FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 29, N° 4, December 2016, pp. 701 - 720
DOI: 10.2298/FUEE1604701A

MULTI-VALUED GALOIS SHANNON - DAVIO TREES AND THEIR COMPLEXITY

Anas N. Al-Rabadi[1,2]

[1]Electrical Engineering Department, Philadelphia University,
[2]Jordan & Computer Engineering Department,
The University of Jordan, Amman-Jordan

Abstract. *The idea of Shannon-Davio (S/D) trees for binary logic is a general concept that found applications in the Sum-Of-Product (SOP) minimization and the generation of new diagrams and canonical forms. Extended S/D trees are used to generate forms that include a minimum Galois Field Sum-of-Products (GFSOP) forms. Since there exist many applications of Galois field of quaternary radix especially that GF(4) is considered as an important extension of GF(2), the extension of the S/D trees to GF(4) is presented here. A general formula to calculate the number of Inclusive Forms (IFs) per variable order for an arbitrary Galois field radix and arbitrary number of variables is derived and introduced. A new fast method to count the number of IFs for an arbitrary Galois radix and functions of two variables is also introduced; the $IF_{n,2}$ Triangles. The results introduced in this work can be useful for the creation of an efficient GFSOP minimizer for Galois logic and in other applications such as in reversible logic synthesis.*

Key words: *Complexity, Galois Field Sum-of-Product (GFSOP), Galois Forms, Inclusive Forms, Multi-Valued Logic, Quaternary Logic, Shannon-Davio (S/D) Trees.*

1 Introduction

Spectral transforms play an important role in the synthesis, analysis, testing, classification, formal verification and simulation of logic circuits and systems. Dyadic families of discrete transforms; Reed-Muller and Green-Sasao hierarchy, Walsh, Arithmetic, Adding and Haar transforms and their generalizations to p-adic (multi-valued) transforms, have found a fruitful use in digital system design [1, 2, 6-35]. Reed-Muller-like spectral transforms [2-6, 12-14, 16-18, 21, 25, 27, 29, 33, 35] have found a variety of useful applications in minimizing Exclusive Sum-Of-Products (ESOP) and Galois field SOP (GFSOP) expressions, creation of new forms, binary decision diagrams, spectral decision diagrams, regular

Received November 28, 2015; received in revised form April 13, 2016
Corresponding author: Anas N. Al-Rabadi
Electrical Engineering Department, Philadelphia University, Jordan & Computer Engineering Department,
The University of Jordan, Amman-Jordan
(email: alrabadi@yahoo.com)

structures, besides their well-known uses in digital communications, digital signal processing, digital image processing and fault detection and testing [1-7, 12, 13, 15-19, 21-25, 27, 29, 32, 35]. The method of generating the new families of multi-valued Shannon and Davio spectral transforms is based on the fundamental multi-valued Shannon and Davio expansions, respectively.

The remainder of this paper is organized as follows: Basic definitions of the fundamental binary expansions and their multi-valued extensions are given in Section 2. Section 3 presents the quaternary Galois Shannon-Davio (S/D) Trees. The number of S/D inclusive forms and the new $IF_{n,2}$ triangles are introduced in Section 4. Conclusions and future work are presented in Section 5.

2 Basic Shannon and Davio Decompositions

This Section presents necessary mathematical background and the fundamental formalisms of the work that will be introduced and further developed in the following Sections. Normal canonical forms play an important role in the synthesis of logic circuits which includes synthesis, testing and optimization [2, 8, 13, 15, 17, 18, 21, 23, 25, 27, 29, 32, 35-37]. The main algebraic structure which is used in this work for developing the canonical normal forms is the Galois field (GF) algebraic structure, which is a fundamental algebraic structure in the theory of algebras [2, 8, 17, 18, 21, 32]. Galois field has proven high efficiency in various applications of logic synthesis, such as in the design for test, error correction codes, and even in the proof of the well-known Fermat's last theorem. The importance of GF for logic synthesis results from the fact that every finite field is isomorphic to a Galois field. In general, the attractive properties of GF-based circuits, such as high testability of such circuits, are due to the fact that the GF operators exhibit the Cyclic Group - also called Latin Square - property which can be explained, for example, using GF(4) (quaternary) operators as shown in Figures 1(a) and 1(b), respectively; note that in any row and column of the addition table in Figure 1(a), the elements are all different which is cyclic, and that the elements have a different order in each row and column. Another cyclic group can be observed in the multiplication table; if the zero elements are removed from the multiplication table in Figure 1(b), then the remaining elements form a cyclic group. In binary, for example, GF(2) addition operator - EXOR - has the cyclic group property.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

(a)

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

(b)

Fig. 1: GF(4) addition and multiplication tables.

Reed-Muller based normal forms have been classified using the Green-Sasao hierarchy. The Green-Sasao hierarchy of families of canonical forms and corresponding decision di-

agrams is based on three generic expansions; Shannon, positive Davio and negative Davio expansions. The corresponding Shannon, positive Davio and negative Davio expansions are given as follows [2, 32]:

$$\begin{aligned} f(x_1,x_2,\ldots,x_n) &= \bar{x_1}\cdot f_0(x_1,x_2,\ldots,x_n)\oplus x_1\cdot f_1(x_1,x_2,\ldots,x_n), \\ &= \begin{bmatrix}\bar{x_1} & x_1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}f_0\\ f_1\end{bmatrix}, \end{aligned} \tag{1}$$

$$\begin{aligned} f(x_1,x_2,\ldots,x_n) &= 1\cdot f_0(x_1,x_2,\ldots,x_n)\oplus x_1\cdot f_2(x_1,x_2,\ldots,x_n), \\ &= \begin{bmatrix}1 & x_1\end{bmatrix}\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}\begin{bmatrix}f_0\\ f_1\end{bmatrix}, \end{aligned} \tag{2}$$

$$\begin{aligned} f(x_1,x_2,\ldots,x_n) &= 1\cdot f_1(x_1,x_2,\ldots,x_n)\oplus \bar{x_1}\cdot f_2(x_1,x_2,\ldots,x_n) \\ &= \begin{bmatrix}1 & \bar{x_1}\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}f_0\\ f_1\end{bmatrix}, \end{aligned} \tag{3}$$

where $f_0(x_1,x_2,\ldots,x_n) = f(0,x_2,\ldots,x_n) = f_0$ is the negative cofactor of variable x_1, $f_1(x_1,x_2,\ldots,x_n) = f(1,x_2,\ldots,x_n) = f_1$ is the positive cofactor of variable x_1, and $f_2(x_1,x_2,\ldots,x_n) = f(0,x_2,\ldots,x_n)\oplus f(1,x_2,\ldots,x_n) = f_0\oplus f_1$. An arbitrary n-variable function $f(x_1,x_2,\ldots,x_n)$ can be represented using the Positive Polarity Reed-Muller (PPRM) expansion as follows:

$$\begin{aligned} f(x_1,x_2,\ldots,x_n) =& a_0\oplus a_1x_1\oplus a_2x_2\oplus\ldots\oplus a_nx_n\oplus a_{12}x_1x_2\oplus a_{13}x_1x_3\oplus a_{n-1,n}x_{n-1}x_n \\ &\oplus\ldots\oplus a_{12\ldots n}x_1x_2\ldots x_n. \end{aligned} \tag{4}$$

For each function f, the coefficients a_i in Equation (4) are determined uniquely, so PPRM is a canonical form. If we use either only the positive literal or only the negative literal for each variable in Equation (4) we obtain the Fixed Polarity Reed-Muller (FPRM) form. There are 2^n possible combinations of polarities and as many FPRMs for any given logic function [2, 32]. If we freely choose the polarity of each literal in Equation (4), we obtain the Generalized Reed-Muller (GRM) form. In GRMs, contrary to FPRMs, the same variable can appear in both positive and negative polarities. There are $n2^{(n-1)}$ literals in Equation (4) so there are $2^{n2^{(n-1)}}$ polarities for an n-variable function and as many GRMs [32]. Each of the polarities determines a unique set of coefficients, and thus each GRM is a canonical representation of a function. Two other types of expansions result from the flattening of certain binary trees that will produce Kronecker (KRO) forms and Pseudo Kronecker (PKRO) forms for Shannon, positive Davio and negative Davio expansions. There are 3^n and at most 3^{2^n-1} different KROs and PKROs, respectively [32].

