Fundamental Theorems and Families of Forms for Binary and Multiple-Valued Linearly Independent Logic

Marek A. Perkowski
Dept. of Elect. Engn.
Portland State Univ.
Portland, OR 97207-0751
mperkows@ee.pdx.edu

Andisheh Sarabi
Viewlogic Systems, Inc.
47211 Lakeview Blvd.,
Fremont, CA 94538
andisheh@wcdf.viewlogic.com

F. Rudolf Beyl
Dept. of Math. Sci.
Portland State Univ.
Portland, OR, 97207-0751
rudy@math.pdx.edu

Abstract—
Paper introduces Nonsingular Forms (NSF) for binary and multiple-valued functions. Fundamental theorems of Linearly Independent Logic are given, as well as circuit interpretation and methods of minimization. While the UXFs introduced by us earlier provide all possible XOR canonical representations of Boolean functions, NSFs provide extensions of UXFs to the multiple-valued case. Two types of NSFs will be described in more detail: "polynomial" and "Min/Max". Polynomial forms are generalizations of UXFs to fields larger than GF(2). The Min/Max is the Post Algebraic type generalization.

I. Introduction

In previous papers [9, 10, 13, 12], we introduced the concepts of Orthogonal Logic, orthogonal and orthonormal expansions, and their applications. Unfortunately, as pointed out by our readers, the term "orthogonal" has some other meaning when applied to transforms. It is quite confusing in our case, since the standard meaning of this term would imply a restricting definition. The so-called "orthogonal" transforms introduced by us do not need to be orthogonal in a common sense, and in most cases they are not. In this paper we further generalize these concepts - to incompletely specified, multi-output, multiple-valued logic, and we use this oportunity to introduce a correct terminology as well.

What was called orthogonal matrix will be called *nonsingular matrix*, orthogonal transform will be called *nonsingular transform.* If the transform matrix is linearly independent, we will use the name *linearly independent transform.* Since in some applications we are interested not only in nonsingular sets of functions (vectors) but just in linearly independent functions, and because linear independence is most important to all our concepts, we will call the whole area of logic that we introduce - the *"Linearly Independent Logic"*, LI-logic, for short.

The area of "Linearly Independent Logic" will thus include both binary and multiple-valued functions. What was called earlier Orthogonal Logic will become now *binary Linearly Independent Logic.* The canonical forms will be called *Non-singular Forms* (NS) forms. The Universal XOR canonical forms introduced in [12] become now a special case of binary NS forms.

The LI-logic is based on the concept of *linear gate.* Let us assume a set of values of multiple-valued (mv) logic; let assume that 0 is a unique value of the algebra, and that for each value a there exists a unique value a^{-1}. A *linear gate* is a gate executing operation $+$ such that:

$$0+a=a+0=a; \qquad a^{-1}+a=a+a^{-1}=0$$

The linear gate can be for instance a modulo_k addition, a Galois addition, or a complex modulo addition. In the simplest variant of LI-logic an arbitrary mv function is implemented as a *multi-input linear gate* of *NS terms*, where NS terms are arbitrary linearly independent mv functions. Our interest is in how to generate NS terms and how to minimize the number of terms in the NS form.

UXF forms were shown to encompass all possible canonical forms based on XOR of general uxf terms. Similarly, NS forms include all possible canonical forms based on **any** linear gate of general NS terms. Thus NS forms include UXF forms which in turn include classical AND/XOR forms. A linear algebraic treatment for generation of UXF forms was presented and their number was calculated [12]. Furthermore, a useful subset of these canonical forms, namely AND/OR/XOR canonical forms were introduced. In this paper we introduce two families of NS forms that will allow to realize functions in regular two-dimensional layouts with only local connections and buses. These families are mv generalizations of some UXF families. Although our generalization is for an arbitrary number of values, and even different number of values for every variable, we use here only ternary case for the lack of space.

Both of the NSF families are based on multi-input linear gates that collect the outputs of NS terms (called also *basis functions* or *standard trivial functions*). The first family is based on the same vector space treatment of the functions as in UXFs, but for larger fields than GF(2). This would be what we term as the "polynomial" family of forms. This approach requires, at the minimum, only one kind of gate other than linear: a multiplication gate to create NS terms. This is an analogy to Positive Polarity Reed-Muller forms that require only EXOR gates in the collecting plane and only AND gates in terms. In Polynomial Forms, additional arbitrary functions of single variable can be used as "literal generators", which generalizes the concept of polarity, and is a counterpart of the binary Fixed Polarity RM. The second family builds NS terms using Post-like operations MIN and MAX, which are mv counterparts of binary AND and OR gates.

The paper is organized as follows. In section 2, fundamental definitions and theorems for nonsingular and linearly independent logic are given. Section 3 discusses generative approaches to creation of bases, that use generalized matrix operations. Section 4 discusses circuit realizations of some NS forms, Sections 5 - 7 present various generative forms for binary and multiple-valued logic: particularly, section 5 explains the Positive Polarity NS forms, section 6 the Fixed Polarity NS forms, and section 7 the Kronecker NS forms. Section 8 lists some other interesting families of NS forms. Section 9 explains the transformative approach to forms generation, and section 10 concludes the paper.

II. Fundamental Theorems for Non-singular Logic

In [12], it was shown that the set of n-variable Boolean functions under addition mod-2 forms a vector space under which each basis provides a unique representation of a Boolean function. In other words, every Boolean function can be represented uniquely as Exclusive-or sum of the elements of the basis, called *basis functions*. The set of all bases forms what is termed as Universal XOR canonical forms, UXF. The task of identification of all these bases was related to the transformations from one basis to another. This transformation was formulated in terms of nonsingular matrices. Given one basis, any nonsingular matrix would transform this basis to another basis. Hence, the identification of all possible bases was formulated in terms of identification of all nonsingular matrices to be multiplied by e.g. the minterm basis. The number of nonsingular matrices is given be following lemmas.

Lemma 1 *Let $k = GF(q)$ be the Galois field with q elements. The order of the group of all nonsingular m-by-m matrices with entries in the field k is:*

$$q^{m(m-1)/2} \prod_{i=1}^{m} (q^i - 1). \qquad (1)$$

Lemma 2 *Let $f(x_1, x_2, ..., x_n)$ be a Boolean binary function of n variables. The number of all possible XOR canonical representations of the function is given by:*

$$\frac{2^{(2^{n-1})(2^n-1)}}{2^n!} \prod_{i=1}^{2^n} (2^i - 1). \qquad (2)$$

By above lemma, the number of different XOR canonical forms for a 2-variable binary function was shown to be 840.

