The Generalized Orthonormal Expansion of Functions With Multiple-Valued Inputs and Some of its Applications

Marek A. Perkowski
Department of Electrical Engineering
Portland State University
P.O. Box 751, Portland, Oregon 97207

Abstract

The paper introduces the fundamental concept of Generalized Orthonormal Expansion, which generalizes the ring forms of the Shannon expansion to the logic with multiple-valued (mv) inputs and standard trivial functions of arbitrary number of variables. Some applications of the Generalized Orthonormal Expansion are also presented, including several generalizations of canonical forms; both known from the literature and new. For instance, we define a family of canonical tree circuits. Such circuits can be considered for binary and multiple-valued input cases. They can be multi-level (trees and Directed Acyclic Graphs (DAGs)) or flattened to two-level AND-EXOR circuits.

1 Introduction

There are several reasons why there is recently an increased interest in logic synthesis using EXOR gates [1,12,30,48,52]. (Such circuits will be called *"Exor Circuits"*). These reasons include: **(1)** new technologies (Programmable Logic Devices (PLD): arithmetic [31], LHS501 [55]; Field Programmable Gate Arrays (FPGA) - Actel [21], Xilinx [57], CLi6000 [9]; and Standard Cells) either include EXOR gates, or allow to realize them in "universal logic modules". **(2)** Exor circuits can have smaller cost (usually calculated as number of terms) than inclusive (AND/OR) circuits [10,45,47,48]. **(3)** They are always very easily testable, much better than their inclusive counterparts [40,42,45]. The methods outlined in this paper are particularly suitable for the new FPGAs series CLi6000 for which no special design methods have been yet proposed. This "cellular logic" devices require regular connection patterns and provide AND/EXOR gates in basic blocks, which is an ideal match with AND/EXOR trees presented below.

The problem of finding the minimal generalized Reed-Muller (GRM) canonical form of optimal polarity [32] (called also fixed-polarity Reed-Muller [23]), as well as the problem of finding the minimal Exclusive-OR Sum of Products (ESOP) of a Boolean function [3,24,48,50], are the classical ones in logic synthesis theory. Recently efficient solutions have been proposed: to ESOPs in [2], and to GRMs in [15,45], but the problems are still far from being solved.

A book by Davio [11], and papers by Green [23] and Sasao [49] give information on the numbers and properties of various canonical forms being specializations of binary ESOPs, which may be useful to create efficient algorithms for them. In [37,38] we presented a family of *multiple-valued input expansions*. Here we will introduce new canonical binary and multiple-valued forms and expressions. Forms, Directed Acyclic Graphs (DAGs), Trees and expressions obtained by the tree searching methods introduced here will be all called *expansion circuits* or *expansions*, for short. Some of them will be canonical, while others will not. There are three main goals for the research reported here: **(1)** to create synthesis programs, exact and approximate, for all known and some new canonical forms being subsets of binary and multiple-valued input ESOPs **(2)** to create programs for their respective tree-like counterparts as well as tree-like circuits for non-canonical expressions. **(3)** to create programs for multi-level multi-output circuits being mixtures of EXOR, AND and OR gates with fan-in constraint, based on methods from (2) and to be applied for the new circuit technologies mentioned above.

Multiple-valued input ESOP (MIESOP) expressions (see Table 1) correspond to AND-EXOR PLAs with input decoders. It was stated in [48] that in most cases AND-EXOR PLAs with input decoders require fewer products than AND-OR PLAs with input decoders. Since efficient algorithms for general ESOPs still do not exist, the recent research is concentrated on efficient logic minimization algorithms for some special cases of ESOPs like: the GRMs [15,23,45], the Kronecker Reed-Muller (KRMs [20,23], and the Canonical Restricted Mixed Polarity (CRMP) Forms [7,8,10,11]. The Multiple-Valued Input, Generalized Reed-Muller (MIGRM) forms were introduced in [51,52,53]. The Kronecker and Pseudo-Kronecker forms are described in [11,23] and Quasi-Kronecker in [23]. The Multiple-Valued Input, Multiple-Valued Output Generalized Reed-Muller (MIOKRM) forms are in [25,28], and the 4-valued MIOESOP logic is described in [13] (more

[0]†This research was partially supported by the NSF grant MIP-9110772

0195-623X/92 $3.00 © 1992 IEEE

exactly, [25,28] and [13] discuss special cases of our MIOKRMs and MIOESOPs, respectively).

All those forms can be represented as modulo-n (EXOR in binary output cases) of terms, where each term is a constant, or a *minimum operation* (min(A,B) = if A ¡ B then A else B) of literals and a constant. (A minimum reduces to a *product operation* for the case of binary output). In p-valued logic the constants are 0,...,p-1. The literal $X_i^{S_i}$, where $S_i \in P_i$ is defined as follows: $X_i^{S_i} = 1$ if $X_i \in S_i$; and $X_i^{S_i} = 0$ otherwise. The GRM forms, which also include Reed-Muller (RM) form, are canonical. All literals in the RM form are positive, all literals in a Negative RM (NRM) form are negative. RM forms and NRM forms can be called Restricted GRM (RGRM) forms. In GRM forms all variables are in a fixed form, i.e. each occurrence of a variable in products of the form is either consistently positive or consistently negative. In CRMP forms [10,11] the literals have the same polarity for any subset of corresponding variables (for every subset of variables, if there are terms, they create a GRM form). For instance, $x_1\ x_2\ x_3 \oplus \bar{x_1}\ \bar{x_2}\ x_3 \oplus \bar{x_4}\ \bar{x_5}\ x_6$ is not a CRMP since for the subset of variables x_1, x_2, x_3 since for CRMP only one polarity of each of its literals is allowed while here variable x_1 occurs in two polarities: $x_1^0 = \bar{x_1}$, $x_1^1 = x_1$ (the same is the case about variable x_2). The CRMPs include "consistent" fixed-polarity GRMs and the "inconsistent" mixed-polarity forms introduced in [7]. In ESOP there is no restriction on literal polarities in terms.