The good selection of the various permutations using the Shannon and Davio expansions as internal nodes in decision trees (DTs) and diagrams (DDs) will result in DTs and DDs, that represent the corresponding logic functions, with smaller sizes in terms of the total number of hierarchical levels used, and the total number of internal nodes needed. The minimization of the size of DD, to represent a logic function, will result in speeding up the manipulations of logic functions using DD as data structure, and the minimization of the use of memory space during the execution of such manipulations. One can observe that

by going from PPRM to GRM forms, less constraints are imposed on the canonical forms due to the enlarged set of polarities that one can choose from. The gain of more freedom (less constraints) on the polarity of the canonical expansions will provide an advantage of obtaining Exclusive-Sum-Of-Product (ESOP) expressions with less number of terms and literals, and consequently expressing Boolean functions using ESOP forms will produce on average expressions with less size as if compared to Sum-Of-Product (SOP) expressions.

In general, a literal can be defined as any function of a single variable [2, 18, 32]. Basis functions in the general case of multi-valued expansions are constructed using literals. Galois field SOP expansions can be performed on variety of literals. For example, one can use among others: K-Reduced Post literal (K-RPL) to produce K-RPL GFSOP, Post literal to produce PL GFSOP, Window literal to produce WL GFSOP, Generalized (Post) literal to produce GL GFSOP, or Universal literal to produce UL GFSOP. Figure 2 demonstrates set-theoretic relationships between the various literals, where the shaded Reduced Post literal is the type of literal that will be used through this paper. One may note that the RPL in the discrete domain is analogous to the delta function in the continuous domain.

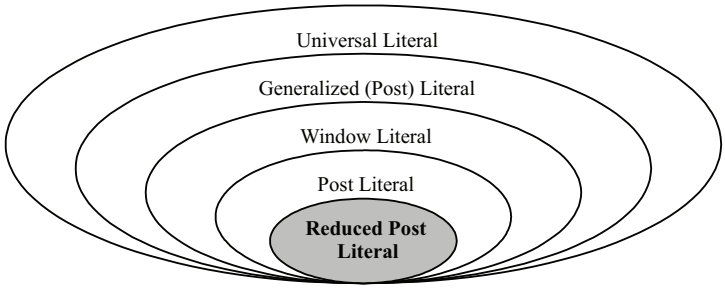


Fig. 2: Inclusion relationship of various types of literals.

Example 1. *Figure 3 demonstrates several literal types, where one proceeds from the simplest RPL literal in Figure 3(a) to the more complex WL literal in Figure 3(c). For RPL in Figure 3(a), a value K is produced by the literal when the value of the variable is equal to a specific state, and in this particular example a value* $K = 1$ *is generated by the 1-RPL when the value of variable x is equal to certain state (here this state is equal to one). Figure 3(b) shows PL where the value generated by the literal at a specific state is equal to the maximum value (i.e., radix) of that logic, and WL in Figure 3(c) generates a value equal to the radix for a "window" of specific states.*

Since K-RPL GFSOP is as simple as PL and it is simpler from implementation point of view than other kind of literals, we will perform all of the GFSOP expansions utilizing the corresponding 1-Reduced Post Literal GFSOP. Consequently, let us define the 1-RPL as [2, 32]:

$$ {}^{i}x = 1 \quad \text{iff} \quad x = i \quad \text{else} \quad {}^{i}x = 0. \tag{5}$$

For example $\{ {}^{0}x,\ {}^{1}x,\ {}^{2}x\}$ are the zero, first and second polarities of the 1-Reduced Post Literal, respectively. Also, let us define the ternary shifts over x variable $\{x,x',x''\}$ as

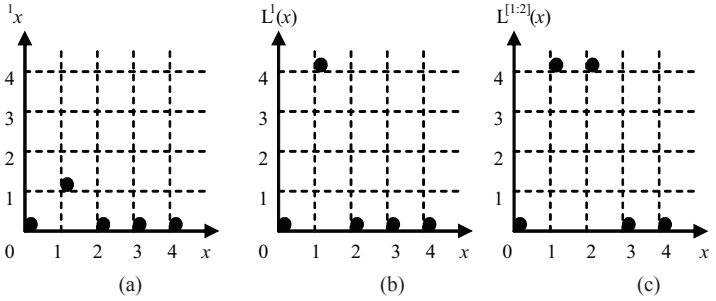


Fig. 3: An example of different types of literals over an arbitrary five-radix logic: (a) 1-Reduced Post Literal (1-RPL), (b) Post Literal (PL), and (c) Window Literal (WL).

the zero, first and second shifts of the variable x respectively (i.e., $x = x+0$, $x' = x+1$ and $x'' = x+2$, respectively), and x can take any value in the set $\{0,1,2\}$. We chose to represent the 1-Reduced Post Literals in terms of shifts and powers, among others, because of the ease of the implementation of powers of shifted variables in hardware such as in Universal Logic Modules (ULMs) [2]. Analogously to the binary and ternary cases, quaternary Shannon expansion over GF(4) for a function with single variable is:

$$f = {}^0x f_0 + {}^1x f_1 + {}^2x f_2 + {}^3x f_3, \tag{6}$$

where f_0 is the cofactor of f with respect to variable x of value 0, f_1 is the cofactor of f with respect to variable x of value 1, f_2 is the cofactor of f with respect to variable x of value 2, and f_3 is the cofactor of f with respect to variable x of value 3.

Example 2. *Let $f(x_1,x_2) = x_1''x_2 + x_2'''x_1$. By using Figure (1), the quaternary truth vector in the variable order $\{x_1,x_2\}$ is $F = [0,3,1,2,2,1,3,0,3,0,2,1,1,2,0,3]^T$. Utilizing Equation (6), one obtains the following GF(4) Shannon expansion for f:*

$$\begin{aligned} f =& 2 \cdot {}^0x_1\,{}^1x_2 + 3 \cdot {}^0x_1\,{}^2x_2 + {}^0x_1\,{}^3x_2 + 3 \cdot {}^1x_1\,{}^0x_2 + {}^1x_1\,{}^1x_2 + 2 \cdot {}^1x_1\,{}^3x_2 \\ &+ {}^2x_1\,{}^0x_2 + 3 \cdot {}^2x_1\,{}^1x_2 + 2 \cdot {}^2x_1\,{}^2x_2 + 2 \cdot {}^3x_1\,{}^0x_2 + {}^3x_1\,{}^2x_2 + 3 \cdot {}^3x_1\,{}^3x_2. \end{aligned}$$