Definition 1 *Let f_i be a k-valued switching function of n variables taking values in a set with k elements. Let us assume that the values satisfy the axioms of the linear gate $+$. Let M be a k^n by k^n matrix with columns corresponding to k-valued mv minterms and rows corresponding to functions f_i. Let the set of rows be linearly independent with respect to linear $+$ operation. The set of all such functions forms a k^n-dimensional vector space over the Galois field of k elements. Any basis in this vector space provides a unique representation of the switching function. Hence, all possible bases in a given field provide what will be termed as Nonsingular Universal Canonical Forms, Nonsingular Forms, or NS forms, for short. Any particular NS form (NSF) is uniquely determined by the above matrix.*

In view of these basis functions, it is possible to extend the Fundamental Theorem given in [13] to the case of all NS forms. This extension will be given in the following theorems.

Theorem 1 *Let f_i be k-valued switching functions and n the number of variables in the functions. Let M be a k^n by k^n matrix with columns corresponding to multiple-valued minterms and rows corresponding to f_i. If the set of rows is linearly independent with respect to linear $+$ operation, then the matrix is nonsingular and has an inverse matrix M^{-1}. In particular, this inverse can be used for transformation from the minterm basis to a basis specified by M.*

Theorem 2 *If basis functions are linearly independent, and there is only $K < k^n$ of them, then $k^n - K$ other linearly independent basis functions can be added to make a nonsingular set of basis functions.*

Theorem 3 *Given is a p-output function $F(x_1,...,x_n) = [F(x_1,...,x_n)^1, F(x_1,...,x_n)^2, ... F(x_1,...,x_n)^i ......, F(x_1,...,x_n)^p]$, represented as a matrix FV with k^n rows corresponding to its mv minterms in binary ordering, and p columns corresponding to each single-output component function. There exists a canonical three-level realization of each component function*

$F(x_1,...,x_n)^i = f_0 \cdot S_0{}^i \oplus ... \oplus f_{2^n-1} \cdot S_{2^n-1}{}^i$, *where functions f_i are the given basis functions, and coefficients $S_j{}^i$ are determined by the coefficient matrix $CV = (M^T)^{-1} \cdot FV$. T stands for matrix transpose. Operation $\cdot$ is mod-k multiplication in case of polynomial families, and MIN in case of MIN/MAX families.*

Theorem 4 *Given is an p-output function $F(x_1,...,x_m)$ $[F(x_1,...,x_n)^1, F(x_1,...,x_n)^2, ... F(x_1,...,x_n)^i ......, F(x_1,...,x_n)^p]$, such that the set of input variables $x_1,...,x_m$ includes properly the set $\{x_1,...,x_n\}$. There exists an unique expansion*

$F(x_1,...,x_m)^i = f_0(x_1,...,x_n) \cdot SF_0{}^i(x_{n+1},...,x_m) \oplus ... \oplus f_{2^n-1}(x_1,...,x_n) \cdot SF_{2^n-1}{}^i(x_{n+1},...,x_m)$, *where functions f_i are the given basis functions of n variables, and the coefficient functions (called also the "data input functions") $SF_j{}^i$ of the remaining input variables are determined from the coefficient matrix $CV = (M^T)^{-1} \cdot FV$.*

We will denote $M^T = N$. See [8] - [15] for examples and applications. Above theorems can be also easily extended to the case that every variable has a different number of values.

Most of the properties of UXFs shown in [13, 12] are preserved in the NS forms as well. For each NS form there exists a Generalized PLA circuit. For each NS form for which Theorem 4 holds there exists a programmable Generalized PLA circuit, a Universal Logic Cell, and an expansion for a function that has more than n variables. Similarly for all those forms there exists a generalization of the concept defined in [13] as the Orthogonal Decision Diagram. We will call the new diagrams based on NS expansions the *Linearly-Independent Decision Diagrams*, or LI-DDs, for short. All the well-known categories of decision diagrams, such as, Positive Polarity, Single Polarity, Fixed Polarity (Functional DDs), Kronecker, and Pseudo-Kronecker, MV, and Orthogonal Diagrams have their counterparts here and can be easily reconstructed by a careful reader using methods from [9, 10, 19].

For binary functions, the matrix multiplication method can be generalized to *incompletely specified functions* as follows. In formula $FV = N \cdot CV$ we represent spectral coefficients from CV as symbols $S_{i,j}$, $i = 1, ...2^n$, $j = 1, ..., p$. Now, for each care minterm $FV_{r,s}$ from matrix FV we create an equation

$$\bigoplus_{(i,j)} S_{i,j} \odot FV_{r,s} = \bigoplus_{(i,j)} S_{i,j} \oplus \overline{FV_{r,s}}, \quad (3)$$

where $\odot$ is the equivalence operator. Next for all care minterms one creates the decision formula:

$$\prod_{((s,r)\ is\ a\ care\ minterm)} (\bigoplus_{(i,j)} S_{i,j} \oplus \overline{FV_{r,s}}) = 1 \quad (4)$$

which plays the same role as the Petrick function in SOP minimization, and the Helliwell function in ESOP minimization. By finding the minimum set of variables $S_{i,j}$ that satisfies this formula we find the exact solution for the selected NS form N. There exist quite efficient *approximate* binary search algorithms for this problem. Let us observe that the more strongly unspecified is the function, the more efficient the proposed method becomes. A similar method was created for k-valued functions (Only left part of formula (3) is used where: $\oplus$ is interpreted as $+$ operator, $a \odot b = 1\ iff\ a = b,\ otherwise\ 0$. Branching of nodes in search uses all values $0, 1, ...k-1$ of variables $S_{i,j}$).

III. Generative Families

UXFs and their linear algebraic treatment were shown to encompass a huge number of possible canonical forms. Obviously, UXFs include, as their relatively very small subset, all possible AND/XOR canonical forms which have been a subject of several studies in the literature such as PPRMs, FPRMs, KRMs and GRMs. The sets of all UXFs, however, is substantially larger and includes other more general terms than just products of literals as basis functions. Among these numerous forms, some seem to have more immediate applications in current technologies. These are mostly those that are based on regular array-like connections of AND, OR, and NOT gates [13, 18]. While in such FPGAs as the Lookup-table based ones, the distinction between different logics is removed, the above logical operations continue to be the most widely used in other array technologies, and especially in fine-grain FPGAs, such as those from ATMEL, Xilinx (Algotronix) and Motorola (Pilkington).

Generation of different AND/OR/XOR canonical forms was discussed in [12] based on two operations of Reed-Muller and AND/OR. Here, it will be shown that similar to various AND/XOR canonical forms, it is possible to define even more general families of binary NS forms. The operations presented in this paper demonstrate that for each family of AND/XOR canonical forms there exist a super-family of more generalized forms that use arbitrary binary gates, and next its super-super family in multiple-valued logic.