The MIGRM forms have all literals with various but consistent polarities - this is a a multiple-valued input counterpart of GRMs. MIGRMs are shown to be equivalent to Multiple-Valued Input Kronecker Reed-Muller Forms (MIKRMs) in [38,54] and will be called MIKRMs from now. Correspondingly, MIRKRM [51,52,53], and MIORKRM "Restricted" forms can be considered.

The relations between some constrained forms of Multiple-Valued Input, Multiple-Valued Output ESOPs (MIOESOP) expressions introduced in the recent papers are illustrated in Table 1. We attempt to introduce here a comprehensive and unique terminology, since now various inconsistent names are used in the literature. For the multiple-valued input, binary output logic, and the multiple-valued input, multiple-valued output logic the notions of Generalized and Restricted forms and Unrestricted expressions are analogous to the binary logic. For example, a MICRMP (see Table 1) is a Multiple-Valued Input, Binary Output Canonical Restricted Mixed Polarity form, where for every subset of variables there can exist not more than one MIKRM form. This is a multiple-valued input counterpart of the CRMP.

A motivation for investigating the newly introduced forms, besides the general arguments listed earlier, is that they are canonical. Since the binary GRM forms are used for efficient coding of images [43], it is obvious that their multiple-valued input generalizations can give not worse results since, for instance, the number of MIKRMs is much larger than the number of the binary GRMs. It is thus more probable to find among them efficient codes for any given image. Finally, the excellent testability properties which were investigated basically for strict Reed-Muller forms [27,40,42,45] are also expansible to forms presented here.

By "flattening" we understand applying recursively the Boolean rule, $a(b \oplus c) = a\,b \oplus a\,c$. Flattening is used to convert trees and multi-level expressions to two-level expressions, such as Reed-Muller forms, or ESOPs. In this paper we will introduce Multi-Valued Input Orthonormal Trees (MIOT), and particularly the optimum Multi-Valued Input Kronecker Reed-Muller Trees (MIKRMT) and all their special cases [37]. After flattening, such circuits will produce several known and new generalizations of Generalized Reed-Muller Forms.

In section 2 the MIKRMs are introduced. Section 3 reviews Canonical Binary Forms. In section 4 we introduce the concept of the *Generalized Orthonormal Expansion*. The Shannon theorem has been generalized in [5] to arbitrary family of disjoint functions sum of which is equal to 1 (called Orthonormal Expansions), and by Sasao to the logic with multiple-valued inputs. Those expansions are of AND-OR type, to be used for inclusive and multiple-valued inclusive synthesis, respectively. On the other hand, three generalizations of Shannon expansion for Boolean rings (AND-EXOR type) are known [11,23]. A generalization of Shannon expansion for the logic with multiple-valued inputs, called the *Orthogonal Expansion* and of AND-EXOR type, was introduced in [38]. All these generalizations will be further generalized in this paper. The new generalization of this author, called Generalized Orthonormal Expansion, covers all expansions discussed in [37]; which allows for defining even larger families of canonical and non-canonical expansions than those from [38]. Some of those tree expansions are discussed in [37]. A special form of Generalized Orthonormal Expansion which is used to create canonical expansions with respect to single (binary and multi-valued) variable is also presented and used in section 5 to create families of expansion circuits corresponding to them. The circuits generated by the method shown there include the MIESOPS, MIKRMs, MIKRMTs and selected other expansion circuits from [37]. Also, some special cases of the algorithm from [37] which were only mentioned in [37] are now analyzed in more detail, and the new families of expansions corresponding to them are named.

2 MIKRM Forms

A multiple-valued input, two-valued output, completely specified switching function f (*multiple-valued function* , for short) is a mapping: $f(X_1, X_2, \ldots, X_n)$: $P_1 \times P_2 \times \ldots P_n \rightarrow \{0,1\}$, where X_i is a *multiple-valued variable,* and $P_i = \{0, 1, \ldots, p_i - 1\}$ is a *set of truth values* that this variable may assume. For any subset, $S_i \subseteq P_i$, $X_i^{S_i}$ is a *literal* of X_i. The set of values S_i will be called the *polarity of literal* $X_i^{S_i}$. A product of literals, $X_1^{S_1}\ X_2^{S_2} \ldots X_n^{S_n}$, is referred to as a *product term* (also called *term* or *product* for short). A (multi-

valued input) *sum-of-products expression* is denoted as a SOPE. Orthogonal matrix has all rows pairwise linearly independent (orthogonal) and is a ***basis of the vector space***, which means that each binary vector can be created by an EXOR of rows of this matrix. An *Expanded Polarity Matrix (EPM)* is a binary orthogonal matrix whose rows correspond to *allowed literals*. For instance for p=4 the selected two-input decoder type determines the following set of allowed literals: $\{X^{0,1,2}, X^{0,1,3}, X^{0,2,3}\}$, which is described by the *Polarity Matrix*:

$$PM(X) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \end{bmatrix} \quad (1)$$

When it is assumed that logic value **1** (universe) is available, **1** is treated as an allowed literal. This corresponds to a literal with all possible values, which in turn means a row of all ones in the EPM. Then, for the above example the expanded polarity matrix EPM(X) is:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \\ X^{0,1,2,3} \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \\ \mathbf{1} \end{bmatrix} \quad (2)$$

EPM can be also created without the row of **1**, for instance EPM for Shannon expansion is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Let us observe that all possible literals can be created by exoring rows of the EPM. For short, the Expanded Polarity Matrix of variable X_i is also called the *polarity* of this variable.