Using the axioms of GF(4) that are manifested in the operators shown in Figure 1, the 1-RPL defined in Equation (5) is related to the shifts of variables over GF(4) in terms of powers [2-5] as follows:

$$^{0}x = x^3 + 1, \tag{7}$$
$$^{0}x = x' + (x')^2 + (x')^3, \tag{8}$$
$$^{0}x = 3(x'') + 2(x'')^2 + (x'')^3, \tag{9}$$
$$^{0}x = 2(x''') + 3(x''')^2 + (x''')^3, \tag{10}$$
$$^{1}x = x + (x)^2 + (x)^3, \tag{11}$$
$$^{1}x = (x')^3 + 1, \tag{12}$$
$$^{1}x = 2(x'') + 3(x'')^2 + (x'')^3, \tag{13}$$
$$^{1}x = 3(x''') + 2(x''')^2 + (x''')^3, \tag{14}$$
$$^{2}x = 3(x) + 2(x)^2 + (x)^3, \tag{15}$$
$$^{2}x = 2(x') + 3(x')^2 + (x')^3, \tag{16}$$
$$^{2}x = (x'')^3 + 1, \tag{17}$$
$$^{2}x = x''' + (x''')^2 + (x''')^3, \tag{18}$$
$$^{3}x = 2(x) + 3(x)^2 + (x)^3, \tag{19}$$
$$^{3}x = 3(x') + 2(x')^2 + (x')^3, \tag{20}$$
$$^{3}x = x'' + (x'')^2 + (x'')^3, \tag{21}$$
$$^{3}x = (x''')^3 + 1, \tag{22}$$

where $\{ {}^{0}x, {}^{1}x, {}^{2}x, {}^{3}x\}$ are the zero, first, second and third polarities of the 1-RPL, respectively. Also, $\{x,x',x'',x'''\}$ are the zero, first, second and third shifts (inversions) of the variable x respectively, and variable x can take any value of the set $\{0,1,2,3\}$. Analogous to the ternary case, we chose to represent the 1-RPL in terms of shifts and powers, among others, because of the ease of the implementation of powers of shifted variables in hardware. After the substitution of Equations (7) - (22) in Equation (6), and after the rearrangement and reduction of terms according to the GF(4) operations in Figure 1, one obtains:

$$f = 1 \cdot f_0 + x(f_1 + 3f_2 + 2f_3) + (x)^2(f_1 + 2f_2 + 3f_3) + (x)^3(f_0 + f_1 + f_2 + f_3), \tag{23}$$
$$f = 1 \cdot f_1 + (x')(f_0 + 2f_2 + 3f_3) + (x')^2(f_0 + 3f_2 + 2f_3) + (x')^3(f_0 + f_1 + f_2 + f_3), \tag{24}$$
$$f = 1 \cdot f_2 + (x'')(3f_0 + 2f_1 + f_3) + (x'')^2(2f_0 + 3f_1 + f_3) + (x'')^3(f_0 + f_1 + f_2 + f_3), \tag{25}$$
$$f = 1 \cdot f_3 + (x''')(f_2 + 3f_1 + 2f_0) + (x''')^2(f_2 + 2f_1 + 3f_0) + (x''')^3(f_0 + f_1 + f_2 + f_3). \tag{26}$$

Equations (6) and (23) - (26) are the 1-RPL quaternary Shannon (S) and Davio $(\mathrm{D}_0,\mathrm{D}_1,\mathrm{D}_2,\mathrm{D}_3\}$ expansions for a single variable, respectively. These Equations can be re-written in the following matrix-based convolution-like forms, respectively:

$$f = \begin{bmatrix} {}^0x & {}^1x & {}^2x & {}^3x \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}, \tag{27}$$

$$f = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 2 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}, \tag{28}$$

$$f = \begin{bmatrix} 1 & x' & (x')^2 & (x')^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}, \tag{29}$$

$$f = \begin{bmatrix} 1 & x'' & (x'')^2 & (x'')^3 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 3 & 2 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}, \tag{30}$$

$$f = \begin{bmatrix} 1 & x''' & (x''')^2 & (x''')^3 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 3 & 1 & 0 \\ 3 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}. \tag{31}$$

One can observe that Equations (27) - (31) are expansions for a single variable. Yet, these canonical expressions can be generated for arbitrary number of variables N using the Kronecker (tensor) product. This can be expressed formally as in the following discrete convolution-like forms for Shannon (S), and Davio (D_0, D_1, D_2 and D_3) expressions, respectively [2, 32]:

$$f = \bigotimes_{i=1}^{N} \begin{bmatrix} {}^0x_i & {}^1x_i & {}^2x_i & {}^3x_i \end{bmatrix} \bigotimes_{i=1}^{N} [\mathrm{S}][\vec{F}], \tag{32}$$

$$f = \bigotimes_{i=1}^{N} \begin{bmatrix} 1 & x_i & x_i^2 & x_i^3 \end{bmatrix} \bigotimes_{i=1}^{N} [\mathrm{D}_0][\vec{F}], \tag{33}$$

$$f = \bigotimes_{i=1}^{N} \begin{bmatrix} 1 & x_i' & (x_i')^2 & (x_i')^3 \end{bmatrix} \bigotimes_{i=1}^{N} [\mathrm{D}_1][\vec{F}], \tag{34}$$

$$f = \bigotimes_{i=1}^{N} \begin{bmatrix} 1 & x_i'' & (x_i'')^2 & (x_i'')^3 \end{bmatrix} \bigotimes_{i=1}^{N} [\mathrm{D}_2][\vec{F}], \tag{35}$$

$$f = \bigotimes_{i=1}^{N} \begin{bmatrix} 1 & x_i''' & (x_i''')^2 & (x_i''')^3 \end{bmatrix} \bigotimes_{i=1}^{N} [\mathrm{D}_3][\vec{F}]. \tag{36}$$

The following section utilizes the presented GF(4) spectral-based functional decompositions for the synthesis of decision trees. In this spectral interpretation of decision trees,

different decision trees can be defined by using different decomposition forms which are specified by the corresponding transform matrices and multi-valued literals. The utilized decision trees can therefore be viewed as graphical representations of functional expressions where different trees produce different functional expressions, and by counting the number of possible different trees, that are derived by assigning different decomposition rules to their nodes, we can count the number of possible functional expressions.

3 Quaternary Shannon-Dvaio (S/D) Trees

The basic S, D_0, D_1, D_2 and D_3 quaternary expansions (i.e., flattened forms) introduced previously in Equations (32) - (36) can be represented in quaternary DTs (QuDTs) and the corresponding varieties of reduced quaternary DDs (RQuDDs) according to the corresponding reduction rules. For one variable (i.e., one level of the DT), Figures 4(a) - 4(e) represent the expansion nodes for $\{S, D_0, D_1, D_2, D_3\}$, respectively, and the notation in Figure 4(f) means that $\underline{x}$ corresponds to the four possible shifts of the variable x as:

$$\underline{x} \in \{x, x', x'', x'''\}, \text{ over GF(4).} \tag{37}$$

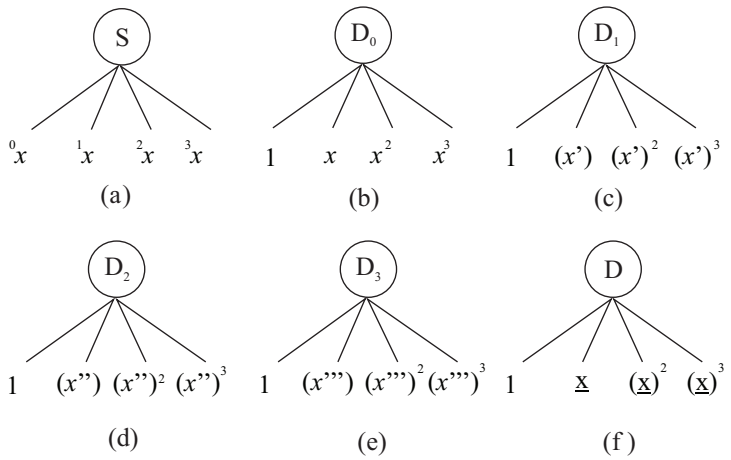


Fig. 4: Quaternary decision nodes: (a) Shannon in Eq. (32), (b) Davio_0 in Eq. (33), (c) Davio_1 in Eq. (34), (d) Davio_2 in Eq. (35), (e) Davio_3 in Eq. (36), and (f) generalized quaternary Davio defined in Eq. (37).