Although formally our generalizations look similar to those from [3, 7, 19, 11], they are in such relation to the ones from [12], as for instance the Generalized Reed-Muller forms are to Positive Polarity forms in case of classical AND/XOR forms, or, as the binary forms to mv forms. Therefore, the practical importance of the new forms is much broader.

There exist *Reed-Muller Canonical (RMC)* form [16], introduced earlier by Zhegalkin [21], *Consistent Generalized Reed-Muller (CGRM)* Canonical forms [3, 17] introduced by Akers [1], also known as *fixed polarity Reed-Muller (FPRM)* Canonical forms [19]. *Generalized Reed-Muller (GRM)* Canonical forms [3], also termed as *Canonical Restricted Mixed Polarity (CRMP)* [11, 2], *Kronecker Reed-Muller (KRM)* canonical forms [3, 6], *Pseudo-Kronecker Reed-Muller (PKRM)* canonical forms, *Quasi-Kronecker Reed-Muller (QKRM)* canonical forms, etc. We show here that all these forms have their generalizations.

Many, but not all, families of AND/XOR forms can be generated by Kronecker Products of matrices and Decision Diagrams. The same is true for NS forms. We will show that there exist two basic ways of creating nonsingular transform matrices: *Generative methods* and *Transformative methods.* While the generative methods create matrix N recursively from core matrices for single variables or core matrices for groups of variables, or use LI-DDs to create multi-level circuit that is next flattened, the transformative methods transform a nonsingular base to another nonsingular base. Matrix generation can be done by three methods: standard matrix multiplication, Kronecker matrix multiplication, and *Generalized Matrix Operators* (for examples of such operators see [12, 4, 5]). The generalized matrix operator should be interpreted as a generalization of the Kronecker Matrix Product, in which the symbol b of the first matrix M_{oper} is interpreted as a symbol of an **operator** b to be applied to the respective second matrix, M_{data}. For instance, for 2×2 matrices, the generalized operator is defined as follows:

$$M_{oper} \diamond M_{data} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \diamond \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \quad (5)$$

$$= \begin{bmatrix} apply\ a\ to \begin{bmatrix} e & f \\ g & h \end{bmatrix} & apply\ b\ to \begin{bmatrix} e & f \\ g & h \end{bmatrix} \\ apply\ c\ to \begin{bmatrix} e & f \\ g & h \end{bmatrix} & apply\ d\ to \begin{bmatrix} e & f \\ g & h \end{bmatrix} \end{bmatrix} \quad (6)$$

This way, the standard Kronecker multiplication becomes a special case of our generalized operator. Section 5 will illustrate several operators.

Let us observe that if matrix N can be factorized, then a more efficient way of finding the transform exist:

If $CV = N_1 \cdot N_2 \cdot \ldots\ldots \cdot N_r \cdot FV$

then $FV = {N_r}^{-1} \cdot {N_{r-1}}^{-1} \cdot \ldots\ldots \cdot {N_1}^{-1} \cdot CV$

Similar formulas exist for Kronecker multiplication, and Generalized Matrix operators. It should be pointed out, that matrices N_i can be any nonsingular matrices of size comparable to any subset of input variables, not necessarily to a single variable. Obviously, a mv variable with 2^k values requires k binary wires to be coded, so uniform methods can be applied to circuit realization of binary and mv logic. For instance, a cell for operator $+$ can be realized with two EXORs; in general, a two-input quaternary gate can be realized as a cell with four binary inputs and two binary outputs. Because of the regularity of our structures, the pairs of these signals that represent a 4-valued variable are always routed together, and are very short or are used in global buses.

IV. Regular Circuit Realizations of Two Families of NS Forms

While the methods presented in last two sections are general and allow to generate many NS forms, there are the following problems:

- How to generate matrices of forms that have some desirable circuit properties?
- What are the families of forms that have certain structure?
- The well known AND/XOR forms are particular cases of NS forms. Are there any special familes of NS forms that are generalizations of these AND/XOR forms and have similar regular structures?

One subfamily of NS Forms will be called the *Polynomial family of NSFs.* It is obvious from well-known properties of fields, that Theorem 1 is satisfied for any k-valued function where k is a prime number or a power of a prime number ($k = p^r$, $r \neq 0$, p = prime number). The operations in this field, rather than AND and XOR for the case of $GF(2)$, will be the mod-k

multiplication and the mod-k addition. Or, they can be the $Galois_k$ multiplication and the $Galois_k$ addition. The linear + operations (additions) are done in the output (collecting) plane and in the input plane there are only multipliers and wires. This is the reason that we call it also the *"Multiply/wire" expansion.*

A k-valued switching function in this representation will be a polynomial where the k-valued basis functions form the Multiply terms and the k-valued coefficients determine the exact unique representation of the function in that field. In other words, familiar algebraic techniques will be used to represent the k-valued function. However, in our generalization the polynomial can use not only the original variables, but also arbitrary functions from a set of single-variable nonsingular functions. For instance, the arguments of Multiply terms can be any permutations of function "a variable", f(i) = i, i=0,..,k-1. As an example, for ternary logic, function "variable" and some of its permutations are as follows:

$f(0) = 0,\ f(1) = 1,\ f(2) = 2;$
$fnot(0) = 2,\ fnot(1) = 1,\ fnot(2) = 0;$
$fr(0) = 2,\ fr(1) = 0,\ fr(2) = 1;$
$fb(0) = 1,\ fb(1) = 0,\ fb(2) = 2;$

In "MIN/MAX/wire" forms the operations to create basis functions will be that of minimum and maximum, as for instance in Post logic. These basis functions are then again collected by a linear gate as in the previous case. While in the Multiply/wire forms the basis functions are in terms of powers of variables or powers of single-variable permutations, in this case, the terms take multiple-valued literals that take constant values from 1 to $k-1$. The MIN operation in this multiple-valued logic can be seen as the counterpart of AND for binary logic. It is simply the minimum value of the values of its two arguments. The MAX operation can be seen as the counterpart of OR for binary. It is the maximum value of the values of two variables.

As examples, we will present one basis set in the Polynomial family, and one basis set in the Min/Max family.

