Ternary Galois Field Expansions for Reversible Logic and Kronecker Decision Diagrams for Ternary GFSOP Minimization

Mozammel H. A. Khan[$], Marek A. Perkowski[*], and Mujibur R. Khan[$]
[$]*Department of Computer Science and Engineering, East West University, 43 Mohakhali, Dhaka 1212, BANGLADESH. mhakhan@ewubd.edu and mrkhan@ewubd.edu*
[*]*Department of Electrical and Computer Engineering, Portland State University, 1900 SW 4th Avenue, Portland, OR 97201, USA. mperkows@ee.pdx.edu*

Abstract

Ternary Galois Field Sum of Products (TGFSOP) expressions are found to be good choice for ternary reversible, and especially quantum, logic design. In this paper, we propose 16 Ternary Galois Field Expansions (TGFE) and introduce three Ternary Galois Field Decision Diagrams (TGFDD) using the proposed TGFEs, which are useful for reversible and quantum logic design. We also propose a heuristic for creating TGFDDs and a method for flattening the TGFDDs for determining TGFSOP expressions. We provide experimental results to show the effectiveness of the developed methods.

1. Introduction

It has been shown that Galois Field Sum of Products (GFSOP) expressions are a good choice for multiple-valued reversible logic synthesis [1-4]. It has been also shown in [4] that Ternary Galois Field Sum of Products (TGFSOP) expressions are a natural choice for multiple-valued quantum logic synthesis. Such expressions can be either realized directly in quantum cascades or become a starting point of factorization processes leading to factorized cascades [3,4]. Therefore, efficient methods for representing and minimizing TGFSOP expressions are very important.

Multiple-Valued Decision Diagrams (MDD) for many functions over multiple-valued domains are presented in the literature [5-29]. Many of these decision diagrams are based on Reed-Muller like multiple-valued expressions and their related forms. Many others are based on algebraic and finite field structures such as Galois Field. However, because of the requirement of reversible realization of literals, the TGFSOP expressions introduced in [4] require a special form of multiple-valued decision diagrams. To achieve a useful way of determining TGFSOP expressions for a given ternary function, in this paper, we introduce three types of Ternary Galois Field Decision Diagrams (TGFDD), flattening of which directly gives TGFSOP expressions. These diagrams are adaptations of known diagrams for quantum and reversible computing, in which only some literals are physically realizable. For constructing such TGFDDs we propose, in this paper, 16 Ternary Galois Field Expansions (TGFE) that use the reversible literals of the intended TGFSOP expressions. Experimental results show that the TGFDDs produce good quality TGFSOP expressions for many ternary functions.

Constructing a cascade of reversible "permutative" gates from some initial specification is one of the most fundamental problems in binary quantum circuit design. Recently, multiple-valued quantum gates and circuits have been presented [30] and ternary quantum gates have been built [31]. The concept of binary quantum circuit synthesis has been generalized to multiple-valued quantum circuits synthesis [3]. Among the multiple-valued quantum circuits the GFSOP cascades are the most fundamental ones [3,4]. The three types of Ternary Galois Field Decision Diagrams using the proposed expansions are the starting point to create such cascades in our software system [3,4]. They can be used also to create other types of quantum cascades. They are thus of a basic importance in multiple-valued quantum logic synthesis.

The remaining of the paper is organized as follows. In Section 2, Ternary Galois Field Sum of Products (TGFSOP) expression is introduced. In

Section 3, 16 Ternary Galois Field Expansions that use the reversible ternary literals are proposed. In Section 4, three types of Ternary Galois Field Decision Diagrams (TGFDDs) are introduced that use the expansions of Section 3. In Section 5, a heuristic for creating one of the TGFDDs introduced in Section 4 (named the Kronecker Ternary Galois Field Decision Diagram, KTGFDD) is presented. In Section 6, a method for flattening the TGFDDs introduced in Section 4 is discussed. In Section 7, experimental results are presented to show the complexity of KTGFDD and the resulting TGFSOP. In Section 8, conclusion about the paper and the future research guidelines are presented. In Section 9, references are given. Finally, in Section 10, some ternary benchmark functions are given as an appendix.

2. Ternary Galois Field Sum of Products expression

In this section we introduce Ternary Galois Field Sum of Products (TGFSOP) expressions that are found to be the natural choice for ternary quantum logic synthesis, especially for synthesis of ternary quantum cascades.

Ternary Galois Field (TGF) consists of the set of elements $T = \{0, 1, 2\}$ and two basic binary operations – ***addition*** (denoted by +) and ***multiplication*** (denoted by · or absence of any operator) as defined in Table 1. Readers should note that TGF is also known as GF3 and the addition and multiplication operations shown in Table 1 are modulo 3 addition and multiplication.

Table 1. Ternary Galois Field (TGF) operations.

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

•	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Literals of a ternary variable $x = (012)$ can be defined as follows:

Constant literals: Ternary constants 1 = (111) and 2 = (222) may be used as literals of a ternary variable $x = (012)$.