Any form in which all variables are in the same polarity is called a Multiple-Valued Input, Binary Output Restricted KRM (MIRKRM) form. Such a form is canonical since the expansion is unique for each of its variables. It can be shown that for a logic with 3-valued inputs there are 28 various polarities, and 28 MIRKRMs. The number of MIRKRM forms for a logic with p-valued inputs can be calculated from the known mathematical results on the number of orthogonal zero-one matrices. *Polarity vector* PV = $[PM_1, PM_2, ..., PM_n]$ is a vector of polarities of input variables $X_1, X_2, ..., X_n$. By a ***Multiple-Valued Input Kronecker Reed-Muller (MIKRM) Expression*** for the polarity vector PV one understands an exclusive-OR sum of products in which each literal $X_i^{S_j}$ is an allowed literal for the respective polarity PM_i from PV. It can be proven [53] that a MIKRM expression is canonical (which means that if for each variable a single polarity is selected, then there exists only one MIKRM expression for this set of variables and their corresponding polarities). Therefore from now on we will refer to a family of *MIKRM forms*.

3 Shannon Expansions for Rings and the Canonical Binary Forms

The well-known Shannon expansion for the case of ESOP expansion is as follows:

$$f = x_i \cdot f_{x_i} \oplus \overline{x_i} \cdot f_{\overline{x_i}} \quad (3)$$

$$f = f_{\overline{x_i}} \oplus x_i \cdot [f_{x_i} \oplus f_{\overline{x_i}}] \quad (4)$$

$$f = f_{x_i} \oplus \overline{x_i} \cdot [f_{x_i} \oplus f_{\overline{x_i}}] \quad (5)$$

where $f_{x_i} = f(x_1, ..., x_i = 1, ..., x_n)$ and $f_{\overline{x_i}} = f(x_1, ..., x_i = 0, ..., x_n)$. Let us observe that these expansion formulas have been applied by several authors for the synthesis of GRM forms for completely specified functions [11]. Davio [11] and Green [23] use them as a base of *Kronecker Reed-Muller (KRM)*, *Pseudo-Kronecker Reed-Muller (PKRM)*, and *Quasi-Kronecker Reed-Muller (QKRM)* forms (Green uses also trees for better explanation). If only rule 4 is used repeatedly for some fixed order of expansion variables, the RM Trees are created, which correspond to RM forms after their flattening. If for every variable one uses either rule 4 or rule 5, the GRM Trees are created, from which GRMs are obtained by flattening (which proves in other way why there is 2^n of such forms). If for every variable one uses either rule 3, rule 4, or rule 5, the KRM Trees are created, from which KRMs are obtained by flattening (which proves in other way why there is 3^n of such forms). If rules 3, 4 and 5 are used, but in each subtree there is a choice of a rule, the PKRM Trees are generated from which PKRM forms are obtained by flattening. Now, if additionally we allow the expansion variables to have various orders (but the same in the entire tree), one obtains the QKRM Trees, and QKRM flattened forms, respectively. One can now see that a further natural generalization is to allow various orders of variables in subtrees of QKRM trees to create an even wider family of trees [38,39]. When function f is incompletely specified and represented by sets ON and OFF [37], the above three equations are generalized to ones in which $f_{x_i} = [ON_{x_i}, OFF_{x_i}]$ and $f_{\overline{x_i}} = [ON_{\overline{x_i}}, OFF_{\overline{x_i}}]$.

4 The Generalized Orthonormal Expansion for Multiple-Valued Input Switching Functions

Below we will introduce several categories of expansions. For each category there are two basic types of Generalized Orthonormal Expansion: EXOR type and its dual EQUIVALENCE type. OR-type and AND-type expansions are their particular cases, respectively. Generalized Orthonormal Expansions include two main categories: *"Generalized Orthogonal Expansions"* and *"Generalized Orthonormal Expansions which are not Orthogonal"*. Below we will specify these two expansions in more detail.

Let $\mathbf{X} = X_1, X_2, ..., X_n$ be a set of mv input variables, and $\mathbf{X1} \subseteq \mathbf{X}$, $\mathbf{X1} \neq \Phi$. We will say that the set of (*standard trivial*) functions $stf_i(X_{i_1}, ..., X_{i_r})$, $X_{i_1}, ..., X_{i_r} \in \mathbf{X1}$, is an *orthogonal set of functions with respect to set* **X1**

when the matrix EM_{stf_i}, called *Expansion Matrix*, is orthogonal with respect to expansion operator on rows (below the expansion operators are EXOR, EQUIVALENCE, OR, and AND). Matrix EM_{stf_i} is created as follows. Its columns correspond to minterms $m_j(X_{i_1}, ..., X_{i_r})$ of variables from **X1**. Its rows correspond to set of functions $stf_i(X_{i_1}, ..., X_{i_r})$. $EM_{stf_i}[stf_i(X_{i_1}, ..., X_{i_r}),\ m_j(X_{i_1}, ..., X_{i_r})] = 1$ **iff** $stf_i(X_{i_1}, ..., X_{i_r}) \supseteq m_j(X_{i_1}, ..., X_{i_r})$. Rows of EM_{stf_i} are called *"standard trivial functions"* and the set of standard trivial functions is denoted by STF. $STF_{\oplus}$ denotes set of standard trivial functions from matrix orthogonal with respect to EXOR operator. Matrix is orthogonal when every single minterm can be created in an unique way from functions specified by rows of the matrix. In case of EXOR and EQUIVALENCE operators this means a matrix orthogonal with respect to respective bit-wise operators on rows. (For instance matrix EPM(X) from (2) is orthogonal with respect to EXOR). In case of an OR operator this means rows corresponding to single minterms $m_j(X_{i_1}, ..., X_{i_r})$. In case of an AND operator this means rows corresponding to single maxterms $M_j(X_{i_1}, ..., X_{i_r})$.

(EXP 1). The *Generalized Orthogonal Expansion for Orthogonal Set of Functions* STF is described by the formula:

$$f = \bigoplus_{stf_i \in STF_{\oplus}} (f_i \cdot stf_i) \qquad (6)$$

We will call the above expansion the *EXOR type expansion*, since the expansion is with respect to EXOR gate. (Matrix EM_{stf_i} has rows orthogonal with respect to operator EXOR).

One can observe that the *EQUIVALENCE type expansion* can be created that is dual to formula 6.

$$f = \bigodot_{stf_i \in STF_{\odot}} (f_i + stf_i) \qquad (7)$$

We call the above expansion the *EQUIVALENCE type expansion*, since the expansion is with respect to EQUIVALENCE gate. (Matrix EM_{stf_i} has rows orthogonal with respect to operator EQUIVALENCE).