Example 1 *A basis set for ternary functions in Polynomial family includes basis functions 1, b, b^2, a, ab, ab^2, a^2, a^2b, and a^2b^2. The single argument operations are here the powers of variables, and the two-argument operation is the mod-k multiplication. Single variable nonsingular functions are here generalizations of polarities of variables in binary logic.*

Example 2 *A basis set for ternary functions in Min/Max NSFs is represented by basis functions 2, b^{12}, b^2, a^{12}, $a^{12}b^{12}$, $a^{12}b^2$, a^2, a^2b^{12}, a^2b^2. The single argument operations are Post literals ($a^S = 1$ if $a \in S, a^S = 0$ otherwise), and the two-argument operation is the minimum.*

In particular, the theorems from the previous sections are true for all the NS forms, their special families from the literature, the Polynomial family, and the MIN/MAX family. They are very useful to find and verify some properties of new NS forms.

There are several applications of these theorems [13, 12, 10, 4, 5]. They can be used to calculate expansion coefficients for any given mv function F in any nonsingular expansion, to be realized by a *"Generalized PLA", GPLA*. It is done by a generalization of procedures to find coefficients in AND/XOR canonical forms. Other application is to calculate the input functions to a Universal Logic module knowing its output function, as a part of a multi-level synthesis method. Such universal modules generalize multiplexers and AND/XOR gates used for realizations of Shannon, and Davio expansions. Finally, similarly to algorithms that go through all Fixed Polarity Reed-Muller forms or other canonical forms [20, 22] in order to select the best expansion for a given function, an exhaustive search algorithm to find the best expansion can be created. Creation of such methods can be simplified with respect to fast permutations and other linear operations on matrices N and N^{-1}, as well as the existence of fast transforms for binary [4, 5] and ternary logic.

In NSFs, the nonsingular matrix and the operations are defined over a finite field, $GF(k)$. Therefore, similar to UXfs, any nonsingular matrix will transform one basis to another, or one NSF to another. The number of such bases can again be found through equation 2, replacing all instances of 2 with k.

In cellular realizations of various AND/OR/wire terms in [13, 12], AND, OR, and wires were needed in a column of a generalized PLA. The rows in the *input plane* (called also *a basis plane*) of this GPLA correspond to the inputs, or to inputs and their negations. The rows of the output plane were composed of XOR gates and wires.

Similarly, in the more general NSFs, various terms can be realized in various cellular realizations. All these realizations have three component structures: *one-argument functions*, the *input plane,* and the *output plane.* One-argument functions are: Post-like literals, variables (wires), modulo-k powers of variables, permutation functions, or any other generalized literals (arbitrary functions of single variable). These are nonsingular sets of functions of a single variable. The outputs from one-argument functions are horizontal inputs, *input buses*, to the input plane. Columns of the input plane realize linear combinations of the following two-input gates: MIN, MAX, and MULTIPLY gates. Each gate in the input plane takes one input from its top gate and one from the corresponding input bus, and sends its output to the lower gate. Each lowest gate in the input plane starts the vertical intermediate bus going to the output plane. In Universal Modules, the gates MIN or MULTIPLY are inserted on the path from the input plane to the output plane, possibly at every input to a linear gate. The two-input linear gates create the output plane. Each gate takes its one input from the respective vertical bus and one from the previous (left) linear gate in this row, and gives its result to the next (right) linear gate. The cells that realize all above gates can be also switched to perform the role of wires, both in the input and the output plane.

The following specialized variants of generalized PLAs for LI logic can be created:

- Post literals in inputs; wire, and MIN in input plane - a counterpart of Reed-Muller,
- Post literals in input; MIN, and MAX in input plane - a counterpart of AND-OR,
- Post literals in input; wire, MAX, and MIN in input plane - a counterpart of AND-OR/Reed-Muller,
- Permutation and powers in inputs: wire, and MULTIPLY-mod-k in input plane,
- Permutation and powers in inputs: wire, MULTIPLY-mod-k, MIN and MAX in input plane.

The last two cases above do not have the counterparts described before in UXFs. A Field-Programmable Analog Array (FPAA) that allows to realize such structures has been described in [15].

The generalized PLA can be further generalized allowing the cells to be multi-input, multi-output, and creating respective NS families for them. The nice thing about LI logic is that one can create his own transforms and synthesis methods for a given set of logic blocks and patterns of their connections.

V. Positive Polarity NSF Forms

From now on, the basis of reference consists of the minterms in reverse binary order with reversed bits.

Positive Polarity NS forms are the counterpart of Positive Polarity RM Forms. Here we will present three types of these forms. Binary families are generated by operators: Reed-Muller (AND/wire), dual (OR/wire) and AND/OR. Their mv generalizations are generated by operators: MIN/wire, MAX/wire and MIN/MAX. Other mv generalization uses MULTIPLY/wire operators. The examples of these transforms for ternary-valued functions will be shown.

A. Positive Polarity Binary Families

Let R be a nonsingular matrix. Operator R means taking matrix R without modification. Operator $\mathbf{i}$ creates a square matrix of the size of matrix R with all entries i. For, instance, $\mathbf{0}$ creates a square matrix with all entries 0. Operator C replaces the lowest row of argument matrix with zeros.

Definition 2 *The* Reed-Muller Operator $\boldsymbol{\rho}$ *on* R *is:*

$$\boldsymbol{\rho}(R) = \begin{bmatrix} R & \mathbf{0} \\ R & R \end{bmatrix} \tag{7}$$

Definition 3 *The* AND-OR Operator $\boldsymbol{\alpha}$ *on* R *is:*

$$\boldsymbol{\alpha}(R) = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{1} & R \end{bmatrix} \tag{8}$$

Definition 4 *The* OR/wire Operator $\boldsymbol{\beta}$ *on* R *is:*

$$\boldsymbol{\beta}(R) = \begin{bmatrix} \mathbf{1} & C \\ R & R \end{bmatrix} \tag{9}$$

It can be shown that applications of these transforms result in nonsingular bases of a higher dimension.

The starting matrix for a single binary variable, used in the generation of the positive polarity $\boldsymbol{\alpha\rho}$ family of bases, was given as:

$$T_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{10}$$

The basis represented by T_1 was shown to be:

$$\begin{bmatrix} a \\ 1 \end{bmatrix} \tag{11}$$

The Reed-Muller/AND/OR expansion is constructed by $\boldsymbol{\alpha}$ $(\boldsymbol{\rho}(T_1))$. Other similar constructs are possible - incorporating different orders of application of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\rho}$ operators - which give rise to various AND/OR/wire connections in the basis plane. While the order of variables is irrelevant for Reed-Muller basis, for AND/OR and all other combinations of the $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\rho}$ operators, it gives rise to a new basis.

Definition 5 *The family of bases generated by applications of* $\boldsymbol{\alpha}$ *and* $\boldsymbol{\rho}$ *operators in all possible orders and all possible permutations of the variables is the* positive polarity $\boldsymbol{\alpha\rho}$ family of bases.