Basic literals: There are six basic literals of a ternary variable $x = (012)$ corresponding to six possible permutations of the elements 0, 1, and 2, which are reversible in nature:

$$x = (012) \quad \text{Normal literal}$$
$$x' = x + 1 = (120) \quad \text{Single-Shift literal}$$
$$x'' = x + 2 = (201) \quad \text{Dual-Shift literal}$$
$$x''' = 2x = (021) \quad \text{Self-Shift literal}$$
$$x^{\#} = 2x + 1 = (102) \quad \text{Self-Single-Shift literal}$$
$$x^{\wedge} = 2x + 2 = (210) \quad \text{Self-Dual-Shift literal}$$

These are the only literals that can be realized by 1-qubit reversible gates [4], so they have a special place in the diagrams and cascades developed by us. Other literals are created as Galois products of these basic literals, so their design cost is higher, which is taken into account in the synthesis methods.

Self-Shift of a basic ternary literal yields another basic ternary literal as follows (can be verified from Table 1):

$$2x = x''' \qquad 2x' = x^{\wedge} \qquad 2x'' = x^{\#}$$
$$2x''' = x \qquad 2x^{\#} = x'' \qquad 2x^{\wedge} = x'$$

Composite literals: There are some other literals, which are Galois products of two basic literals each, as below:

$$xx = x'''x''' = x^2 = (x''')^2 = (011)$$
$$xx' = x'''x^{\wedge} = (020)$$
$$xx'' = x'''x^{\#} = (002)$$
$$xx''' = (022)$$
$$xx^{\#} = x''x''' = {}^2x = (001)$$
$$xx^{\wedge} = x'x''' = {}^1x = (010)$$
$$x'x' = x^{\wedge}x^{\wedge} = (x')^2 = (x^{\wedge})^2 = (110)$$
$$x'x'' = x^{\wedge}x^{\#} = (200)$$
$$x'x^{\#} = x''x^{\wedge} = {}^0x = (100)$$
$$x'x^{\wedge} = (220)$$
$$x''x'' = x^{\#}x^{\#} = (x'')^2 = (x^{\#})^2 = (101)$$
$$x''x^{\#} = (202)$$

1-Reduced Post literals (RPLs): 1-RPLs of a variable $x = (012)$ are defined as

$${}^i x = \begin{cases} 1 & \textit{iff } x = i \\ 0 & \textit{otherwise} \end{cases}$$

For ternary Galois field the 1-RPLs of a variable $x = (012)$ are ${}^0x = (100)$, ${}^1x = (010)$, and ${}^2x = (001)$. These 1-RPLs are related with the basic and composite literals as follows (can be verified from Table 1):

$$\begin{aligned} {}^{0}x &= 2x^2+1 = 2(x')^2+2x' = 2(x'')^2+x'' \\ &= 2(x''')^2+1 = 2(x^{\#})^2+2x^{\#} = 2(x^{\wedge})^2+x^{\wedge} \end{aligned} \qquad (1)$$

$$\begin{aligned} {}^{1}x &= 2x^2+2x = 2(x')^2+x' = 2(x'')^2+1 \\ &= 2(x''')^2+x''' = 2(x^{\#})^2+1 = 2(x^{\wedge})^2+2x^{\wedge} \end{aligned} \qquad (2)$$

$$\begin{aligned} {}^{2}x &= 2x^2+x = 2(x')^2+1 = 2(x'')^2+2x'' \\ &= 2(x''')^2+2x''' = 2(x^{\#})^2+x^{\#} = 2(x^{\wedge})^2+1 \end{aligned} \qquad (3)$$

A product term is a TGF product of a constant, basic literals, and composite literals of ternary variables. For example, $2xx''y''$ is a product term. Ternary Galois Field Sum of Products (TGFSOP) expression is TGF sum of some product terms. For example, $2+xy''+yy'''z'+x'z''$ is a TGFSOP.

3. Ternary Galois Field Expansions

Binary Kronecker Functional Decision Diagrams (KFDDs) [32] are created using the following well-known expansions:

$f = x'f_0 \oplus xf_1$ Shannon Expansion

$f = f_0 \oplus xf_2$ Positive Davio Expansion

$f = f_1 \oplus x'f_2$ Negative Davio Expansion

where,

f_0 = cofactor of f with respect to $x=0$,

f_1 = cofactor of f with respect to $x=1$,

$f_2 = f_0 \oplus f_1$, and

$\oplus$ is EXOR (GF2 addition) operation.

In this section we extend the concept of binary Shannon and Davio expansion to the Ternary Galois Field and propose 16 Ternary Galois Field Expansions (TGFE) that use the reversible ternary literals and their composite forms. Among these 16 TGFEs we call 9 expansions the Pseudo-Davio expansions, because these expansions have sum of cofactors like binary Davio expansions but do not have a cofactor with no literal multiplied with it. These 16 TGFEs can be used to create Ternary Kronecker Decision Diagrams.

For the purpose of defining the TGFEs, we begin with the concept of cofactors of ternary functions. A ternary function f has the following cofactors:

f_0 = cofactor of f with respect to $x=0$

f_1 = cofactor of f with respect to $x=1$

f_2 = cofactor of f with respect to $x=2$

To derive expressions for the TGFEs we first define sums of two (possible weighted by a factor of 2) or three cofactors, as shown below:

$$\begin{array}{lll} f_{01}=f_0+f_1 & f_{02}=f_0+f_2 & f_{12}=f_1+f_2 \\ f_{012}=f_0+f_1+f_2 & f_{011}=f_0+2f_1 & f_{022}=f_0+2f_2 \\ f_{001}=2f_0+f_1 & f_{122}=f_1+2f_2 & f_{002}=2f_0+f_2 \\ f_{112}=2f_1+f_2 & & \end{array}$$

All these expressions generalize the concept of binary Boolean Difference used in (binary) Davio Expansions.