Utilizing the two nodes defined for quaternary Shannon in Figure 4(a) and quaternary generalized Davio in Figure 4(f), and analogously to the binary and ternary cases, one can obtain the quaternary Shannon-Davio (S/D) trees for two variables (cf. Figure 5), where general family called Inclusive Forms (IFs) is obtained as flattened expressions generated by these S/D trees. For example, the corresponding S/D trees for IFs of two variables can be generated for variable order $\{a, b\}$ and for variable order $\{b, a\}$ as well.

The number of these S/D trees per variable order is $2^{(4+1)} = 32$, where the number of QIFs per S/D tree will be later derived in Section 4 in two different ways; the first method

is by using the general formula for an arbitrary number of variables over GF(4) and the second method is performed by using the general formula for any radix. The number of all possible forms is important because it can be used as an upper-bound parameter in a search heuristic that searches for a minimum GFSOP expression using the corresponding S/D trees. Example 3 illustrates some of the quaternary S/D trees and some of the quaternary trees they produce. The numbers on top of S/D trees in Figures 5(a) and 6(a) are the numbers of total QIFs (i.e., total number of quaternary trees) that are generated.

Example 3. *Utilizing the notation in Equation (37), we obtain, for the S/D trees in Figures 5(a) and 6(a), the corresponding S/D trees in Figures 5(b) - 5(c) and Figures 6(b) - 6(c), respectively.*

From the quaternary S/D trees shown in Figures 5 and 6, by taking any S/D tree, multiplying the two-level cofactors (which are in the QuDT leafs) each by the corresponding path in that QuDT, and next summing all the resulting cubes (terms or products) over GF(4), one obtains the flattened IF form for the function f as a certain GFSOP expression (expansion). For each QuDT in Figures 5(a) and 6(a), there are as many IF forms obtained for the function f as the number of all possible permutations of the polarities of the variables in the second level branches of each QuDT.

4 Count of The Number of S/D Inclusive Forms Over GF(P^K) and the New $IF_{N,2}$ Triangles

This section provides the count for the numbers of Inclusive Forms, which are flattened expressions generated by the corresponding S/D trees, where these counts can be used as numerical parameters for upper-bounds in search heuristics that search for minimum GFSOP expressions.

Theorem 1. *For GF(3) and N variables, the total number of ternary IFs (TIFs) per variable order is:*

$$\#\text{TIF}s = \sum_{k_1=0}^{(3)^{N-1}} \sum_{k_2=0}^{(3)^{N-2}} \sum_{k_3=0}^{(3)^{N-3}} \cdots \sum_{k_N=0}^{(3)^0} \left\{ \left[\frac{3^{(N-1)}!}{(3^{(N-1)}-k_1)!k_1!} \frac{3^{(N-2)}!}{(3^{(N-2)}-k_2)!k_2!} \right.\right. \\ \left. \frac{3^{(N-3)}!}{(3^{(N-3)}-k_3)!k_3!} \cdots \frac{3^{(0)}!}{(3^{(0)}-k_N)!k_N!} \right] \\ \left.\left[(3^{2(3)^0})^{k_1} (3^{2(3)^1})^{k_2} (3^{2(3)^2})^{k_3} \cdots (3^{2(3)^{N-1}})^{k_N} \right] \right\}. \tag{38}$$

Proof. The following is the derivation of Equation (38) to calculate #TIFs per variable order. The total number of nodes for any GF(3) tree with N levels (N variables) equals:

$$\sum_{k=0}^{N-1} (3)^k. \tag{39}$$

For any S-type node there is only one type of nodes as the branches have the possibility of single value each. Yet, for D-type node there are N possible types of nodes where N is the

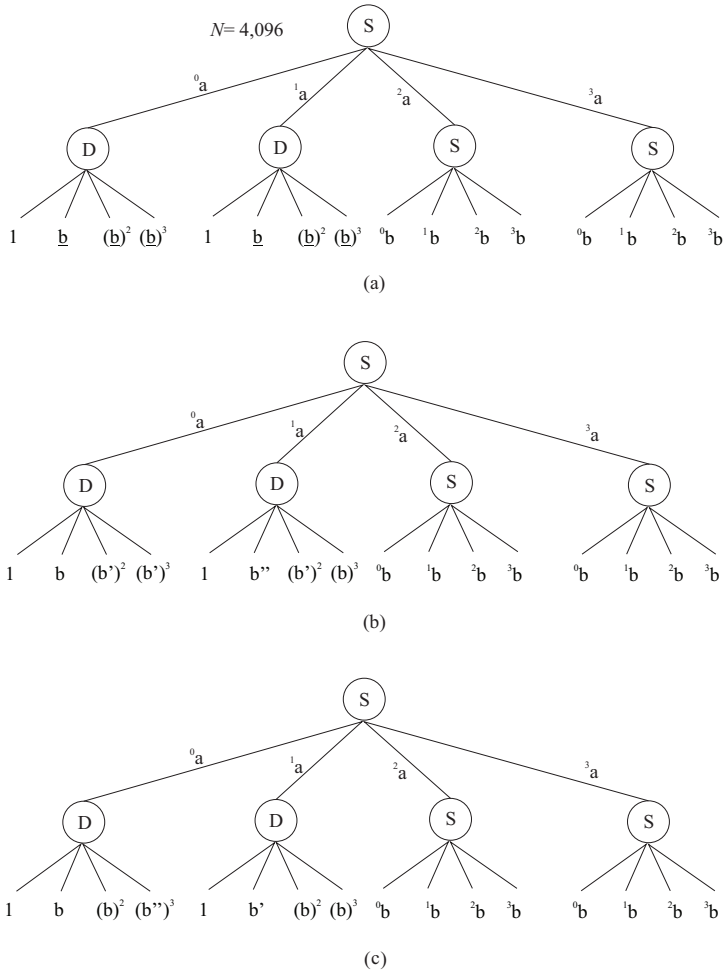


Fig. 5: Examples of S/D trees: (a) quaternary S/D tree for two variables of order $\{a, b\}$ with three Shannon nodes and two generalized Davio nodes, and (b) - (c) some of the quaternary trees that it generates.

number of variables which is equal to the number of levels. The highest possible number of forms for the D-type node is when the D-type node exists in the first (highest) level, and the lowest possible number of forms for the D-type node is when the D-type node exists in the N-level (lowest level). Therefore, for certain number M of S-type nodes the following equation describes the number of the D-type nodes for N variables:

$$\#S = M \Rightarrow \#D = [\sum_{k=o}^{N-1} (3)^k - M]. \tag{40}$$

It can be shown that for GF(3) (i.e., ternary decision tree (TDT)) and N-levels (N-variables), the general formulas that count the number of D-type nodes, and the number of all possible