The particular case of formula 6 is created when all functions $stf_i \in$ STF are minterms $m_j(X_{i_1}, ..., X_{i_r})$.

$$f = \bigcup_{stf_i \in STF_{\cup}} (f_i \cdot stf_i) \qquad (8)$$

We will call 8 the *OR type expansion*, since the expansion is with respect to OR gate, and $\bigcup$ replaces $\bigoplus$ in formula 6, producing formula 8.

The particular case of formula 7, dual to 8, is formula 9:

$$f = \bigcap_{stf_i \in STF_{\cap}} (f_i + stf_i) \qquad (9)$$

where all functions $stf_i \in$ STF are maxterms $M_j(X_{i_1}, ..., X_{i_r})$. We will call 9 the *AND type expansion*, since the expansion is with respect to AND gate.

The expansions of category EXP1 totally eliminate the set of variables **X1** from the function $f(X_1, ..., X_n)$. They will be called *Expansions that Totally Eliminate Variables*. The problem of determining functions f_i in this and other orthonormal expansions is a subject of a forthcoming paper.

(EXP2). Particular case of the expansion from (EXP1) is when it is applied to set **X1** of a single variable. This case is called a *Single-Variable, Totally Eliminating Variable Expansion* and allows to derive simple formula for functions f_j in the expansion. When the (EXP2) EXOR-type expansion is applied to function f with respect to multiple-valued input variable X_i of polarity $EPM(X_i)$, the following can be derived:

$$f = \bigoplus_{X_i{}^{S_j} \in EPM(X_i)} f_{X_i{}^{S_j}} \ X_i{}^{S_j} \qquad (10)$$

where the values of $f_{X_i{}^{S_j}}$ are calculated as follows: $[f_{X_i{}^{S_j}}]^T = [f_{X_{i,j}}]^T\ [NP]^{-1}$; $[f_{X_i{}^{S_j}}]$ is a vector of single-literal orthogonal expansions of literal $X_i{}^{S_j}$, $j = 0, ..., p-1$; $[f_{X_{i,j}}]$ is a vector of single-literal standard expansions of single-value literal X_i, $(X_i = j)$, $j = 0, ..., p-1$; $[A]^T$ means matrix $[A]$ transpose; $[A]^{-1}$ means matrix $[A]$ inverse; $[NP]$ is a normalized polarity matrix, which relates polarities of multiple-valued input literals to single-value literals. In this case, $X_i{}^{S_j}$ are the standard trivial functions stf_j, and functions $f_{X_i{}^{S_j}}$ correspond to f_i from formula 6.

(EXP3). Another particular case of (EXP1) is when **X1** = **X**. This leads to generalizations of canonical MIKRM, Multiple-valued Input Sum of Products (MISOP) and Multiple-valued Input Product of Sums (MIPOS) forms to more general types of expansions. Finding f_j is done by generalizing the matrix methods from [54].

(EXP4). The case of (EXP1) when **X1** $\neq$ **X** and card(**X1**) $\neq 1$ is a new type of generalization, which leads to multi-level layered decomposition, in which each layer corresponds to set **X1** of variables (this is a generalization of a tree in which a layer corresponds to a variable). This is the most general category of Generalized Orthogonal Expansions discussed here. It is also called *Expansion that Totally Eliminates Variables.*

(EXP5). A more general case than category (EXP1) are the *Expansions that Do Not Totally Eliminate Variables.* They are Orthonormal expansions that are not Orthogonal.

By a *"normal" (universe creating)* set of functions for a type FUNCTOR of expansion we will call the set of functions which allows to create the value 1 for FUNCTOR $\in$ {EXOR, OR} and the value 0 for FUNCTOR $\in$ {EQUIVALENCE, AND}, by applying FUNCTOR operator to them, or to their subset. For instance when FUNCTOR = EXOR the functions de-

scribed by the matrix

$$EPM(X_i) = \begin{bmatrix} X^{0,2} \\ X^{0,1} \\ X^{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$

are normal since $110 \oplus 001 = 111 = 1$. Analogously, functions $a + b$, $\bar{a}\,\bar{b}$ are normal for FUNCTOR = OR since $(a+b) + (\bar{a}\,\bar{b}) = 1$. From practical point of view it is reasonable to assume that constant 1 is available. If we assume 1 available, for EXOR type of expansion any subset of rows of orthogonal matrix EM_{stf_i} becomes a normal set of functions. Similarly for 0 in the EQUIVALENCE type expansion.

Matrix EM_{stf_i} in expansions 6, 7, 8, and 9 can be created for any set of functions stf_i on variables from set **X1** such that:

1. functions stf_i are orthogonal (linearly independent),
2. functions stf_i are normal.

The set of functions stf_i that is both orthogonal and normal will be called the *orthonormal set of functions.* The expansion for the orthonormal set of functions, the *Generalized Orthonormal Expansion*, is the most general expansion discussed here. The difference with the expansion from (EXP1) is that the single expansion does not eliminate totally the variables from **X1**, so that another expansions must be next used in the tree for those variables, but using their different polarities. The expansions are carried out until all pairs of true and false minterms of function $f(X_1, ..., X_n)$ are separated. For instance, in the case of OR expansion, each product of all standard trivial functions selected in a branch of a tree should not include 0's and 1's of the function (but can include 0's and -'s, or 1's and -'s). Examples of such expansions are given in [37]. Expansion (EXP5) becomes additionally the "Expansion that Totally Eliminates Variables" from (EXP1) when matrix EM_{stf_i} becomes orthogonal (Keep here in mind that matrix of orthonormal functions can be not orthogonal). For EXOR type expansion the matrix of orthonormal expansion is created by taking any normal subset of rows from an orthogonal matrix EM_{stf_i}. For OR type expansion the stf_i functions must be disjoint and add to **1**, i.e.:

$$stf_i \cap stf_j = 0 \; when \; i \neq j, \; and \bigcup_{stf_i \in STF} stf_i = 1 \quad (12)$$

For AND type expansion the stf_i functions must add pairwise to 1 and intersect to 0:

$$stf_i \cup stf_j = 1 \; when \; i \neq j, \; and \bigcap_{stf_i \in STF} stf_i = 0 \quad (13)$$

Remark 1. Since conditions 12 and 13 are sufficient for set of functions STF to be used in OR type orthonormal expansion, when functions stf_i are disjoint an "inhibiting gate" can be used to create the remaining function to make a set of normal functions. For instance, for disjoint functions f_1, f_2, f_3 one can create inexpensively in hardware the function $f_4 = \overline{f_1 + f_2 + f_3}$ to create the orthonormal set of standard trivial functions STF = $\{f_1, f_2, f_3, f_4\}$. Similarly for AND type expansion.