The proposed TGFEs are defined below as theorems:

Theorem 1 (Shannon Ternary Galois Field Expansion): A ternary function f can be expanded with respect to the variable x as follows:

$f = {}^{0}xf_0 + {}^{1}xf_1 + {}^{2}xf_2$ (TGFE 1) (4)

Proof. If $x=0$, then ${}^{0}x=1$, ${}^{1}x=0$, ${}^{2}x=0$, and $f = 1\cdot f_0 + 0\cdot f_1 + 0\cdot f_2 = f_0$. If $x=1$, then ${}^{0}x=0$, ${}^{1}x=1$, ${}^{2}x=0$, and $f = 0\cdot f_0 + 1\cdot f_1 + 0\cdot f_2 = f_1$. If $x=2$, then ${}^{0}x=0$, ${}^{1}x=0$, ${}^{2}x=1$, and $f = 0\cdot f_0 + 0\cdot f_1 + 1\cdot f_2 = f_2$. Thus, we have (4). **QED**

Theorem 2 (Pseudo-Davio Ternary Galois Field Expansions): A ternary function f can be expanded with respect to the variable x as follows:

$f = x'x^{\#}f_{01} + x^{\wedge}f_1 + xx^{\#}f_2$ (TGFE 2) (5)

$f = x'f_0 + xx^{\wedge}f_{01} + xx^{\#}f_2$ (TGFE 3) (6)

$f = x'x^{\#}f_{02} + xx^{\wedge}f_1 + x''f_2$ (TGFE 4) (7)

$f = x^{\#}f_0 + xx^{\wedge}f_1 + xx^{\#}f_{02}$ (TGFE 5) (8)

$f = x'x^{\#}f_0 + xx^{\wedge}f_{12} + x'''f_2$ (TGFE 6) (9)

$f = x'x^{\#}f_0 + xf_1 + xx^{\#}f_{12}$ (TGFE 7) (10)

$f = x^{\#}f_0 + xf_1 + xx^{\#}f_{012}$ (TGFE 8) (11)

$f = x'f_0 + xx^{\wedge}f_{012} + x'''f_2$ (TGFE 9) (12)

$f = x'x^{\#}f_{012} + x^{\wedge}f_1 + x''f_2$ (TGFE 10) (13)

Proof. By substituting the first part of (1), (2), and (3) in (4), we have

$$\begin{aligned} f &= (2x^2+1)f_0 + (2x^2+2x)f_1 + (2x^2+x)f_2 \\ &= (2x^2+1)f_0 + (2x^2+1+2x+2)f_1 + (2x^2+x)f_2 \\ &= (2x^2+1)(f_0+f_1) + (2x+2)f_1 + (2x^2+x)f_2 \\ &= x'x^{\#}f_{01} + x^{\wedge}f_1 + xx^{\#}f_2 \end{aligned}$$

Thus we have (5). Similarly, we can prove (6) to (13). **QED**

Theorem 3 (Davio Ternary Galois Field Expansions): A ternary function f can be expanded with respect to the variable x as follows:

$$f = f_0 + xf_{112} + xx'''f_{012} \qquad \text{(TGFE 11)} \qquad (14)$$

$$f = f_0 + x'''f_{122} + xx'''f_{012} \qquad \text{(TGFE 12)} \qquad (15)$$

$$f = f_1 + x''f_{022} + x''x^{\#}f_{012} \qquad \text{(TGFE 13)} \qquad (16)$$

$$f = f_1 + x^{\#}f_{002} + x''x^{\#}f_{012} \qquad \text{(TGFE 14)} \qquad (17)$$

$$f = f_2 + x'f_{001} + x'x^{\wedge}f_{012} \qquad \text{(TGFE 15)} \qquad (18)$$

$$f = f_2 + x^{\wedge}f_{011} + x'x^{\wedge}f_{012} \qquad \text{(TGFE 16)} \qquad (19)$$

Proof. By substituting the first part of (1), (2), and (3) in (4), we have

$$f = (2x^2+1)f_0 + (2x^2+2x)f_1 + (2x^2+x)f_2$$

$$= f_0 + x(2f_1+f_2) + 2x^2(f_0+f_1+f_2)$$

$$= f_0 + xf_{112} + xx'''f_{012}$$

Thus we have (14). By substituting the other parts of (1), (2), and (3) in (4) and after some manipulation, we can prove (15) to (19). **QED**

Some of the features of these expansions are explained using the pure decision tree (all levels having the same expansion with a fixed variable ordering) for the ternary function $f(x, y) = [0, 1, 1, 1, 2, 2, 1, 2, 2]^T$ using all 16 TGFEs for the variable ordering x, y as shown in Figure 1. From Figure 1 we observe that Davio expansions, that is, TGFEs 11 to 16 produce relatively more constant 0-leaves for function having less than one-third 0s in the truth vector. Among the six basic literals only three (normal, single-shift, and dual-shift) were previously used in the context of ternary reversible logic design. If we consider only the normal, single-shift, and dual-shift literals, then we will have only three Davio TGFEs (namely TGFEs 11, 13, and 15). From Figure 1 we also observe that TGFEs 12, 14, and 16 produce the same order of constant 0-leaves as TGFEs 11, 13, and 15. Therefore, we can see that the three new literals (self-shift, self-single-shift, and self-dual-shift) are equally useful as the previously used literals for the purpose of minimizing non-zero paths in the Ternary Galois Field Decision Diagrams and consequently for minimizing TGFSOP expression.