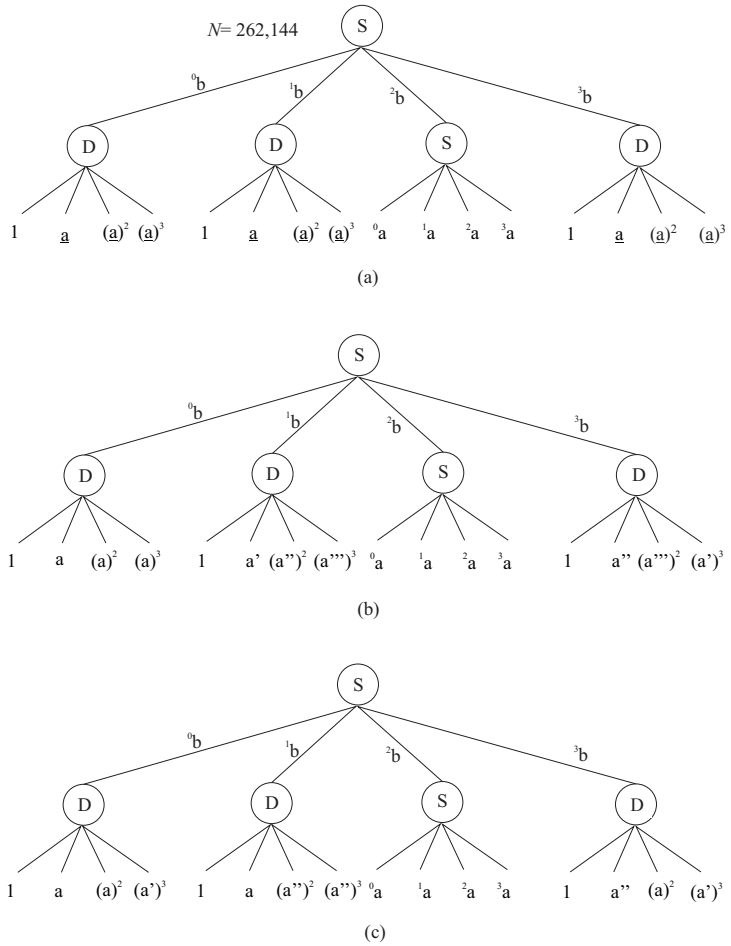


Fig. 6: Examples of S/D trees: (a) quaternary S/D tree for two variables of order $\{b,a\}$ with two Shannon nodes and three generalized Davio nodes, and (b) - (c) some of the quaternary trees that it generates.

forms for the D-type node in the k level of the N-level TDT are:

$$\#\mathrm{D}_k = (3)^{(K-1)}, \tag{41}$$

$$|\mathrm{D}_k|_{\text{per node}} = (3)^{2(3)^{(N-K)}}, \tag{42}$$

where $\#\mathrm{D}_k$ is the number of D-type nodes in k level, and $|\mathrm{D}_k|$ is the number of all possible forms for the D-type node in the k level. Let us define S/D tree category to be the S/D trees that have in common the same number of S-type nodes and same number of

D-type nodes within the same variable order. Also, define:

$$\Psi \equiv \text{ number of variable orders,} \tag{43}$$

$$\Omega \equiv \text{ number of S/D tree categories per variable order,} \tag{44}$$

$$\phi \equiv \text{ number of S/D trees per category,} \tag{45}$$

$$\Phi \equiv \text{ number of TIFs per variable order.} \tag{46}$$

From Equations (39) - (42), and using some elementary count rules, we can derive by mathematical induction the following general formulas for N being the number of variables:

$$\Psi = N!, \tag{47}$$

$$\Omega = \sum_{k=0}^{N-1} (3)^k + 1, \tag{48}$$

$$\phi = \frac{[\sum_{k=0}^{N-1}(3)^k]!}{[\sum_{k=0}^{N-1}(3)^k - k]!k!}, \text{ where } k = 0,1,2,3,\ldots,\sum_{k=0}^{N-1}(3)^k, \tag{49}$$

$$\begin{aligned}\Phi = \sum_{k_1=0}^{(3)^{N-1}} \sum_{k_2=0}^{(3)^{N-2}} \sum_{k_3=0}^{(3)^{N-3}} \cdots \sum_{k_N=0}^{(3)^0} \Bigg\{ \Bigg[\frac{3^{(N-1)}!}{(3^{(N-1)}-k_1)!k_1!} \frac{3^{(N-2)}!}{(3^{(N-2)}-k_2)!k_2!} \\ \frac{3^{(N-3)}!}{(3^{(N-3)}-k_3)!k_3!} \cdots \frac{3^{(0)}!}{(3^{(0)}-k_N)!k_N!} \Bigg] \Bigg[(3^{2(3)^0})^{k_1} (3^{2(3)^1})^{k_2} (3^{2(3)^2})^{k_3} \cdots \\ (3^{2(3)^{(N-1)}})^{k_N} \Bigg] \Bigg\}.\end{aligned} \tag{50}$$

□

From Equations (47) - (50), it can be noticed that the total number of TIFs for all variable orders is equal to [N!][#TIFs per order].

Example 4. *For number of variables equal to two* $(N = 2)$*,* Φ *reduces to:*

$$\Phi = \sum_{k_1=0}^{(3)^1} \sum_{k_2=0}^{1} \left\{ \frac{3^1!}{(3^1-k_1)!k_1!} \frac{3^0!}{(3^0-k_2)!k_2!} (3^{2(3)^0})^{k_1} (3^{2(3)^1})^{k_2} \right\}$$

$$\begin{aligned}\Phi =& \Phi|_{k_1=0,k_2=0} + \Phi|_{k_1=1,k_2=0} + \Phi|_{k_1=2,k_2=0} + \Phi|_{k_1=3,k_2=0} \\ &+ \Phi|_{k_1=0,k_2=1} + \Phi|_{k_1=1,k_2=1} + \Phi|_{k_1=2,k_2=1} + \Phi|_{k_1=3,k_2=1} \\ =& \Phi_{00} + \Phi_{10} + \Phi_{20} + \Phi_{30} + \Phi_{01} + \Phi_{11} + \Phi_{21} + \Phi_{31} \\ =& 1 + 27 + 243 + 729 + 729 + 19683 + 177147 + 531441 = 730,000.\end{aligned}$$

Utilizing multi-valued map representation, there are $N^{\#\text{Minterms}}$ different functions for N-valued input-output logic. Therefore, for ternary logic, there are $3^9 = 19,683$ different ternary functions of two variables, and 730,000 ternary Inclusive Forms generated by the S/D trees. Thus, on the average every function of two variables can be realized in approximately 37 ways.

Theorem 2. *For GF(4) and N variables, the total number of quaternary IFs (QIFs) per variable order is:*

$$\begin{aligned}\#\text{QIFs} = \Phi = \sum_{k_1=0}^{(4)^{N-1}} \sum_{k_2=0}^{(4)^{N-2}} \cdots \sum_{k_N=0}^{(4)^0} \Bigg\{ & \frac{4^{(N-1)}!}{(4^{(N-1)}-k_1)!k_1!} \frac{4^{(N-2)}!}{(4^{(N-2)}-k_2)!k_2!} \cdots \\ & \frac{4^{(0)}!}{(4^{(0)}-k_N)!k_N!} (4^{3(4)^0})^{k_1} (4^{3(4)^1})^{k_2} (4^{3(4)^2})^{k_3} \cdots (4^{3(4)^{(N-1)}})^{k_N} \Bigg\}.\end{aligned} \tag{51}$$

Proof. A general proof that includes GF(4) as special case will be provided later in this Section. □

The extension of the concept of S/D trees to higher radices of Galois fields (i.e., higher than four) is a systematic and direct process that follows the same method developed for the ternary case and the quaternary case. The following example demonstrates the counts of QIFs using Theorem 2.

Example 5. *For number of variables equal to two (N = 2), Equation (51) reduces to:*

$$\begin{aligned}\Phi &= \sum_{k_1=0}^{(4)^1} \sum_{k_2=0}^{(4)^0} \left\{ \frac{4^{(1)}!}{(4^{(1)}-k_1)!k_1!} \frac{4^{(0)}!}{(4^{(0)}-k_2)!k_2!} (4^{3(4)^0})^{k_1} (4^{3(4)^1})^{k_2} \right\} \\ &= \Phi|_{k_1=0,k_2=0} + \Phi|_{k_1=1,k_2=0} + \Phi|_{k_1=2,k_2=0} + \Phi|_{k_1=3,k_2=0} + \Phi|_{k_1=4,k_2=0} \\ &\quad + \Phi|_{k_1=0,k_2=1} + \Phi|_{k_1=1,k_2=1} + \Phi|_{k_1=2,k_2=1} + \Phi|_{k_1=3,k_2=1} + \Phi|_{k_1=4,k_2=1} \\ &= \Phi_{00} + \Phi_{10} + \Phi_{20} + \Phi_{30} + \Phi_{40} + \Phi_{01} + \Phi_{11} + \Phi_{21} + \Phi_{31} + \Phi_{41} \\ &= 1 + 256 + 24,576 + 1,048,576 + 16,777,216 + 16,777,216 + 4,294,967,296 \\ &\quad + 412,316,860,416 + 1.75921860444 \times 10^{13} + 2.81477976711 \times 10^{14} \\ &= 2.99483809211 \times 10^{14}.\end{aligned}$$

Utilizing multi-valued map representation, we can easily prove that there are $4^{16} = 4,294,967,296$ quaternary functions of two variables, and $2.99483809211 \times 10^{14}$ quaternary Inclusive Forms generated by the S/D trees. Thus, on the average, every function of two variables can be synthesized (realized) in approximately 69,729 ways. This high number of realizations means that most functions of two variables are realized with less than five expansions, and all functions with at most five expansions.