Remark 2. Let us observe that when the standard trivial functions are EXORs of variables, they are the same as the standard trivial functions used in Walsh expansions [17,18]. Expansions such as

$$f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2)\, f_{x_1 \oplus x_2} + \overline{(x_1 \oplus x_2)}\, f_{\overline{x_1 \oplus x_2}} \quad (14)$$

can be created. A set of standard trivial functions of any spectral transform [15,16,17,18] can be used this way. Spectral coefficients give correlation of function f to standard trivial functions and can be thus used to find good polarities for expansion.

More detailed analysis of applications of the Generalized Orthonormal Expansion is included in our forthcoming paper. Here we will briefly reiterate some key issues.

1. The Generalized Orthonormal Expansion applied in various restricted ways to a multiple-valued input function creates a family of canonical tree expansions. These expansions are called *Orthonormal Trees* and are analogous to those for binary [11] and multiple-valued input [38] logic, but are much more general. The multiple-valued input function to be manipulated by those expansions (whether completely or incompletely specified) is represented in a disjoint or ESOP representation; as arrays of disjoint cubes; Ordered Multiple-valued Decision Diagrams (OMDDs), ESOP array of cubes, or other disjoint or ESOP representation).

2. Applying the expansion in a tree uniformly for a fixed order of expansion variables of the same polarity one obtains the *Single Polarity Orthonormal Trees (SPO)* that are the generalizations of the binary and multiple-valued input Single Polarity Reed-Muller Trees (such as MIRKRM Trees [38,39]).

3. Applying the expansion in a tree uniformly for a fixed order of expansion variables of various polarities one obtains trees that are generalizations of the *Multiple-Valued Kronecker Reed-Muller Trees (MIKRM Trees).* We will call them *Multiple Polarity Consistent Orthonormal Trees.*

4. Applying the expansion in a tree for a fixed order of expansion variables, but selecting various variable polarities in different sub-expressions (sub-trees) one obtains the generalizations of the *Multiple-Valued Pseudo-Kronecker Reed-Muller Trees (MIPKRM Trees).* We will call them *Multiple Polarity Inconsistent Orthonormal Trees.*

5. Applying the expansion in a tree for all possible but fixed orders of expansion variables, and

selecting various variable polarities in different sub-expressions (sub-trees) one obtains generalizations of the *Multiple-Valued Quasi-Kronecker Reed-Muller Trees (MIQKRM Trees)*. We will call them *Multiple Polarity Multiple Order Orthonormal Trees.*

6. Applying the expansion in a tree for all possible orders of expansion variables, selecting various orders in various sub-trees, and selecting various variable polarities in different sub-expressions (sub-trees) one obtains generalizations of the above expansions. These new expansions are also generalizations of circuit expansion families presented in the next section.

7. The expansion algorithm can be applied with little modification to multi-output functions: it is applied to a vector of single-output functions, step-by-step for each component function separately. The logically equivalent sub-trees can be combined, which leads to DAG circuits (This transformation preserves the canonicity of the tree circuits). It is also possible to combine only some common sub-trees to preserve planar netlist of the forest of trees for better placement to FPGAs.

8. The trees from all the above new families of canonical trees can be flattened to respective canonical mv forms. This leads to MIRKRM forms, MIKRM forms, MIPKRM forms, MIQKRM forms, and new mv canonical forms.

9. Several other families of Orthonormal Trees can be created by restricting types of expansions (i.e. applying in the node of the search tree not all possible expansions, but only the expansions of certain type, such as EXOR type expansion, OR, AND, EQUIVALENCE, or other). Other families can be created by selecting various numbers of variables in one expansion. Yet another families can be obtained by considering separately the variable eliminating expansions (Orthogonal Expansions) and several special subcategories of Orthonormal Expansions being not Orthogonal.

10. Orthonormal (and also non-orthonormal) expansions can be constructed for arbitrary sets of mv functions, possibly modifying first the functions to make them normal and/or linearly independent. This leads to new general decomposition methods for standard library modules of certain type.

11. The above four expansion types can be comprehensively generalized to arbitrary Symmetric Function Type Expansions and next to Arbitrary Function Type Expansions.

12. The expansions can be also generalized by replacing standard trivial functions in formulas with some functions of them, for instance (6) is generalized to: $f = \oplus_{stf_i \in STF_1,\ stf_j \in STF_2} (f_{i,j} \cdot stf_i \cdot stf_j)$.

13. All the above expansions can be also generalized to incompletely specified functions [37].

5 The Family of Single Variable EXOR Type Expansions for Multiple Valued Input Logic

Our tree searching algorithm for tree expansion operates on functions represented as [ON, OFF] disjoint cube arrays. It applies recursively the expansion formulas described above so that one obtains all tree expansions of a certain category and type. Each solution obtained by the algorithm is a multi-level circuit, tree or DAG, which can be directly mapped to the above mentioned FPGA technologies. Also, the tree can be next flattened to expressions to be realized in AND/EXOR PLAs. Below we will describe only the trees for single-variable EXOR expansions from section 4.