4. Ternary Galois Field Decision Diagrams

Among the different BDD types, Kronecker DDs use all three types of binary expansions [32]. In this section we extend the concept of binary Kronecker DDs to Ternary Galois Field.

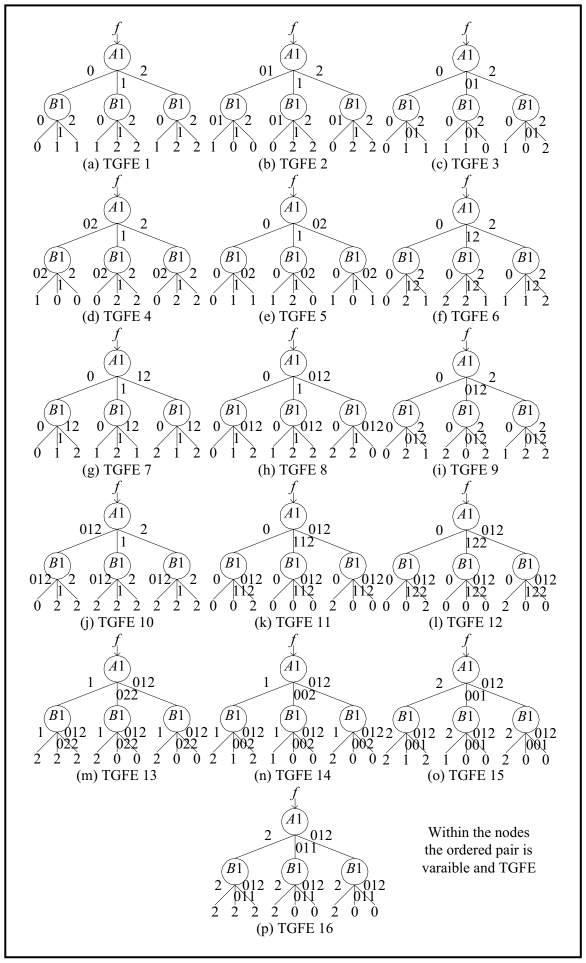

Figure 1. 16 types of pure ternary decision trees for the function $f(x, y) = [0, 1, 1, 1, 2, 2, 1, 2, 2]^T$.

For the purpose of deriving minimized TGFSOP expression of a given ternary function we use three types of Ternary Galois Field Decision Diagrams (TGFDDs) based on the proposed TGFEs that are useful for reversible logic design as described below:

Kronecker Ternary Galois Field Decision Diagram (KTGFDD): In KTGFDD the nodes of the same level have the same variable and the same TGFE. For example, in the KTGFDD of Figure 2.a, the first level variable is z and the expansion is TGFE16. In the second level the variable is x and the expansion for both the nodes is TGFE12. In the third level the variable is y and the expansion for both the nodes is TGFE15.

Pseudo-Kronecker Ternary Galois Field Decision Diagram (PKTGFDD): In PKTGFDD the nodes of the same level have the same variable, but each of the nodes may have one of the 16 TGFEs. For example, in the PKTGFDD of Figure 2.b, the first level variable is *z* and the expansion is TGFE16. In the second level the variable is *x* and the expansions for the left and the right nodes are TGFE14 and TGFE12, respectively. In the third level, the variable is *y* and the expansions for the left and the right nodes are TGFE14 and TGFE15, respectively.

Free Kronecker Ternary Galois Field Decision Diagram (FKTGFDD): In FKTGFDD each of the paths may have a different variable ordering and each of the nodes may have one of the 16 TGFEs. For example, in the FKTGFDD of Figure 2.c, the first level variable is *z* and the expansion is TGFE16. In the second level, the left node has variable *y* with expansion TGFE14, but the right node has variable *x* with expansion TGFE12. In the third level, the only node has variable *y* with expansion TGFE15. It can be seen that each of the paths of this diagram has different ordering of the variables and in the same level the expansions are also different.

For these three decision diagrams, in general, the following relationship holds: ***number of nodes of KTGFDD ≥ number of nodes of PKTGFDD ≥ number of nodes of FKTGFDD***.

Observe that KTGFDDs, PKTGFDDs and FKTGFDDs are adaptations of known concepts to reversible logic, and it is thanks to the realizability of multiple-valued quantum logic that these concepts may become more practical than their non-reversible counterparts.

5. Creating Kronecker Ternary Galois Field Decision Diagram

In this section we propose a heuristic for creating a KTGFDD in which the number of nodes as well as the number of paths terminating at constant 1-leaf and 2-leaf is minimized so that after flattening of the decision diagram the number of product terms in the resultant TGFSOP is also minimized. For this purpose we maximize the number of 0s in the truth vectors of each sub-function at every level of the KTGFDD with the hope that local optimization will lead to global optimization. For discussing the heuristic the following weight functions are useful.