4.1 General Formula to Compute the Number of IFs for an Arbitrary Variable Number and Arbitrary Galois Radix GF(p^k)

Although the S/D trees and Inclusive Forms that were developed are for GF(4), the same concept can be directly and systematically extended to the case of n radix of Galois fields and N variables. Theorem 3 provides the total number of IFs per variable order for N variables (i.e., N decision tree levels) and n radix of any arbitrary algebraic field, including GF(p^k) where p is a prime number and k is a natural number ≥ 1. The generality of Theorem 3 comes from the fact that algebraic structures specify the type of operations

(e.g., addition and multiplication operations) in the functional expansions but do not specify the counts which are an intrinsic property of the tree structure and are independent of the algebraic operations performed. Thus, Theorem 3 is valid, among others, for Galois fields of arbitrary radix.

Theorem 3. *The total number of Inclusive Forms for N variables and n-radix Galois field logic is equal to:*

$$\#nIFs = \Phi_{n,N} = \sum_{k_1=0}^{(n)^{N-1}} \sum_{k_2=0}^{(n)^{N-2}} \cdots \sum_{k_N=0}^{(n)^0} \left\{ \frac{n^{(N-1)}!}{(n^{(N-1)}-k_1)!k_1!} \frac{n^{(N-2)}!}{(n^{(N-2)}-k_2)!k_2!} \cdots \right.$$
$$\left. \frac{n^{(0)}!}{(n^{(0)}-k_N)!k_N!} (n^{(n-1)(n)^0})^{k_1} (n^{(n-1)(n)^1})^{k_2} (n^{(n-1)(n)^2})^{k_3} \cdots (n^{(n-1)(n)^{N-1}})^{k_N} \right\}. \tag{52}$$

Proof. The following is the derivation of the general Equation (52) to calculate the number of IFs per variable order. The total number of nodes for any GF(n) tree with N levels (i.e., N variables) equals to:

$$\sum_{k=0}^{N-1} (n)^k. \tag{53}$$

For any S-type (i.e., Shannon type) node there is only one type of nodes as the branches of the Shannon node have the possibility of single value each. Yet, for D-type (i.e., Davio type) node there are N possible types of nodes where N is the number of variables which is equal to the number of levels. The highest possible number of forms for the D-type node exists when the Davio node exists in the first (highest) level, and the lowest possible number of forms for the D-type node is when the Davio node exists in the N-level (lowest level). Therefore, for certain number M of S-type nodes the following formula describes the number of the D-type nodes for N variables:

$$\#\mathrm{S} = M \Rightarrow \#\mathrm{D} = \sum_{k=0}^{N-1} (n)^k - M. \tag{54}$$

It can be shown that for GF(n) (n-ary decision tree with N-levels, i.e., N variables), the general formulas that count the number of D-type nodes, and the number of all possible forms for the D-type node in the k level (where k is less than or equal the total number of levels N) are, respectively:

$$\#D_k = (n)^{k-1}, \tag{55}$$

$$|D_k| = (n)^{(n-1)(n)^{(N-K)}}, \tag{56}$$

where $\#\mathrm{D}_k$ is the number of D-type nodes in the k level and $|\mathrm{D}_k|$ is the number of all possible forms (per node) for the D-type node in the k level. Let us define the S/D tree category to be the S/D trees that have in common the same number of S-type nodes and the same number of D-type nodes within the same variable order. Let us define the following

entities for n radix Galois field and N variables (i.e., N decision tree levels):

$$\Psi_{n,N} \equiv \text{ number of variable orders,} \tag{57}$$

$$\Omega_{n,N} \equiv \text{ number of S/D tree categories per variable order,} \tag{58}$$

$$\phi_{n,N} \equiv \text{ number of S/D trees per category,} \tag{59}$$

$$\Phi_{n,N} \equiv \text{ number of IFs per variable order.} \tag{60}$$

From the previous Equations, and using elementary count rules, one can derive by mathematical induction the following general formulas for N being the number of variables and n being the field radix:

$$\Psi_{n,N} = N!, \tag{61}$$

$$\Omega_{n,N} = \sum_{k=0}^{N-1} (n)^k + 1, \tag{62}$$

$$\Phi_{n,N} = \frac{[\sum_{k=0}^{N-1} (n)^k]!}{[\sum_{k=0}^{N-1} (n)^k - k]!k!}, \text{ where } k = 0,1,2,3,\ldots,N-1, \tag{63}$$

$$\Phi_{n,N} = \sum_{k_1=0}^{(n)^{N-1}} \sum_{k_2=0}^{(n)^{N-2}} \cdots \sum_{k_N=0}^{(n)^0} \left\{ \frac{n^{(N-1)}!}{(n^{(N-1)} - k_1)!k_1!} \frac{n^{(N-2)}!}{(n^{(N-2)} - k_2)!k_2!} \cdots \frac{n^{(0)}!}{(n^{(0)} - k_N)!k_N!} \right. \\ \left. (n^{(n-1)(n)^0})^{k_1} (n^{(n-1)(n)^1})^{k_2} \cdots (n^{(n-1)(n)^{(N-1)}})^{k_N} \right\}. \tag{64}$$

□

One can note that the formula in Equation (52) used to obtain the total number of Inclusive Forms for N variables and n radix of Galois field is a very general formula that includes the ternary case in Equation (38) and the quaternary case in Equation (51) as special cases. Numerical counting results that are obtained from Equation (52) can be used in search heuristics as numerical bounds that could be incorporated into efficient search of S/D trees in order to obtain minimal GFSOP forms for specific multi-valued logic functions. Since such search for minimal forms is already a difficult problem in two-valued logic - for example using binary S/D trees - especially when the number of variables is large, the search for minimal GFSOP forms in multi-valued Galois logic will be very difficult. Thus, further numerical evaluations have to be conducted in order to estimate the usefulness of the utilizations of the numerical bounds obtained from Equation (52) in such extended multi-valued search heuristics.

