse functions introduced by us in previous papers on exact ESOP minimization [10]. Whenever a form like (5.2) is found, the branch of the tree is cutted-off, because no better form can be found by exoring subsets of terms. This way, the EXORCISM-3-MV search can be improved. More details on look-ahead search strategy of EXORCISM-3-MV are in [30].

It is well-known that set of ESOP expressions includes the set of all known and not known canonical expansions, but it was now known, if every minimal ESOP can be found from some canonical family of forms.

Theorem 8. For every minimum ESOP one can create a canonical form for which this ESOP is one of the solutions.

Proof. The form is created from the ESOP as follows. Assume the function of n variables. Let the minimal ESOP has k terms. Thus k columns of matrix M can be obtained from these terms.

The remaining $2^n - k$ columns are created one be one by adding columns that are EXOR linearly independent with all previous columns, such that the $2^n \times 2^n$ matrix M becomes nonsingular.

Theorem 9. The family of all canonical AND/EXOR forms includes the set of minimal ESOPs.

Theorem 9 can lead to a new ESOP minimization algorithm. Algorithms are known, [31] that enumerate all AND/EXOR forms of some type in certain order, for instance, the Grey order (this approach was used for FPRMs or GRMs [31]). The search is for the form with the smallest cost. If one were able to parametricize all canonical AND/EXOR forms, then, according to Theorem 9, the same approach would be possible to find the exact minimum ESOP.

VIII. CONCLUSIONS

In the paper, we proposed a new AND/EXOR representation. We have proved, that in general, this new representation will be better than all the known canonical AND/EXOR representations with respect to the circuit complexity. Based on the previous results in the field of AND/EXOR representations, we can expect that the new trees, diagrams, and expansions will find broad applications, especially in multi-level logic synthesis for FPGAs, gate arrays and standard cell technologies and in the design for high testability. For small number of variables, the introduced methods can be also used to design highly optimized ESOP circuits. We showed also some new theorems that will allow to create improved exact ESOP algorithms for small number of variables, and prove exactness for some special classes of ESOP expressions. Obviously, the future research should include counting the new forms and comparing them with the known forms.

Moreover, it is known from AND/OR minimization that the graph based representations allow to represent much larger functions than the two-level forms. The new graphs may be thus useful to represent large functions, larger than those possible to represent by the Ordered Kronecker DDs (which themselves allow to represent larger functions than the BDDs).

In this paper the difficulties of calculating the expansions were not addressed, and matrices were used for illustration. It should be stressed, however, that the discussed expansions can be executed on *any* representation of completely or incompletely specified single- or multi- output function, and we have already developed efficient algorithms that are not based on matrices [16].

Adaptation of the proposed approaches to multiple-valued, multiple-output functions, both completely and incompletely specified, as well as Boolean relations, is also of our interest, [16]. It will be also interesting to create a special algorithm to find exact minimum solutions for symmetric functions represented in the new forms [4].

Since AND/EXOR canonical representations have universal tests and have very good testability properties, and/or are easily modifiable to AND/EXOR circuits with such properties, it would be interesting to investigate how these properties can be best used for the introduced here GK representations.

REFERENCES

[1] M. Davio, J.P. Deschamps, A. Thayse, "Discrete and Switching Functions," *McGraw Hill*, 1978.

[2] D. Debnath, T. Sasao, "GRMIN: A Heuristic Simplification Algorithm for Generalized Reed-Muller Expressions," *Proc. RM'95*, pp. 257-264.

[3] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski, "Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams," *Proc. DAC'94*, pp. 415-419, 1994.

[4] R. Drechsler, "Pseudo Kronecker Expressions for Symmetric Functions," *Proc. VLSI Design Conference*, pp. 511-513, 1997.

[5] B. J. Falkowski, S. Rahardja, "Family of fast transforms for GF(2) orthogonal logic," *Proc. RM'95*, pp. 273-280.

[6] J. Froessl, B. Eschermann, "Module Generation for AND/XOR-Fields (XPLAs)," *Proc. ICCD '91,* pp. 26-29, 1991.

[7] D.H. Green, "Families of Reed-Muller Canonical Forms," *Intern. J. of Electr,* pp. 259-280, Febr. 1991, No 2.

[8] M. Helliwell, and M. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed Polarity Generalized Reed-Muller Forms," *Proc. DAC'88*, pp. 427- 432.