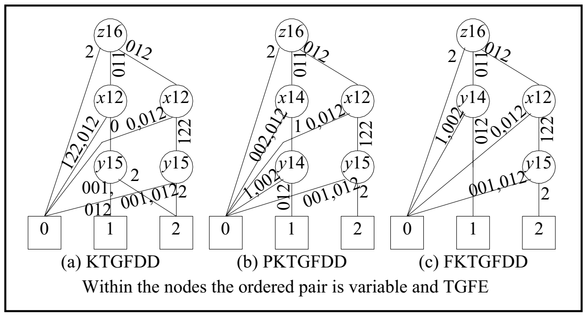

Figure 2. Three types of Ternary Galois Field Decision Diagrams for the function $f(x, y, z) =$ $[1,2,0,0,0,0,1,2,0,0,1,0,2,2,0,0,1,0,2,0,0,1,1,0,2,0,0]^T$.

Definition 1: Given an *n*-variable ternary function *f* represented as a truth vector, where the locations are designated from 0 to $3^n - 1$. The number of occurrences of a group of 3^{n-i} consecutive 0s, 1s, and 2s starting from $j3^{n-i}$ location for $i = 1, 2, \cdots, n$ and $j = 0, 1, \cdots, (3^i - 1)$ are denoted by Z_i, O_i, and T_i, respectively.

In the weight functions, *i* determines the length of a group of consecutive 0s or 1s or 2s and *j* determines the starting location of the group. For example, if $n = 3$, then for $i = 1$ the group length is $3^{n-i} = 3^{3-1} = 9$ and $j = 0, 1, (3^i - 1 = 3^1 - 1) = 2$. So, the starting locations are 0, 9, and 18. Similarly, these weight functions define the number of occurrences of a group length 1 starting from locations 0, 1, 2, …, $3^n - 1$; of a group length 3 starting from locations 0, 3, 9, …, $3^n - 3$; of a group length 9 starting from locations 0, 9, 18, …, $3^n - 9$ and so on to of a group length 3^{n-1} starting from location 0, 3^{n-1}, and $2 \times 3^{n-1}$. Consider the function $f(x,y,z) =$ $[0,0,0,0,1,2,1,1,1,2,0,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0]^T$. Then $Z_1 = 1$, $Z_2 = 5$, $Z_3 = 18$, $O_1 = 0$, $O_2 = 1$, $O_3 = 4$, $T_1 = 0$, $T_2 = 1$, and $T_3 = 5$.

The proposed heuristic is as follows:

1. If $Z_n \le m3^{n-1}$, where *m* is the number of outputs, then use TGFEs 1 and 11 to 16, otherwise use all TGFEs. The reason behind this selection is that if the truth vector contains less than one-third 0s, then using TGFEs 11 to 16 is likely to produce more 0s because of the use cofactor sums f_{112},

f_{122}, f_{022}, f_{002}, f_{001}, f_{011}, and f_{012} (can be verified from Figure 1).

2. For each of the n variables, find expansion for the TGFEs selected in step 1. For each expansion compute Z_1 to Z_n and $O_1 + T_1$ to $O_n + T_n$.
3. Find the expansion with highest Z_1. The reason behind this selection is that one or more cofactors will be straight constant 0. In case of a tie, break it using highest values of Z_2 up to Z_n. This selection will produce constant 0 cofactors in the later levels in the KTGFDD. For further tie, break it by using highest values of $O_1 + T_1$ up to $O_n + T_n$. This selection will produce a constant 0 or 1 or 2 cofactors in the later levels in the KTGFDD. For further tie, break it arbitrarily. The selected expansion is the expansion for the root of the KTGFDD.
4. For the next level of the KTGFDD, repeat the steps 1 to 3 for the remaining $n-1$ variables.
5. Repeat steps 1 to 4 until all the variables are exhausted.

The heuristic is illustrated using a 2-input 2-output ternary function $f(x, y) = [0,1,2,1,0,2,2,2,2]^T$ and $g(x, y) = [1,0,2,1,1,1,1,2,0]^T$. Here $Z_2 = 4 < 2 \times 3 = 6$. So we used TGFEs 1 and 11 to 16 for both the variables x and y. The weights are shown in Table 2. There are four ties as shown bold. We select variable y with TGFE 14 arbitrarily as the expansion for the roots. Then the resulting six sub-functions are $f_1(x) = [1,0,2]^T$, $f_{002}(x) = [2,1,0]^T$, $f_{012}(x) = [0,0,0]^T$, $g_1(x) = [0,1,2]^T$, $g_{002}(x) = [1,0,2]^T$, and $g_{012}(x) = [0,0,0]^T$. Here $Z_1 = 10 > 6 \times 1 = 6$. So, we used TGFEs 1 to 16. The weights are also shown in Table 2. There are two ties and we break it arbitrarily by selecting TGFE 10. The created KTGFDD is shown in Figure 3.

6. Flattening a Ternary Galois Field Decision Diagram

Flattening of a TGFDD yields a TGFSOP expressions for the function represented by the TGFDD. Different possible types of edges for TGFE1 in a TGFDD are shown in Figure 4. The literal associated with each of the edge types is shown in Table 3. Similarly, different types of edges and their corresponding literals for TGFE2 to TGFE16 are determined and are shown in Table 3.