Example 3 *The nonsingular transition matrix of AND/OR expansion for a 2-input binary function is given as:*

$$\alpha[T_1] = \begin{bmatrix} T_1 & \mathbf{0} \\ \mathbf{1} & T_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (12)$$

The transition matrix above results in the following basis functions:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} ab \\ \bar{a}b \\ a\bar{b} \\ \bar{a}\bar{b} \end{bmatrix} = \begin{bmatrix} ab \\ b \\ a+b \\ 1 \end{bmatrix} \quad (13)$$

B. Positive Polarity Ternary Polynomial Families

The counterparts of these transformations for the ternary-valued functions can be represented in the following:

Definition 6 *Let R be a nonsingular matrix. The* Ternary Reed-Muller Operator ρ_3 *on R is:*

$$\rho_3(R) = \begin{bmatrix} R & R & \mathbf{0} \\ 2R & R & \mathbf{0} \\ R & R & R \end{bmatrix} \quad (14)$$

The corresponding starting matrix for a single ternary variable is:

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (15)$$

which corresponds to the basis:

$$\begin{bmatrix} a^2 \\ a \\ 1 \end{bmatrix} \quad (16)$$

Similarly, any permutation function of a can replace all occurrences of function a in the basis above, which creates 6 single-variable bases for Ternary Reed-Muller generalization. In case of polynomial forms the generation can use either the standard Kronecker Product or its generalization in the form of the Reed-Muller operator.

Example 4 *Using the Kronecker Product $\times$, the ternary Reed-Muller NSF is found:*

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \quad (17)$$

$$= \begin{bmatrix} 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \\ 2\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \\ 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{bmatrix} = \quad (18)$$

$$= \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (19)$$

It can be easily verified, that the same result would be obtained by using the Reed-Muller operator defined in formula (14). The columns in the above matrix correspond to ternary minterms 22, 21, 20, 12, 11, 10, 02, 01, and 00 where the first value corresponds to b and the second to a. The rows resulting terms then as given before can be seen to be a^2b^2, ab^2, b^2, a^2b, ab, b, a^2, a, and 1.

C. Positive Polarity Ternary MIN/MAX Families

It is also possible to generalize the Reed-Muller UXF in MIN/MAX format. The main characteristic of the Reed-Muller expansion is that it is only involved with the AND operation in binary form. As mentioned previously, the counterpart of the AND operation in MIN/MAX generalization is the MIN. Hence, it is possible to show a generalization of the Reed-Muller expansion with the use of MIN alone.

Similarly to the polynomial family from previous subsection, it is possible in MIN/MAX to take a variety of starting matrices. In the case of ternary and multiple-valued logic in general, this can take different number of matrices. This can be also looked at as there are no positive and negative transforms anymore and we are dealing with a variety of polarities depending on the possible values for the variables. The number of polarities for a single 3-valued-input variable is 27 [10]. This gives 27 starting bases with values 0 and 1, 27 bases with values 0 and 2, and 27 starting bases with values 1 and 2. Multiplying any subsets of rows in these bases by 1 or 2 gives many more other bases. In effect, even only in MIN/MAX family, and even only for a single variable, the number of ternary bases is extremely high comparing to three bases for a binary variable, which are: Shannon, Positive Davio, and Negative Davio.

As one example, a starting matrix can be taken as:

$$\begin{bmatrix} 2 & 0 & 0 \\ 2 & 2 & 0 \\ 2 & 2 & 2 \end{bmatrix} \tag{20}$$

This gives essentially the basis functions a^2, a^{12}, and 2. By Post Logic, it is possible to choose different standard literals. Possible other examples are:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{21}$$

to give a^0, a^1, and a^2. Or a more generalized one:

$$\begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & 2 \\ 1 & 1 & 0 \end{bmatrix} \tag{22}$$

which corresponds to a^{02}, a^{01}, and $1 \cdot a^{21}$.

Example 5 *Let a and b be two ternary-valued variables. Taking the basis functions a^2, a^{12}, and 2 as the basis for a single variable, the NSF basis of MIN forms for two variables can be represented as:*

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 2 & 0 & 2 & 2 & 0 & 2 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{23}$$

The corresponding basis function will then be $a^2 \cdot b^2$, $a^{12} \cdot b^2$, b^2, $a^2 \cdot b^{12}$, $a^{12} \cdot b^{12}$, b^{12}, a^2, a^{12}, and 2. The symbol $\cdot$ stands for the MIN operation.

A generalization of $OR/wire$ operator to $MAX/wire$ for the ternary variable case is shown in the following example.

Example 6 *Let a and b be two ternary-valued variables. Taking the basis functions a^2, a^{12}, and 2 as the basis for a single variable, the NSF basis for two variables can be represented as:*

$$\begin{bmatrix} 2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 2 & 0 & 2 & 2 & 0 & 2 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{24}$$

The corresponding basis function will then be $a^2 + b^2$, $a^{12} + b^2$, b^2, $a^2 + b^{12}$, $a^{12} + b^{12}$, b^{12}, a^2, a^{12}, and 2. The symbol $+$ stands for the MAX operations.

In the RM forms, Dual RM forms, and polynomial forms, the solution does not depend on the order of variables applied in the generating operators. In contrast, in binary UXFs, different orders of application of Reed-Muller and AND/OR operators as well as different permutations of the variables, result in different new bases. The same is true for the multiple-valued Reed-Muller and MIN/MAX operators.

Therefore, for each binary Positive Polarity NS form for given order of variables, one can find all possible Fixed Polarity NS forms by consistently inverting subsets of variables. Thus for n variables, there are n! orders of variables for which this matrix can be generated. Moreover, for each of these orders, one can generate all 2^n subsets of negated variables and thus 2^n different binary Fixed NS Polarity Forms. For each Positive Polarity form there are then $n!{\cdot}2^n$ Fixed Polarity NS forms. Similar approaches can be applied to mv logic. Creation of Fixed Polarity NS forms can be also done in another way, as shown in the following section.

VI. Fixed Polarity NS Forms

For UXFs, it is possible to consistently transform the Reed-Muller and AND/OR bases, and their combinations, into 2^n different fixed polarity bases. This family of bases has been termed the *Consistent Generalized* $\boldsymbol{\alpha\rho}$ *(*$\boldsymbol{CG\alpha\rho}$*) family of forms* [14].