Example 6. *The number of QIFs over GF(4) for two variables (i.e., $N = 2$ and $n = 4$) is:*

$$\begin{aligned}\Phi_{4,2} &= \sum_{k_1=0}^{(4)^{2-1}} \sum_{k_2=0}^{(4)^{2-2}} \left\{ \frac{4^{(2-1)}!}{(4^{(2-1)}-k_1)!k_1!} \frac{4^{(2-2)}!}{(4^{(2-2)}-k_2)!k_2!} (4^{(4-1)(4)^0})^{k_1} (4^{(4-1)(4)^1})^{k_2} \right\}, \\ &= \Phi_{00|4,2} + \Phi_{10|4,2} + \Phi_{20|4,2} + \Phi_{30|4,2} + \Phi_{40|4,2} + \Phi_{01|4,2} + \Phi_{11|4,2} + \Phi_{21|4,2} \\ &\quad + \Phi_{31|4,2} + \Phi_{41|4,2} \\ &= 1 + 256 + 24,576 + 1,048,576 + 16,777,216 + 16,777,216 + 4,294,967,296 \\ &\quad + 412,316,860,416 + 1.75921860444 \times 10^{13} + 2.81477976711 \times 10^{14} \\ &= 2.99483809211 \times 10^{14}.\end{aligned}$$

Corollary 1. *From Equation (52), the count of IFs for N variables and second radix is:*

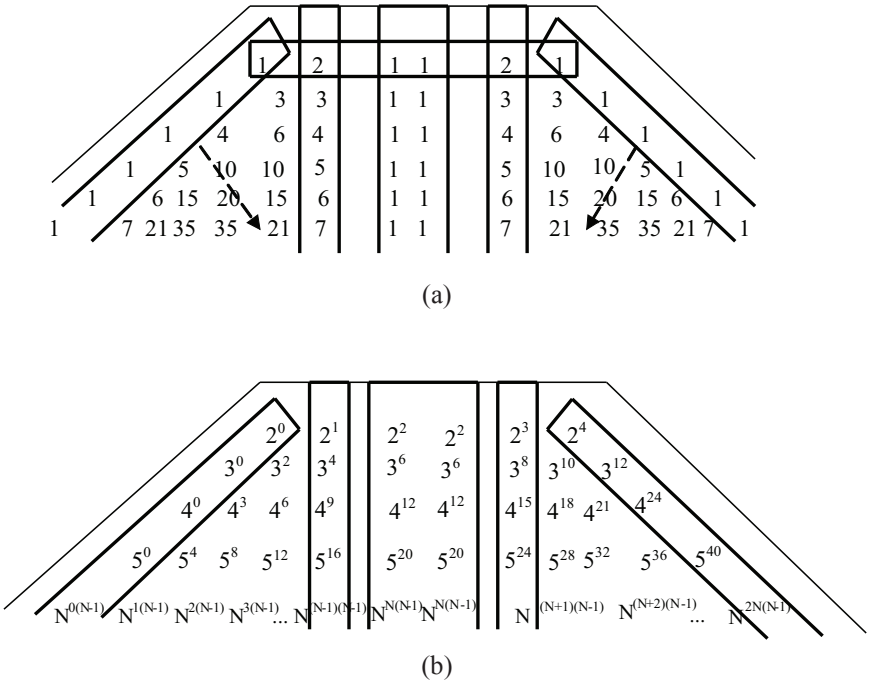
$$\begin{aligned}\prod_{k=0}^{n-1} (1 + 2^{2^{n-k-1}})^{2^k} = \sum_{k_1=0}^{(2)^{N-1}} \sum_{k_2=0}^{(2)^{N-2}} \cdots \sum_{k_N=0}^{(2)^0} \left\{ \frac{2^{(N-1)}!}{(2^{(N-1)}-k_1)!k_1!} \frac{2^{(N-2)}!}{(2^{(N-2)}-k_2)!k_2!} \cdots \right. \\ \left. \frac{2^{(0)}!}{(2^{(0)}-k_N)!k_N!} (2^{(2-1)(2)^0})^{k_1} (2^{(2-1)(2)^1})^{k_2} \cdots (2^{(2-1)(2)^{(N-1)}})^{k_N} \right\}.\end{aligned} \tag{65}$$

As previously mentioned, this enumeration can be useful as a terminating point of minimization algorithms for multi-valued functions. Yet, as shown, the number of combinations is so large that restriction to some particular cases of functional expressions can be more feasible. The following Section introduces a fast method to calculate the number of IFs for an arbitrary Galois field logic for functions with two variables.

4.2 The $IF_{n,2}$ Triangles: Fast Count Calculations of IFs for GF(p^k) and Two-Variable Functions

The count of the number of IFs can be important in many applications, especially in providing upper numerical boundaries for efficient search of a minimum GFSOP. Calculating the numbers of Inclusive Forms can be very time consuming due to the time required to perform the mathematical operations in the general Equation (52). This is why a fast method to generate the number of IFs is needed. Because functions with two variables find an important application such as in Universal Logic Modules (ULMs) for pairs of control variables that generalize Shannon and Davio expansion modules [2], and since two-variable functions are attractive in logic synthesis since many functional decomposition methods exist that produce two control inputs for primitive cells in a standard library of standard cells such as in a multiplexer with two address lines, Theorem 4 provides a fast computational method to calculate the number of IFs over an arbitrary radix of Galois field GF(p^k) for two-variable functions (i.e., $N = 2$).

Theorem 4. *The following $IF_{n,2}$ Triangles provide a fast computational method to calculate the number of IFs over an arbitrary n radix of Galois field GF(p^k) for two-variable functions ($N = 2$).*



(a)

(b)

Fig. 7: The $IF_{n,2}$ Triangles: (a) triangle of coefficients, and (b) triangle of values for fast calculation of the number of Inclusive Forms for arbitrary radix Galois field and functions of two-input variables.

Proof. The proof follows directly from mathematical induction of the number of IFs over GF(p^k) for two- variable functions. This is deduced from the general Equation (52); if the $IF_{n,2}$ Triangles are valid for $n = q$ then they will be also valid for $n = q + 1$, for $n = p^k$ where p is a prime number and $k \geq 1$. □

These triangles are important because the count complexity using Equation (52) for high dimensions is very high, and thus the ability of a computer to compute the counts for number of variables greater than five in a reasonable amount of time becomes difficult. Consequently, the $IF_{n,2}$ triangles provide an alternative numerical and geometrical pattern of computing. It can be observed that the $IF_{n,2}$ triangle of coefficients possesses a close similarity to the well-known Pascal Triangle. This occurs as follows: if one omits the first two rows of the Pascal Triangle and duplicates each row into another horizontally adjacent row, the $IF_{n,2}$ triangle of coefficients will be obtained. This observation helps in creating algorithms that generates the $IF_{n,2}$ triangle of coefficients since many efficient algorithms exist to generate the Pascal Triangle.

Example 7. *Utilizing $IF_{n,2}$ Triangles from Figure 7, one calculates the following number of Inclusive Forms for GF(2), GF(3) and GF(4) for two variables, where the results are*

identical to those obtained previously:

$$\Phi_{2,2} = 1\cdot 2^0 + 2\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^2 + 2\cdot 2^3 + 1\cdot 2^4 = 1+4+4+4+16+16 = 45.$$

$$\Phi_{3,2} = 1\cdot 3^0 + 3\cdot 3^2 + 3\cdot 3^4 + 1\cdot 3^6 + 1\cdot 3^6 + 3\cdot 3^8 + 3\cdot 3^{10} + 1\cdot 3^{12} = 730,000.$$

$$\Phi_{4,2} = 1\cdot 4^0 + 4\cdot 4^3 + 6\cdot 4^6 + 4\cdot 4^9 + 1\cdot 4^{12} + 1\cdot 4^{12} + 4\cdot 4^{15} + 6\cdot 4^{18} + 4\cdot 4^{21} + 1\cdot 4^{24}$$
$$= 2.99483809211\times 10^{14}.$$

The $\mathrm{IF}_{n,2}$ Triangles, for N is the number of variables, possess the following interesting properties:

1. Number of positions (elements) in each row of the triangles in Figure 7 are even starting from six.
2. Sum of elements in each row in Figure 7(a) equals to the number of S/D trees per variable order.
3. Triangle in Figure 7(a) possesses even symmetry around an imaginary vertical axis in the middle.
4. The minimum number of columns required to generate the whole triangle in Figure 7(a) is equal to three due to even symmetry: one wing, one column neighbor to the middle column and one middle column.
5. The triangle in Figure 7(a) can be generated by the process of "Shift Diagonally and Add Diagonally" (SDAAD): shift the left wing diagonally from west to southeast direction and add two numbers diagonally from east to southwest direction, and shift the right wing diagonally from east to southwest direction and add two numbers diagonally from west to southeast direction.
6. The difference in powers in the triangle in Figure 7(b) per row element is $(N-1)$.
7. The first number in each row of the triangle in Figure 7(b) is N^0 and the last number per row is $N^{2N(N-1)}$.
8. The middle two numbers in each row of the triangle in Figure 7(b) are always equal to $N^{N(N-1)}$.