Table 2. Weights of various expansions for the example function.

	Level 1 $Z_1, Z_2, O_1+T_1, O_2+T_2$		Level 2 Z_1, O_1+T_1
TGFE	x	y	x
1	0, 4, 2, 14	0, 4, 2, 14	10, 8
2			10, 8
3			9, 9
4			11, 7
5			11, 7
6			9, 9
7			10, 8
8			**13, 5**
9			12, 6
10			**13, 5**
11	**2, 10, 0, 8**	2, 9, 1, 9	11, 7
12	**2, 10, 0, 8**	2, 9, 1, 9	11, 7
13	2, 9, 1, 9	**2, 10, 0, 8**	12, 6
14	2, 9, 1, 9	**2, 10, 0, 8**	12, 6
15	2, 9, 1, 9	2, 9, 1, 9	11, 7
16	2, 9, 1, 9	2, 9, 1, 9	11, 7

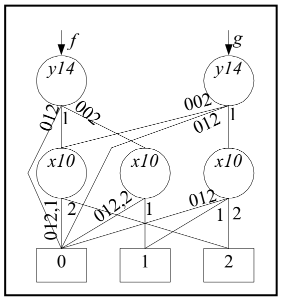

Figure 3. KTGFDD for the example function.

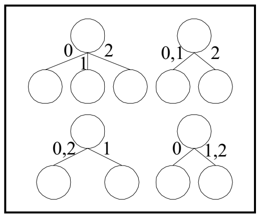

Figure 4. Different types of edges for TGFE1.

A product of literals corresponding to edges along a path from the root to a leaf of the TGFDD gives the TGF product of the represented function. Paths ended at the constant 0-leaf do not contribute to the TGFSOP and the TGF product corresponding to a path ended at the constant 2-leaf is multiplied by 2 to get the required product term. Sum of all such products gives the TGFSOP expression for the function represented in the TGFDD. For example, the TGFSOP expressions derived from the three TGFDDs of Figure 2 are

KTGFDD: $yy^{\wedge}z^{\wedge} + 2z^{\wedge} + 2x'''z'z^{\wedge}$

$$\text{PKTGFDD: } y''y^{\#}z^{\wedge}+2x'''z'z^{\wedge}$$

$$\text{FKTGFDD: } y''y^{\#}z^{\wedge}+2x'''z'z^{\wedge}$$

In general, the following relation holds: ***number of products from KTGFDD ≥ number of products from PKTGFDD ≥ number of products from FKTGFDD***.

Table 3. Different types of edges and their literals.

Edge	Literal	Edge	Literal
TGFE1		**TGFE2**	
0	$(100)=x'x^{\#}$	01	$(100)=x'x^{\#}$
1	$(010)=xx^{\wedge}$	1	$(210)=x^{\wedge}$
2	$(001)=xx^{\#}$	2	$(001)=xx^{\#}$
0,1	$(110)=x'x'$	01,1	$(010)=xx^{\wedge}$
0,2	$(101)=x''x''$	01,2	$(101)=x''x''$
1,2	$(011)=xx$	1,2	$(211)=x^{\wedge}+xx^{\#}$
TGFE3		**TGFE4**	
0	$(120)=x'$	02	$(100)=x'x^{\#}$
01	$(010)=xx^{\wedge}$	1	$(010)=xx^{\wedge}$
2	$(001)=xx^{\#}$	2	$(201)=x''$
0,01	$(100)=x'x^{\#}$	02,1	$(110)=x'x'$
0,2	$(121)=x'+xx^{\#}$	02,2	$(001)=x'x^{\#}$
01,2	$(011)=xx$	1,2	$(211)=x''+xx^{\wedge}$
TGFE5		**TGFE6**	
0	$(102)=x^{\#}$	0	$(100)=x'x^{\#}$
1	$(010)=xx^{\wedge}$	12	$(010)=xx^{\wedge}$
02	$(001)=xx^{\#}$	2	$(021)=x'''$
0,1	$(112)=x^{\#}+xx^{\wedge}$	0,12	$(110)=x'x'$
0,02	$(100)=x'x^{\#}$	0,2	$(121)=x'''+x'x^{\#}$
1,02	$(011)=xx$	12,2	$(001)=x'x^{\#}$
TGFE7		**TGFE8**	
0	$(100)=x'x^{\#}$	0	$(102)=x^{\#}$
1	$(012)=x$	1	$(012)=x$
12	$(001)=xx^{\#}$	012	$(001)=xx^{\#}$
0,1	$(112)=x+x'x^{\#}$	0,1	$(111)=1$
0,12	$(101)=x''x''$	0,012	$(100)=x'x^{\#}$
1,12	$(010)=xx^{\wedge}$	1,012	$(010)=xx^{\wedge}$

Table 3. Continued.