The search uses several heuristics for selection of variables and expansion types in every node of the expanded part of the search tree. For instance, for the MIESOP of disjoint cubes, let us denote by $FR(X_i^{S_j})$ the number of cubes with literal $X_i^{S_j}$. The variable X_i is selected for expansion which has the highest value of $FR(X_i^{S_j})$ among all values of i and j. The selected type of orthogonal expansion is executed for this variable for expanded polarity matrix composed of the most frequent literals from the cubes which include this variable. (A specialization of this method for binary GRMs is the well-known heuristics of selecting the polarity of variable which occurs most among the cubes [6,37,45]). Other heuristics are discussed in more detail in [37].

The selected expansion for the selected variable is applied either to the *entire* expression, or to its $f_{X_i^s}$ part created by the former expansion. Therefore, all those expansions are deterministic. DAGs can be created by factorization of common sub-trees in the trees, which is a unique process, also for multi-output functions. Instead of factorizing, the identical sub-functions are recognized by tautology during the expansion process, which technique is also usable for incompletely specifed functions.

Table 2 presents the characterization of all *"Deterministic, Non-partitioned Tree Expansions"* and corresponding flattened forms. The code of the tree expansion (column 5), which totally specifies the type of expansion is created by concatenating the codes of partial expansion types parameters from the headings of the first four columns: *entire/part* denotes expanding entire expression in nodes (E), or partitioning the expression first (P) (not used in Table 2, used for non-deterministic and partitioned algorithms); *order of variables* can be S, which means that a single and predetermined before search order of variables is used in all expansions; V, which means that all possible orders are investigated, but they are global in the

entire tree; order D means dynamically selected variables in each sub-tree independently. *Relation of order of subtrees* can be G, or F, G means the same expansion on each level of a tree, F means flexible choice of expansion type in a subtree. Expansion types are 1,2, and 3, corresponding to expansions (3), (4), and (5), respectively. The sixth column gives the name of the respective flattened form, if known.

The *"Partitioned" Tree Expansions* are similar to the one outlined above, but the expression in each node is first partitioned to groups, and the expansions are done in those groups independently. There are several rules to create these partitions. For instance, CRMPs can be created from any ESOP by first using (3) for variable X_i expansion, and next each subfunction (the $f_{X_i s_j}$ part) is partitioned *separately*. It is partitioned in a non-deterministic way (which means a non-deterministic choice, or all choices in the implementation). Another method to create CRMPs starts from any CRMP (or practically a GRM), which is next non-deterministically partitioned to subtrees which are independently expanded. This kind of "partitioned", non-deterministic methods create a family similar to the one from Table 2. A Table 3 (not shown), analogous to Table 2, is created for partitioned ESOPs, and particularly the partitioned CRMPs, GRMs, and RMs. The difference is that the first column is described by letter P, which stands for "partitioned". So the letter-codes of all expansion types from this table will have first letter P, instead of E in Table 2. Table 3 includes all CRMPs and many other expansions, some of them are forms, but their counting and analysis is difficult. Finally, two tables, the equivalents of Table 2 and Table 3, were created for functions with multiple-valued inputs. These four tables systematize all the expansions outlined in this paper. The expansions are now under investigation.

Green [23] presents all forms of two binary variables. Additionally, we developed binary and ternary forms for two and three variables. Those forms show interesting structure and mutual relations. For instance, sub-families of CRMPs for binary functions of two variables are shown in Fig. 1. Arrows denoted by N describe the relation of negation of variables. Arrows denoted by C describe the relation of changing the order of variables.

In our opinion, the CRMP class is a very interesting one for practical considerations, since on one hand it generates a variety of solutions (it is not included even in the extremely large QKRM family), and on the other hand it is not that large as the QKRM (for instance, for two-variable functions there are 16 CRMPs and 45 QKRMs). This makes it good for symmetric functions [10].

6 Conclusion.

In this paper the Generalized Orthonormal and Orthogonal Expansions have been introduced for the first time. They generalize several extensions to Shannon theorem including: (1) binary orthonormal expansions of Loewenheim [5], (2) binary ring expansions of Davio [11], (3) Sasao's multiple-valued inclusive expansion [46,44], (4) Perkowski's/Johnson's Orthogonal Expansion [38].

Since the Shannon theorem and all above expansions have several important applications in tautology, complementation, implicants generation, decomposition, Binary Decision Diagrams (BDDs), synthesis with multiplexers, multi-level synthesis, EXOR circuits synthesis, study of canonical forms, and many other fundamental problems of logic synthesis, we expect the Generalized Orthonormal Expansion Theorem to play also a fundamental role in the multiple-valued input logic.

Several well-known canonical forms have been also generalized for the logic with multiple-valued inputs. Several new expansions were formulated and generalized. The reader must bear in mind that the expansions proposed here relate to trees and not "flat" forms. For instance, the GRM forms are independent on the order of variables, but the respective GRM trees do depend on this order. Since several expansions obtained by changing the order of variables produce the same "flat" form, counting of several forms can be difficult, as already observed for Quasi-Kronecker forms by Green [23]. It is even more so for our forms, where different orders of variables in subtrees are possible.

The methods shown here open a wide area of interesting and new applications, especially to new FPGAs, in which high regularity of connections is more important than the number of logic blocks, which requirement favours trees and DAGs. Contrary to most papers from the literature, our expansion algorithms based on the above methods use cubes, not minterms [37,39]. They can be easily modified for OMDD representation which has been done for a particular case of MIKRMs in [54]. All those problems can be also expressed in the language of matrix operations and spectral theory [1,11,23] and the first results can be found in [54].

Similarly as the Shannon expansion is a base of the Binary Decision Diagrams (BDDs), which were recently found to be a very efficient representation for Boolean functions manipulation [56], the Generalized Orthonormal Expansion is the base of the binary and multiple-valued *"Orthonormal Decision Diagrams (ODDs)"* which exhibit similiar properties but have usually less nodes. The ODDs are a starting point to the multi-level synthesis and technology mapping for CLi6000 and other FPGA devices [39].