The negation of variables required in this family was shown to require two negation operations and a new starting transformation matrix. The negative polarity basis of a single binary variable given by T_2 is shown below:

$$T_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \tag{25}$$

which gives essentially the basis

$$\begin{bmatrix} \bar{a} \\ 1 \end{bmatrix} \tag{26}$$

Now, corresponding to the $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$ operators, $\bar{\boldsymbol{\rho}}$ and $\bar{\boldsymbol{\alpha}}$ operators are defined as the following:

Definition 7 *Let R be a nonsingular matrix. The* Negative Reed-Muller Operator $\bar{\boldsymbol{\rho}}$ *on R is:*

$$\bar{\boldsymbol{\rho}}(R) = \begin{bmatrix} \mathbf{0} & R \\ R & R \end{bmatrix} \tag{27}$$

Definition 8 *Let R be a nonsingular matrix. The* Negative AND-OR Operator $\bar{\boldsymbol{\alpha}}$ *on R is:*

$$\bar{\boldsymbol{\alpha}}(R) = \begin{bmatrix} \mathbf{0} & R \\ R & \mathbf{1} \end{bmatrix} \tag{28}$$

Theorem 5 *The $\bar{\boldsymbol{\rho}}$ and $\bar{\boldsymbol{\alpha}}$ operators result in nonsingular matrices of a higher dimension.*

Now, each variable can take either a positive or a negative operation and thus there exist 2^n possible Consistent Generalized forms for each positive $\boldsymbol{\alpha\rho}$ family of bases.

Example 7 *In the following, a $\boldsymbol{CG\alpha\rho}$ basis of three variables will be shown. Here, the order and the polarity of the variables is given as: $\bar{b}\bar{c}\bar{a}$, where the "natural" order of variables is assumed to be abc. First, the transition matrix for the natural order is generated and then the corresponding transition matrix for the given order will be shown.*

$$\boldsymbol{\rho}\bar{\boldsymbol{\alpha}}(T_2) = \begin{bmatrix} \bar{\boldsymbol{\alpha}}(T_2) & \mathbf{0} \\ \bar{\boldsymbol{\alpha}}(T_2) & \bar{\boldsymbol{\alpha}}(T_2) \end{bmatrix};\ \bar{\boldsymbol{\alpha}}(T_2) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{29}$$

The transition matrix above results in the following basis functions:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} abc \\ \bar{a}bc \\ a\bar{b}c \\ \bar{a}\bar{b}c \\ ab\bar{c} \\ \bar{a}b\bar{c} \\ a\bar{b}\bar{c} \\ \bar{a}\bar{b}\bar{c} \end{bmatrix} \tag{30}$$

$$= \begin{bmatrix} \bar{a}\bar{b}c \\ \bar{b}c \\ (\bar{a}+\bar{b})c \\ c \\ \bar{a}\bar{b} \\ \bar{b} \\ \bar{a}+\bar{b} \\ 1 \end{bmatrix} \tag{31}$$

The corresponding transition matrix for the "bca" ordering will be:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} abc \\ \bar{a}bc \\ a\bar{b}c \\ \bar{a}\bar{b}c \\ ab\bar{c} \\ \bar{a}b\bar{c} \\ a\bar{b}\bar{c} \\ \bar{a}\bar{b}\bar{c} \end{bmatrix} \tag{32}$$

$$= \begin{bmatrix} \bar{b}\bar{c}a \\ \bar{c}a \\ (\bar{b}+\bar{c})a \\ a \\ \bar{b}\bar{c} \\ \bar{c} \\ \bar{b}+\bar{c} \\ 1 \end{bmatrix} \tag{33}$$

Operators for all binary Fixed Polarity NS transforms have been classified. The counterpart of the concept of positive and negative polarities for multiple-valued variables is much more powerful than that of *polarity* for binary variables. We work on a characterization of all ternary and quaternary expansions, similar to those that have been done for binary and multiple-valued-input, binary-output expansions [7, 10, 4, 14].

VII. Families of Kronecker Type

A. Kronecker AND/OR/wire Forms

A different generalization of the $\boldsymbol{\alpha\rho}$ family of UXF bases is through the introduction of the Shannon operator, $\boldsymbol{\sigma}$. This we will call $\alpha\rho\sigma$ Family of Bases. For given ordering of variables, this family has 4^n different expansions, since for every variable operator AND/OR $\boldsymbol{\alpha}$, AND/wire (Reed-Muller) $\boldsymbol{\rho}$, OR/wire (Dual Reed-Muller) $\boldsymbol{\beta}$, or Shannon $\boldsymbol{\sigma}$, can be used.

The starting transformation matrix for $\boldsymbol{\sigma}$ extension is that of T_3:

$$T_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{34}$$

Definition 9 *Let R be a nonsingular matrix. The* Shannon Operator $\boldsymbol{\sigma}$ *on R is:*

$$\boldsymbol{\sigma}(R) = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} \tag{35}$$

Theorem 6 *The $\boldsymbol{\sigma}$ operator results in a nonsingular matrix of a higher dimension.*

Definition 10 *Let R be a nonsingular matrix. The* Negative Shannon Operator $\bar{\sigma}$ *on R is:*

$$\bar{\sigma}(R) = \begin{bmatrix} \mathbf{0} & R \\ R & \mathbf{0} \end{bmatrix} \tag{36}$$

The basis for a single element here is:

$$T_4 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{37}$$

which gives essentially the basis

$$\begin{bmatrix} \bar{a} \\ a \end{bmatrix} \tag{38}$$

It can be observed that T_3 and T_4 define essentially the same basis and no new bases will be generated by negative Shannon operator when the corresponding positive operator is present.

Theorem 7 *The $\bar{\sigma}$ operator results in a nonsingular matrix of a higher dimension.*

Operators for all binary Kronecker NS forms can be classified. Similar classifications can be applied to ternary and other mv forms.

VIII. Other Generative Type Forms

Similarly as in previous sections, four types of forms can be generated:
- positive polarity forms,
- single polarity forms,
- fixed polarity forms,
- mixed polarity forms.

For instance, the following list, by no means inclusive, can be compiled.

For binary logic: Positive Polarity AND/OR forms are AND/OR counterparts of Positive Polarity Reed-Muller forms (PPRM forms in unified terminolgy should be called Positive Polarity AND/wire forms).
Fixed Polarity AND/OR forms are AND/OR counterparts of Fixed Polarity Reed-Muller forms (FPRMs in unified terminolgy should be called Fixed Polarity AND/wire forms).
Mixed Polarity AND/OR forms are AND/OR/wire counterparts of Kronecker Reed-Muller forms (KRM forms in unified terminolgy should be called Kronecker AND/wire forms). In general, Kronecker stands for mixing polarities.