Edge	Literal	Edge	Literal
TGFE9		**TGFE10**	
0	$(120)=x'$	012	$(100)=x'x^{\#}$
012	$(010)=xx^{\wedge}$	1	$(210)=x^{\wedge}$
2	$(021)=x'''$	2	$(201)=x''$
0,012	$(100)=x'x^{\#}$	012,1	$(010)=xx^{\wedge}$
0,2	$(111)=1$	012,2	$(001)=xx^{\#}$
012,2	$(001)=xx^{\#}$	1,2	$(111)=1$
TGFE11		**TGFE12**	
0	$(111)=1$	0	$(111)=1$
112	$(012)=x$	122	$(021)=x'''$
012	$(022)=xx'''$	012	$(022)=xx'''$
0,112	$(120)=x'$	0,122	$(102)=x^{\#}$
0,012	$(100)=x'x^{\#}$	0,012	$(100)=x'x^{\#}$
112,012	$(001)=xx^{\#}$	122,012	$(010)=xx^{\wedge}$
TGFE13		**TGFE14**	
1	$(111)=1$	1	$(111)=1$
022	$(201)=x''$	002	$(102)=x^{\#}$
012	$(202)=x''x^{\#}$	012	$(202)=x''x^{\#}$
1,022	$(012)=x$	1,002	$(210)=x^{\wedge}$
1,012	$(010)=xx^{\wedge}$	1,012	$(010)=xx^{\wedge}$
022,012	$(100)=x'x^{\#}$	002,012	$(001)=xx^{\#}$
TGFE15		**TGFE16**	
2	$(111)=1$	2	$(111)=1$
001	$(120)=x'$	011	$(210)=x^{\wedge}$
012	$(220)=x'x^{\wedge}$	012	$(220)=x'x^{\wedge}$
2,001	$(201)=x''$	2,011	$(021)=x'''$
2,012	$(001)=xx^{\#}$	2,012	$(001)=xx^{\#}$
001,012	$(010)=xx^{\wedge}$	011,012	$(010)=xx^{\wedge}$

7. Experimental results

We have written C++ programs for creating KTGFDDs for multiple-output ternary functions and for flattening the KTGFDD for deriving the resultant TGFSOP expression. We have created some ternary benchmark functions as given in the Appendix of Section 10 and performed experimentation with them. Besides, we experimented with two benchmark functions (mm3 and pal3) from [33]. The results of the experimentation are given in Table 4. The fourth column of the table shows the number of nodes in the KTGFDD and the fifth column shows the number of product terms in the resultant TGFSOP expression.

For **prod*n*** functions, the number of nodes is exactly equal to the number of inputs for up to 8 input functions. The number of products is 1 for up to 8 input functions, which is the exact minimum solution for **prod*n*** functions. For **sum*n*** functions, the number of products is exactly equal to the number of inputs for up to 5 input functions, which is the exact minimum solution for **sum*n*** functions. For **3cy2**, **4cy2**, **4cy3**, **5cy2**, **5cy4**, and **6cy5** functions, the number of products is exactly equal to the number of inputs, which is the exact minimum solution. For **sqsum*n*** functions, the number of products is exactly equal to the number of inputs for up to 5 input functions, which is the exact minimum solution for **sqsum*n*** functions. For **a2bcc** and **mul2** functions, the number of products is 2, which is the exact minimum solution for these functions. For other functions, we have no theoretical results to make any comment. However, the results seem to be adequately moderate.

Table 4. Number of nodes in KTGFDD and number of resulting products in TGFSOP for some ternary benchmark functions.

Function	Input	Output	Nodes	Products
prod3	3	1	3	1
prod4	4	1	4	1
prod5	5	1	5	1
prod6	6	1	6	1
prod7	7	1	7	1
prod8	8	1	8	1
prod9	9	1	15	3
prod10	10	1	38	15
sum3	3	1	5	3
sum4	4	1	7	4
sum5	5	1	9	5
sum6	6	1	21	9
sum7	7	1	53	74
sum8	8	1	168	252
sum9	9	1	437	1117
sum10	10	1	905	2759
3cy2	3	1	5	3
4cy2	4	1	9	4
4cy3	4	1	9	4
5cy2	5	1	12	5
5cy3	5	1	15	7

Table 4. Continued.

Function	Input	Output	Nodes	Products
5cy4	5	1	11	5
6cy2	6	1	24	9
6cy3	6	1	34	24
6cy4	6	1	24	9
6cy5	6	1	15	6
sqsum3	3	1	5	3
sqsum4	4	1	7	4
sqsum5	5	1	9	5
sqsum6	6	1	20	18
sqsum7	7	1	58	63
sqsum8	8	1	203	295
sqsum9	9	1	428	855
sqsum10	10	1	1019	2506
avg3	3	1	10	7
avg4	4	1	20	27
avg5	5	1	34	69
avg6	6	1	54	183
avg7	7	1	75	516
avg8	8	1	175	1438
avg9	9	1	423	4396
avg10	10	1	1029	11802
a2bcc	3	1	5	2
thadd	2	2	5	4
tfadd	3	2	10	10
mul2	2	2	4	2
mul3	3	2	8	5
mami4	4	2	14	7
mm3	5	1	18	18
pal3	6	1	12	27