References

[1] Ph.V. Besslich, *Proc. IEE*, Vol. 130, Part E, CDT, No. 6., pp. 203-206, 1983. [2] Ph.V. Besslich, *Proc. Euro ASIC'91*, Paris, 1991. [3] D. Brand, and T. Sasao, *Proc. of 23rd FTC*, pp. 1 - 9, 1990. [4] R.K. Brayton, et al, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer, 1984. [5] F.M. Brown, "Boolean Reasoning. The Logic of Boolean Equations", *Kluwer*, 1990. [6] A. Chan, *IEEE Trans. Comp.*, Vol. C-36, No. 2., Febr. 1987, pp. 212-214. [7]M. Cohn, "Switching Function Canonical Forms over Integer Fields", *Ph.D. Dissertation*, Harvard University, Cambridge, MA, Dec. 1960. [8]

M. Cohn, *IRE Trans. Electron. Comput.*, Vol. EC-11, p. 284, April 1962. [9] Concurrent Logic Inc., "CLI6000 Series Field Programmable Gate Arrays", *Preliminary Information*, Dec. 1 1991, Rev. 1.3. [10] L. Csanky, M. Perkowski, and I. Schaefer, "Canonical restricted mixed-polarity exclusive sum of products and the efficient algorithm for their minimization", *Proc. of ISCAS'92.* [11] P. Davio, et al, *Discrete and Switching Functions.* George and McGraw-Hill, New York, 1978. [12] E. Detjens, "FPGA Devices Require FPGA-specific Synthesis Tools", *Computer Design*, p. 124, Nov 1990. [13] G. Dueck, and D. Miller, *Proc. of 14th ISMVL*, pp. 232-240, 1986. [14] B.J. Falkowski, and M.A. Perkowski, *Int. J. of Electr.*, Vol. 70, No. 3, pp. 533-538, March 1991. [15] B.J. Falkowski, and M.A. Perkowski, *Int. J. of Electr.*, Vol. 71, No. 3, pp. 383-396, Sept. 1991. [16] B.J. Falkowski, and M.A. Perkowski, *Proc: of ISCAS'90*, pp. 2913-2916. [17] B.J. Falkowski, I. Schaefer, and M.A. Perkowski, "Effective Computer Methods for the Calculation of Hadamard-Walsh Spectrum for Completely and Incompletely Specified Boolean Functions," *Accepted to IEEE Trans. on CAD.*, 1991. [18] B.J. Falkowski, and M.A. Perkowski, *Proc. of 20th ISMVL*, pp. 75-82, May 1990. [19] H. Fujiwara, *Logic Testing and Design for Testability*, Computer System Series, The MIT Press, 1986. [2] [20] P. Gilliam, "A Practical Parallel Algorithm for the Minimization of Kronecker Reed-Muller Expansions", M.S. Thesis, Portland State Univ., August 1991. [21] GOTHIC CRELLON, "The Beginners Guide to Programmable ASICs", 1990. [22] D. Green, *Modern Logic Design*, Electronic Systems Engineering Series, 1986. [23] D. Green, *Int. J. Electr.*, Vol. 70, No. 2, pp. 259-280, Jan. 91. [24] M. Helliwell, and M.A. Perkowski, *Proc. of 25th DAC* 1988, pp. 427 - 432. [25] Z. Hu, *Int. J. Electr.* Vol. 63, No. 6, pp. 851-856, June 1987. [26] S.L. Hurst, *The Logical Processing of Digital Signals, Crane-Russak,* New York and Edward Arnold, London, 1978. [27] K.L. Kodandapani, *IEEE Trans. Comp.*, Vol. C-23, pp. 332-333, 1974. [28] K.L. Kodandapani, R.V. Setlur, *IEEE Trans. on Comput.*, pp. 628-636, June 1975. [29] K.L. Kodandapani, and R.V. Setlur, *IEEE Trans. Comp.*, Vol. C-26, pp. 310-313, 1977. [30] H. Landmann, "Logic Synthesis at Sun", *IEEE conference paper*, CH 2686 - 4 / 89 / 0000 / 0469, 1989. [31] MONOLITIC MEMORIES, INC., "XOR PLDs Simplify Design of Counters and Other Devices", *EDN*, May 28, 1987. [32] A. Mukhophadhyay, and G. Schmitz, *IEEE Trans. Comp.*, Vol. C-19, No. 2., pp. 132-140, February 1970. [33] D.E. Muller, *IRE Trans. Electron. Comp.*, Vol EC-3, pp. 6-12, September 1954. [34] G. Papakonstantinou, *IEEE Trans. on Computers*, Vol. C-28, pp. 163-167, February 1979. [35] M.A. Perkowski, and M. Chrzanowska-Jeske, *Proc. of ISCAS*, pp. 1652-1655, May 1990. [36] M.A. Perkowski, M. Helliwell, and P. Wu, *Proc. 19 ISMVL*, Guangzhou, PRC, May 1989, pp. 256-263. [37] M.A. Perkowski, P. Dysko, and B.J. Falkowski, *Proc. IEEE Int. Phoenix Conf. on Comp. and Comm.*, Scottsdale, Arizona, pp. 606-613, March 1990. [38] M.A. Perkowski, and P. Johnson, *Proc. 3-rd NASA Symposium on VLSI Design*, pp. 11.3.1-11.3.13. Moscow, Idaho, October 30-31, 1991. [39] M.A. Perkowski, and L-F. Wu, "A Program to Find Quasi-minimum Canonical Reed-Muller Trees", *PSU EE Report*, Portland, OR, February 1992. [40] D.K. Pradhan, *Fault-Tolerant Computing. Theory and Techniques. Vol. I.*, Prentice-Hall, 1987. [41] I.S. Reed, *IRE Trans. Inf. Th.*, Vol. PGIT-4, pp. 38-49, 1954. [42] S.M. Reddy, *IEEE Trans. Comput.*, Vol. C-21, pp. 1183 - 1188, 1972. [43] B.R.K. Reddy, and A.L. Pai, *Comp. Vision, Graph., and Image Proc.*, Vol. 42, pp. 48 - 61, 1988. [44] R. Rudell, "Multiple-Valued Logic Minimization for PLA Synthesis", M.S, *University of California*, Berkeley, June 1986. [45] A. Sarabi, and M.A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/EXOR Canonical Networks", *Proc. DAC'92*, June 1992. [46] T. Sasao, and H. Terada, *Proc. of 9th ISMVL*, Bath, England, pp. 27 - 37, 1979. [47] T. Sasao, and P. Besslich, *Inst. of Electr. and Comm. Eng. of Japan*, FTS86-17, pp. 1-8, Nov. 17, 1986. [48] T. Sasao, *Proc. of 20th ISMVL*, pp. 128-135, May 1990. [49] T. Sasao, *FTS* 91-35, pp. 29-36, 1991. [50] J.M. Saul, *Proc. ICCD'90*, pp. 372-375, Sept. 1990. [51] I. Schaefer, "An Effective Cube Comparison Method for Discrete Spectral Transformations of Logic Functions", *M. Sc., Thesis*, May 1990. [52] I. Schaefer, and M.A. Perkowski, *Proc. of the ISMVL-91*, Victoria, B.C., Canada, May 1991. [53] I. Schaefer, and M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms", *submitted to IEE Journal*, May 1991. [54] I. Schaefer, and M.A. Perkowski, "An Algorithm to Find the Minimal Multiple-Valued Input Kronecker Reed-Muller Form", *PSU report, submitted to IEEE Tr. on Computers.* [55] SIGNETICS, PLD Data Manual, Signetics' Approach to Logic Flexibility for the '80's", 1986. [56] A. Srinivasan, T. Kam, S. Malik, and R.K. Brayton, *IEEE Proc. of ICCAD'90*, pp. 92-95, 1990. [57] XILINX, Inc., *"The Programmable Gate Array Data Book"*, 1989.