[9] Ch. Tsai and M. Marek-Sadowska, "Multilevel Logic Synthesis for Arithmetic Functions," *Proc. DAC'96,* pp. 242-247.

[10] M.A. Perkowski, and M. Chrzanowska-Jeske, "An Exact Algorithm to Minimize Mixed-Radix Exclusive Sums of Products for Incompletely Specifed Boolean Functions," *Proc. ISCAS'90,* pp. 1652-1655.

[11] M. Perkowski, L. Csanky, A. Sarabi, I. Schaefer, "Fast Minimization of Mixed-Polarity AND/XOR Canonical Networks," *Proc. ICCD'92,* pp. 32-36.

[12] M. Perkowski, "A Fundamental Theorem for Exor Circuits," *Proc. RM'93*, pp. 52-60.

[13] M. Perkowski, A.Sarabi, F. Beyl, "XOR Canonical Forms of Switching Functions," *Proc. RM'93*, pp. 27-32.

[14] M. Perkowski, A.Sarabi, F. Beyl, "Fundamental Theorems and Families of Forms for Binary and Multiple-Valued Linearly Independent Logic," *Proc. RM'95*, pp. 288-299.

[15] M. Perkowski, T. Ross, D. Gadd, J.A. Goldman, N. Song, "Application of ESOP Minimisation in Machine Learning and Knowledge Discovery," *Proc. RM'95*, pp. 102-109.

[16] M. Perkowski, L. Jóźwiak, R. Drechsler, "New Canonical Forms for Galois Logic and their Minimization." *to be submitted to ISMVL'98.*

[17] A. Sarabi, M.A. Perkowski, "Fast Exact and Quasi-Minimal Minimisation of Highly Testable Fixed-Polarity AND/XOR Canonical Networks," *Proc. DAC,'92*, pp. 30-35.

[18] A. Sarabi, M. Perkowski, "Design for Testability Properties of AND/EXOR Networks," *Proc. RM'93*, pp. 147-153.

[19] A. Sarabi, N. Song, M. Chrzanowska-Jeske, M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays," *Proc. DAC '94,* pp. 321 - 326.

[20] T. Sasao, "Optimisation of Multiple-Valued AND-EXOR Expressions using Multiple-Place Decision Diagrams," *Proc. ISMVL'92.*

[21] T. Sasao (editor), "Logic Synthesis and Optimisation," *Kluwer Academic Publishers,* 1993.

[22] T. Sasao, "An Exact Minimisation of AND-EXOR Expressions Using BDDs," *Proc. RM'93*, pp. 91-98.

[23] T. Sasao,"Representation of Logic Functions using EXOR Operators," *Proc. IFIP WG 10.5 Workshop on Applic. of the Reed Muller Expansion in Circuit Design, RM'95,* pp. 11-20.

[24] T.Sasao,"Representation of Logic Functions using EXOR Operators,"*Proc.RM'95*,pp.11- 20.

[25] T. Sasao, H. Hamachi, S. Wada, M. Matsuura, "Multi-Level Logic Synthesis Based on Pseudo-Kronecker Decision Diagrams and Local Transformation," *Proc. RM'95*, pp. 152-160.

[26] I. Schaefer, M. Perkowski, "Synthesis of Multi-Level Multiplexer Circuits for Incompletely Specified Multi-Output Boolean Functions with Mapping Multiplexer Based FPGAs," *IEEE Trans. on CAD,*, Vol. 12, No. 11, November 1993. pp. 1655-1664.

[27] I. Schaefer, M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms," *IEE Proc.,* Pt.E, Vol. 139, No. 6, pp. 519-527, November 1992.

[28] I. Schaefer, M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms," *Proc. ISMVL'91*, pp. 40-48.

[29] N. Song, M. Perkowski, "EXORCISM-MV-2: Minimisation of Exclusive Sum of Products Expressions for Multiple-Valued Input Incompletely Specified Functions," *Proc. ISMVL'93*, May 24, 1993, pp. 132-137.

[30] N. Song, M. Perkowski, "New Fast Approach to Approximate ESOP Minimization for Incompletely Specified Multi-Output Functions," *Proc. RM'97.*

[31] X. Zeng, M. Perkowski, K. Dill, A. Sarabi, "Approximate Minimization of Generalized Reed-Muller Forms," *Proc. RM'95,* pp. 221-230.