The following families of multiple-valued polynomial forms exist: Positive Polarity Polynomial (Multiply/wire) forms which are generalizations of Positive Polarity Reed-Muller forms in which Multiplier is taken instead of AND gate, and all variables have positive polarity. Single Polarity Multiply/wire forms which are generalization of Positive (or Negative) Polarity Reed-Muller forms in which Multiplier is taken instead of AND gate, and all variables have the same polarity. Kronecker Multiply/wire forms are a generalization of Fixed Polarity Reed-Muller forms in which Multiplier is taken instead of AND gate, and for every variable the expansion is done with respect to one of the possible polarities.
The following families of multiple-valued MIN/MAX forms exist: Positive Polarity MIN/MAX forms which are generalizations of Positive Polarity AND/OR forms in which MIN gate is taken instead of AND gate, MAX gate is taken instead OR gate, and all variables have positive polarity. Positive Polarity MIN/MAX/wire forms which are generalizations of Positive Polarity AND/OR/wire forms in which MIN gate is taken instead of AND gate, MAX gate is taken instead OR gate, and all variables have positive polarity. Single Polarity MIN/MAX forms which are generalization of Fixed Polarity Reed-Muller forms in which MIN gate is taken instead of AND gate, MAX gate is taken instead OR gate, and all variables have the same polarity. Kronecker MIN/MAX/wire forms are generalization of Fixed Polarity Reed-Muller forms in which MIN gate is taken instead of AND gate, MAX gate is taken instead OR gate, and for every variable the expansion is done with respect to one of possible polarities.

IX. Transformative Families

It is well known that in binary AND/XOR case the Generalized Reed-Muller forms are a different kind of family than the Kronecker type families (Pseudo, Quasi), and cannot be generated by Kronecker multiplications. This property of families for which no generative structure exists is preserved in NSF forms, both for binary and mv logic. Therefore, if finding generative operators (as presented in section III) is not possible for some family of forms represented by matrices, the transformative methods should be tried. There are many such methods that can be adopted from the existing data-flow methods in AND/XOR logic.

The transformative methods operate on a complete basis (matrix) and create new bases from it. They use two theorems.

Theorem 8 *If a row of a nonsingular matrix is replaced by a column-by-column linear operation of other rows, the new matrix is nonsingular.*

Theorem 9 *If a column of a nonsingular matrix is* *replaced by a column-by-column linear operation of other columns, the new matrix is nonsingular.*

The above row and column exoring operations can be represented by matrix multiplications. Similarly many other matrix transformations can be created and described as transform matrix multiplications by certain *Base Transform* matrices. When some additional constraints are imposed on these transformations, some restricted families of forms are created.

The transformative methods are based on the theorem.

Theorem 10 *Let* $CV_1 = {N_1}^{-1} \cdot FV$, *and* $CV_2 = {N_2}^{-1} \cdot FV$ *are two coefficient matrices obtained from the same function vector* FV *by using various nonsingular transform matrices* N_1 *and* N_2. *Let* $N_2 = N_1 \cdot B$, *where* B *is a* Base Transform Matrix. *Then* $CV_2 = B^{-1} \cdot CV_1$.

When one can find matrix B, then instead of calculationg first all matrices N_i and next matrices CV_i, we can calculate an initial CV_1 and next all CV_i forms iteratively one by one by use of Theorem 10. Dataflow methods similar to those known from AND/XOR forms are used to implement successive multiplications by Base Transform matrices.

We were, however, not able to find matrix B for GRM forms and their NS generalizations, and we resort to other techniques. Let us denote literals by small variables and terms by capital variables (a letter is a term as well, so capital letter can stand for literals). The GRM forms are obtained from other forms, such as FPRM, by applying rules:

$a \oplus 1 = \bar{a}, \quad \bar{a} \oplus 1 = a, \quad a(B \oplus C) = aB \oplus aC$

$A \oplus A = 0, \quad A \oplus 0 = A, \quad A \oplus Ab = A\bar{b}.$

The transformations are carried until an ESOP expression becomes a GRM form.

Similarly to GRMs, the members of the $\alpha\rho$ family of bases need not be confined to fixed polarities in order to provide new bases. The polarity of the literals can be "inconsistently" varied and still result in a new basis, provided that the new basis functions are obtained from valid transformations. The forms obtained like this are generalizations of the Generalized Reed-Muller AND/XOR forms.

From Theorem 8, any two basis functions can be exored to produce a new basis function. Since however the result is not necessarily in $\alpha\rho$ form, applying above rules plus the rules:

$a \oplus (A + a) = A\bar{a}, \quad a + B = a \oplus B\bar{a}$

is only allowed. This will be illustrated in the following example.

Example 8 *The basis generated in Example 7,* $[\bar{b}\bar{c}a, \ \bar{c}a, (\bar{b} + \bar{c})a, a, \bar{b}\bar{c}, \bar{c}, \bar{b}+\bar{c}, 1]^T$, *is changed to basis* $[b\bar{c}a, \bar{c}a, (\bar{b} + \bar{c})\bar{a}, a, \bar{b}\bar{c}, \bar{c}, \bar{b} + \bar{c}, 1]^T$, *by transforming* $\bar{b}\bar{c}a$ *and* $(\bar{b} + \bar{c})a$; *and next to basis* $[b\bar{c}a, ca, (\bar{b} + \bar{c})\bar{a}, a, b\bar{c}, \bar{c}, \bar{b} + \bar{c}, 1]^T$ *by transforming* $\bar{c}a, \bar{b}\bar{c}$ *and* $\bar{c}$.

Let us observe that not all variables can be individually negated in $\alpha\rho$ forms, since this could lead to nonsingular expressions. Let us take, for instance, form $[1, ab, a, a + b]$. Changing polarity of a and b in term ab would lead to form $[1, \bar{a}\,\bar{b}, a, a + b]$, with term $\bar{a}\,\bar{b}$ that is a linear combinations of other terms: $\bar{a}\,\bar{b} = 1 \oplus (a + b)$.

Concluding, the creation of NS counterparts of GRMs is more difficult than the generation of NS counterparts of AND/XOR forms that are based on Kronecker Products, Generalized Matrix operators, or LI Decision Diagrams.

X. Conclusion

The paper introduced the concept of Nonsingular transforms for binary and multiple-valued functions. In case of binary logic, the LI-logic is what we called previously Orthogonal Logic. LI-logic includes also what we called Orthonormal Transforms in [10]. We formulated theorems and showed general techniques to minimize multi-output incompletely specified functions. Theorem 1 can be used to generate new families of NS forms. All that is needed is to prove that some method of creating a family of functions makes a nonsingular matrix. It is, however, not necessary that this matrix is ever created as a data structure, so our approach can be computationally efficient.