5 Conclusions and Future Work

Trees for generalized Shannon-Davio (S/D) expansions over quaternary Galois radix is presented, and the corresponding count for the number of Inclusive Forms (IFs) per variable order for arbitrary Galois radix and arbitrary number of variables is introduced. Also, the $\mathrm{IF}_{n,2}$ Triangles as a new fast computational method to count the number of IFs for an arbitrary Galois radix and functions of two variables is introduced. Since Galois field of quaternary radix has some interesting properties including its implementation utilizing the well-established two-valued logic synthesis methods, the extension of the S/D trees to GF(4) is presented. In addition, the form of S/D trees is a general concept that can be used in applications for the generation of new diagrams and canonical forms, and in the Sum-Of-Product (SOP) minimization where S/D trees can be utilized for generating forms that include minimum Galois Field Sum-of-Products (GFSOP) circuits for binary and *m*-ary radices.

Future work will investigate using other complex types of literals such as the presented Post literal (PL) and window literal (WL) to expand upon and consequently construct the corresponding new S/D trees. The utilization of the results from this research to create an efficient GFSOP minimizer for synthesis applications within the spaces of classical and reversible logic will also be conducted.

Acknowledgement: *This research was performed during sabbatical leave in 2015-2016 granted from The University of Jordan and spent at Philadelphia University.*

REFERENCES

[1] S. B. Akers, "Binary Decision Diagrams," IEEE Trans. Comp., Vol. C-27, No. 6, pp. 509-516, June 1978.

[2] A. N. Al-Rabadi, Reversible Logic Synthesis: From Fundamentals to Quantum Computing, Springer-Verlag, 2004.

[3] A. N. Al-Rabadi, "Reversible Fast Permutation Transforms for Quantum Circuit Synthesis," Proc. IEEE Int. Symposium on Multiple-Valued Logic (ISMVL), Toronto, 2004, pp. 81-86.

[4] A. N. Al-Rabadi, "Quantum Circuit Synthesis Using Classes of GF(3) Reversible Fast Spectral Transforms," Proc. IEEE Int. Symposium on Multiple-Valued Logic (ISMVL), Toronto, 2004, pp. 87-93.

[5] A. N. Al-Rabadi, "Quantum Logic Circuit Design of Many-Valued Galois Reversible Expansions and Fast Transforms," J. Circuits, Systems, and Computers, World Scientific, Singapore, Vol. 16, No. 5, pp. 641 - 671, 2007.

[6] A. N. Al-Rabadi, "Representations, Operations, and Applications of Switching Circuits in the Reversible and Quantum Spaces," Facta Universitatis (FU) - Electronics and Energetics, Vol. 20, No. 3, pp. 507 - 539, 2007.

[7] R. E. Bryant, "Graph-based Algorithms for Boolean Functions Manipulation," IEEE Trans. on Comp., Vol. C-35, No.8, pp. 667-691, 1986.

[8] M. Cohn, Switching Function Canonical Form over Integer Fields, Ph.D. Dissertation, Harvard University, 1960.

[9] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. A. Perkowski, "Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams," Proc. DAC, 1994, pp. 415-419.

[10] M. Escobar and F. Somenzi, "Synthesis of AND/EXOR Expressions via Satisfiability," Proc. Reed-Muller, 1995, pp. 80-87.

[11] B. J. Falkowski and S. Rahardja, "Classification and Properties of Fast Linearly Independent Logic Transformations," IEEE Trans. on Circuits and Systems-II, Vol. 44, No. 8, pp. 646-655, August 1997.

[12] B. Falkowski and L.-S. Lim, "Gray Scale Image Compression Based on Multiple-Valued Input Binary Functions, Walsh and Reed-Muller Spectra," Proc. ISMVL, 2000, pp. 279-284.

[13] H. Fujiwara, Logic Testing and Design for Testability, MIT Press, 1985.

[14] D. H. Green, "Families of Reed-Muller Canonical Forms," Int. J. of Electronics, No. 2, pp. 259-280, 1991.

[15] S. Hassoun, T. Sasao, and R. Brayton (editors), Logic Synthesis and Verification, Kluwer Acad. Publishers, 2001.

[16] M. Helliwell and M. A. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller Forms," Proc. Design Automation Conference, 1988, pp. 427-432.

[17] S. L. Hurst, D. M. Miller, and J. C. Muzio, Spectral Techniques in Digital Logic, Academic Press Inc., 1985.

[18] M. G. Karpovski, Finite Orthogonal Series in the Design of Digital Devices, Wiley, 1976.

[19] C. Y. Lee, "Representation of Switching Circuits by Binary Decision Diagrams," Bell Syst. Tech. J., Vol. 38, pp. 985-999, 1959.

[20] C. Moraga, "Ternary Spectral Logic," Proc. ISMVL, pp. 7-12, 1977.

[21] J. C. Muzio and T. Wesselkamper, Multiple-Valued Switching Theory, Adam-Hilger, 1985.

[22] D. K. Pradhan, "Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays," IEEE Trans. Comp., Vol. 27, pp. 181-187, 1978.

[23] D. K. Pradhan, Fault-Tolerant Computing: Theory and Techniques, Vol. I, Prentice-Hall, 1987.

[24] S. M. Reddy, "Easily Testable Realizations of Logic Functions," IEEE Trans. Comp., C-21, pp. 1183-1188, 1972.

[25] T. Sasao (editor), Logic Synthesis and Optimization, Kluwer Academic Publishers, 1993.

[26] T. Sasao, "EXMIN2: A Simplified Algorithm for Exclusive-OR-Sum-Of-Products Expressions for Muliptle-Valued Input Two-Valued Output Functions," IEEE Trans. Computer Aided Design, Vol. 12, No. 5, pp. 621-632, 1993.

[27] T. Sasao and M. Fujita (editors), Representations of Discrete Functions, Kluwer Academic Publishers, 1996.

[28] T. Sasao, "Easily Testable Realizations for Generalized Reed-Muller Expressions," IEEE Trans. Comp., Vol. 46, pp. 709-716, 1997.

[29] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.

[30] N. Song and M. Perkowski, "Minimization of Exclusive Sum of Products Expressions for Multi-Output Multiple-Valued Input Incompletely Specified Functions," IEEE Trans. Computer Aided Design, Vol. 15, No. 4, pp. 385-395, 1996.

[31] R. S. Stanković, "Functional Decision Diagrams for Multiple-Valued Functions," Proc. ISMVL, 1995, pp. 284-289.

[32] R. S. Stankovic, Spectral Transform Decision Diagrams in Simple Questions and Simple Answers, Nauka, 1998.

[33] R. S. Stanković, C. Moraga, and J. T. Astola, "Reed-Muller Expressions in the Previous Decade," Proc. Reed-Muller, Starkville, 2001, pp. 7-26.

[34] B. Steinbach and A. Mishchenko, "A New Approach to Exact ESOP Minimization," Proc. Reed-Muller, Starkville, 2001, pp. 66-81.

[35] S. N. Yanushkevich, Logic Differential Calculus in Multi-Valued Logic Design, Technical Univ. Szczecin, 1998.

[36] I. Zhegalkin, "On the Techniques of Calculating Sentences in Symbolic Logic," Math. Sb., Vol. 34, pp. 9-28, 1927.

[37] I. Zhegalkin, "Arithmetic Representations for Symbolic Logic," Math. Sb., Vol. 35, pp. 311-377, 1928.