binary input, binary output	*multiple-valued input, binary output*	*multiple-valued input, multiple-valued output*
RGRM [41, 33, 23]	MIRKRM [51, 52]	MIORKRM
GRM [23] Kronecker (KRM) [11, 23]	MIKRM [51, 53]	MIOKRM [25, 28]
Pseudo-Kronecker (PKRM) [11, 23]	MIPKRM (this paper)	MIOPKRM
Quasi-Kronecker (QKRM) [11, 23]	MIQKRM (this paper)	MIOQKRM
CRMP [7, 8, 10]	MICRMP (this paper)	MIOCRMP
ESOP [24, 50]	MIESOP [36, 48]	MIOESOP [13]

Table 1.

TABLE 2

entire/part	order of variables	relation of order of subtrees	type of expansion	code of tree expansion	name of flattened form
E	predetermined fixed order of variables S	global expansions in subtrees G	1.	ESG1	1. Reed-Muller
			2.	ESG2	2. Negative Reed-Muller
			3.	ESG3	3. canon. minterm ESOP
			1+2.	ESG12	4. fixed polarity GRM
			1+3.	ESG13	5. new form
			2+3.	ESG23	6. new form
			1+2+3.	ESG123	7. Kronecker
		flexible choice of expansion type in each subtree F	1.	ESF1	8. same as 1
			2.	ESF2	9. same as 2
			3.	ESF3	10. same as 3
			1+2.	ESF12	11. new form
			1+3.	ESF13	12. new form
			1+2+3.	ESF123	13. pseudo-Kronecker
	variable order of variables V	global expansions in subtrees G	1.	EVG1	14. same as 1
			2.	EVG2	15. same as 2
			3.	EVG3	16. same as 3
			1+2.	EVG12	17. new form
			1+3.	EVG13	18. new form
			2+3.	EVG23	19. new form
			1+2+3.	EVG123	20. Quasi-Kronecker
		flexible choice of expansion type in each subtree F	1.	EVF1	21. same as 1
			2.	EVF2	22. same as 2
			3.	EVF3	23. same as 3
			1+2.	EVF12	24. new form
			1+3.	EVF13	25. new form
			2+3.	EVF23	26. new form
			1+2+3.	EVF123	27. new form
	orders different in subtrees D	G	1,2,3,1+2, 1+3.2+3, 1+2+3	EDG1 to EDG123	28-34, analogously as 21-27
		F	1,2,3,1+2, 1+3.2+3, 1+2+3	EDF1 to EDF123	35-41, analogously as 21-27

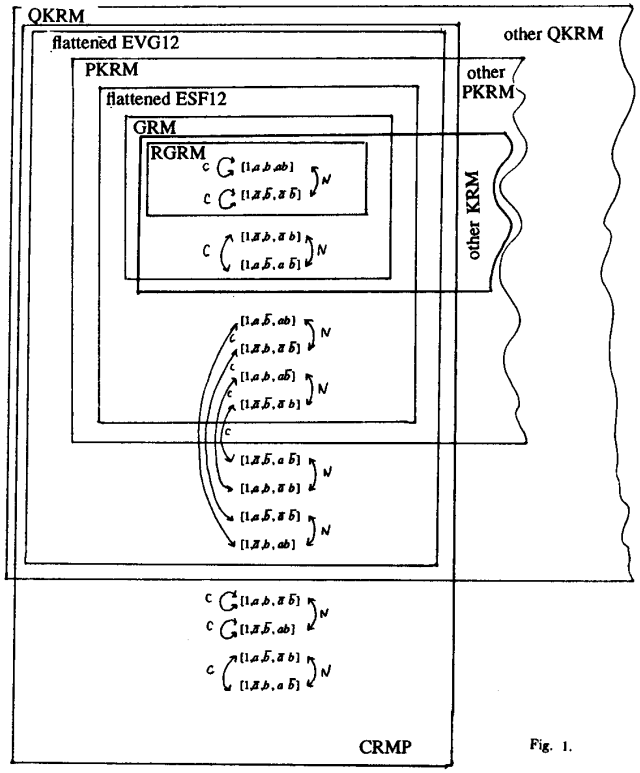

Fig. 1.