Next, new families of NS canonical forms were introduced for both binary and multiple-valued logic. Among these, the Polynomial and Min/Max Generative Families, as well as Transformative families were described, which constitute only a small part of NS forms. The new families are very large and include all the well-known AND/XOR canonical forms, and the UXF forms introduced by us previously. Our new generalization is not only by generalizing to multiple-valued logic, but also by creating more powerful operators that allow various polarities of variables. When no generative or transformative method exists for creation of a family of forms, we can still use the fundamental approach based on Theorems 1 - 4 as the last resort.

The new binary forms require different: AND / OR / $wire$, AND / $wire$, OR / $wire$ configurations in regular (programmable) arrays. The mv forms require MIN / $wire$, MIN / MAX, MIN / MAX

/ *wire*, *wire* / *MULTIPLY*, *wire* / *MULTIPLY* / *MIN* / *MAX* configurations for their cellular realizations. It should be stressed that for logic with 2^k values a multiple-valued operator can be realized with binary gates as a k-output cell. This creates a cellular structure of simple cells with only local and bus connections, and makes a perfect match with contemporary fine-grain FPGAs and future quantum logic devices.

The presented results can find applications in the Levelized Array approach to Cellular Automata-type FPGAs [18]. They can be also used to develop new logic cells and connection structures for binary and multiple-valued Field Programmable Gate Arrays, where with respect to forthcoming fabrication technologies there will be more emphasis on reducing programmable connections than on minimizing the number of logic gates in programmable blocks. It should be observed that this kind of "regular" and "local" logic will be necessary for the realization of binary and mv circuits in quantum devices.

The new class of Decision Diagrams has been also introduced in this paper, that includes all diagrams from [8] - [12] as their small subset.

Acknowledgements

The authors acknowledge Profs. Tsutomu Sasao and Radomir Stankovic for suggesting the names "Linearly Independent" and "Nonsingular", instead of "Orthonormal" and "Orthogonal", respectively, for the introduced by us concepts.

References

[1] S. B. Akers, "On a Theory of Boolean Functions", *J. of SIAM*, Vol. 7, pp. 487-498, Dec. 1959.

[2] L. Csanky, M. A. Perkowski and I. Schäfer, "Canonical Restricted Mixed Polarity Exclusive-OR Sums of Products and the Efficient Algorithm for Their Minimization", *Proc. of IEE Pt. E*, Vol. 140, No. 1, pp. 69-77, Oct. 1992.

[3] M. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, Mc-Graw-Hill, 1978.

[4] B.J. Falkowski, and S. Rahardja, "Fast Transforms for Orthogonal Logic," *Proc. IEEE 28th Int. Symp. Circ. Syst.*, Seattle, Washington, May 1995.

[5] B.J. Falkowski, and S. Rahardja, "Family of Fast Transforms for GF(2) Orthogonal Logic," *in print*, this Workshop 1995.

[6] D. H. Green, "Reed-Muller Canonical Forms With Mixed Polarity and Their Manipulations", *Proc. of IEE Pt. E*, Vol. 137, No. 1, pp. 103-113, Jan. 1990.

[7] D. H. Green, "Families of Reed-Muller canonical forms", *Intern. J. of Electr.*, pp. 259-280, Febr. 1991.

[8] M. A. Perkowski, P. Dysko, B. J. Falkowski, "Two Learning Methods for a Tree Search Combinatorial Optimizer," *Proc. of IEEE Intern. Phoenix Conf. on Comp. and Comm.*, Scottsdale, Arizona, March 1990, pp. 606-613.

[9] M. A. Perkowski, P. Johnson, "Canonical Multi-Valued Input Reed-Muller Trees and Forms," *Proc. of the Third NASA Symposium on VLSI Design,* Moscow, ID, Oct. 30-31, 1991, pp. 11.3.1-11.3.13.

[10] M. A. Perkowski, "The Generalized Orthonormal Expansion of Functions With Multiple-Valued Inputs and Some of its Applications", *Proc. of ISMVL*, pp. 442-450, June 1992.

[11] M. A. Perkowski, L. Csanky, A. Sarabi, and I. Schäfer, "Fast Minimization of Mixed-Polarity AND/XOR Canonical Networks", *Proc. of ICCD*, pp. 33-36, Cambridge, MA, October 1992.

[12] M. A. Perkowski, A. Sarabi, F. R. Beyl, "XOR Canonical Forms of Switching Functions", *Proc. of IFIP WG 10.5 Work. on Appl. of the Reed-Muller Exp. in Circ. Des.*, Hamburg, Germany, Sept. 1993.

[13] M. A. Perkowski, "A Fundamental Theorem for EXOR Circuits", *Proc. of the IFIP WG 10.5 Work. on Appl. of the Reed-Muller Exp. in Circ. Des.*, Hamburg, Germany, Sept. 1993.

[14] M. A. Perkowski, A. Sarabi, and F. R. Beyl, "Universal XOR Canonical Forms of Boolean Functions and its Subset Family of AND/OR/XOR Canonical Forms", *Proc. of the IEEE/ACM IWLS*, Lake Tahoe, California, May 1995.

[15] E. Pierzchala, M. A. Perkowski, and S. Grygiel, "A Field Programmable Analog Arrray for Continuous, Fuzzy and Multi-Valued Logic Applications," *Proc. ISMVL'94*, pp. 148 - 155, Boston, MA, May 25-27, 1994.

[16] I. S. Reed, "A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme", *IRE Trans. on Inf. Th.*, Vol. PGIT-4, pp. 38-49, 1954.

[17] A. Sarabi and M. A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks", *Proc. of the 29th DAC*, pp. 30-35, Anaheim, CA, June 1992.

[18] A. Sarabi, N. Song, M. Chrzanowska-Jeske, and M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays", *Proceedings of the 31st DAC*, pp. 321-326, San Diego, CA, June 1994.

[19] T. Sasao, "AND-EXOR Expressions and their Optimization", in Sasao(ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 287-312, 1993.

[20] Y.Z. Zhang and P.J.W. Rayner, "Minimization of Reed-Muller polynomials with fixed polarity," *IEE Proc.*, Vol. 131, part E, pp. 177-186, Sept. 1984.

[21] I. L. Zhegalkin, "Arifmetizatsiya simbolicheskoĭ logiki (Arithmetization of Symbolic Logic)", *Matematicheskiĭ Sbornik*, Vol. 35, pp. 311-373, 1928 and Vol. 36, pp. 205-338, 1929.

[22] X. Zeng, M. Perkowski, H. Wu, and A. Sarabi, "A New Efficient Algorithm for Finding Exact Minimal Generalized Partially-Mixed-Polarity Reed-Muller Expansion," *in print,* this Workshop, 1995.