8. Conclusions

In this paper we proposed 16 Ternary Galois Field Expansions (TGFEs) which generalize to ternary and adapt to reversible logic the concepts of (binary) Shannon and Davio expansions used in Kronecker Decision Diagrams. We also introduced three types of Ternary Galois Field Decisions Diagrams (TGFDDs) suitable for reversible ternary logic synthesis. They are the Kronecker Ternary Galois Field Decision Diagram (KTGFDD), the Pseudo-Kronecker Galois Field Decision Diagram (PKTGFDD), and the Free-Kronecker Ternary Galois Field Decision Diagram (FKTGFDD). We proposed an efficient heuristic for creating KTGFDD with the reduced node counts and also reduced path counts that terminate at constant 1-leaf and 2-leaf. This method is applicable not only to KTGFDD but also to PKTGFDD and FKTGFDD. We also proposed a method of flattening the TGFDDs for determining Ternary Galois Field Sum of Products (TGFSOP) expression for ternary functions. As our KTGFDD reduces the paths terminating at constant 1-leaf and 2-leaf, the number of product terms in the resulting TGFSOP will also be reduced. The experimental results show that for many functions the resultant GFSOPs are exact minimum solutions.

Designing a cascade of reversible "permutative" gates is one of few fundamental problems in quantum computing. Among the multiple-valued quantum circuits the GFSOP cascades are the most fundamental ones. Various Ternary Galois Field Decision Diagrams are a starting point to create such cascades and can be used also to create other types of quantum cascades. They are thus of a basic importance in multi-valued quantum logic synthesis.

Further research includes implementing ternary counterparts of binary operations on these decision diagrams, such as cofactor, minimum, Galois sum, etc. Other research is on determining heuristics for creating optimal PKTGFDDs and FKTGFDDs that can potentially lead to more simpler TGFSOP expressions and their corresponding quantum cascades, as well as other types of quantum circuits.

9. References

[1] A. Al-Rabadi, L. W. Casperson, M. Perkowski and X. Song, "Multiple-Valued Quantum Logic", *Booklet of 11th Workshop on Post-Binary Ultra-Large-Scale Integration Systems (ULSI)*, Boston, Massachusetts, May 15, 2002, pp. 35-45.

[2] A. Al-Rabadi and M. Perkowski, "Multiple-Valued Galois Field S/D Trees for GFSOP Minimization and their Complexity", *Proc. 31st IEEE Int. Symp. on Multiple-Valued Logic,* Warsaw, Poland, May 22-24, 2001, pp. 159-166.

[3] M. Perkowski, A. Al-Rabadi, and P. Kerntopf, "Multiple-Valued Quantum Logic Synthesis", *Proc. 2002 International Symposium on New Paradigm VLSI Computing,* Sendai, Japan, December 12-14, 2002, pp. 41-47.

[4] M.H.A. Khan, M.A. Perkowski, and P. Kerntopf, "Multi-Output Galois Field Sum of Products Synthesis with New Quantum Cascades", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 146-153.

[5] R. Drechsler, "Evaluation of Static Variable Ordering Heuristics for MDD Construction," *Proc. 32nd IEEE Int. Symposium on Multiple-Valued Logic*, Boston, Massachusetts, May 15-18, 2002, pp. 254-260.

[6] R. Drechsler, D. Jankovic, and R.S Stankovic, "Generic Implementation of Decision Diagram Packages in MVL", Proc. Euromicro Conference (EUROMICRO'99), Vol. 1, Sept. 8-10, 1999, Milan, Italy, p.1352.

[7] R. Drechsler and D.M. Miller, "Decision Diagrams in Multi-Valued Logic., *Multiple-Valued Logic – An International Journal*, vol. 4, 1998, pp. 1-8.

[8] C. Files and M.A. Perkowski, "New Multi-valued Functional Decomposition Algorithm Based on MDDs", *IEEE Trans. on Computer-Aided Design*, vol. 14, 2000, pp. 1081-1086.

[9] S. Hassoun, T. Sasao, and R.K. Brayton (eds.), *Logic Synthesis and Verification*, Kluwer Academic Publishers, Boston/Dordrecht/London 2002, Chapters 11 and 15.

[10] D. Jankovic, W. Günther, and R. Drechsler, "Lower Bound Sifting for MDDs", *Proc. 30th IEEE Int. Symposium on Multiple-Valued Logic*, Portland, Oregon, USA. May 23-25, 2000, pp. 193-198.

[11] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "Multi-Valued Decision Diagrams: Theory and Applications", *Multiple-Valued Logic – An International Journal*, vol. 4, 1998, pp. 9-24.

[12] D.M. Miller, "Multiple-Valued Logic Design Tools", *Proc. 23rd IEEE Int. Symposium on Multiple-Valued Logic,* Sacramento, California, USA, May 24-27, 1993, pp. 2-11.

[13] D.M. Miller, and R. Drechsler, "Implementing a Multiple-Valued Decision Diagram Package", *Proc. 28th IEEE Int. Symposium on Multiple-Valued Logic*, Fukuoka, Japan, 27-29 May, 1998, pp. 52-57.

[14] D.M. Miller and R. Drechsler, "On the Construction of Multi-Valued Decision Diagrams", *Proc. 32nd IEEE Int. Symposium on Multiple-Valued Logic*, Boston, Massachusetts, May 15-18, 2002, pp. 245-253.

[15] H. Sack, E. Dubrova, and C. Meinel, "Mod-p Decision Diagrams: A Data Structure for Multiple-Valued Functions", *Proc. 30th IEEE Int. Symposium on Multiple-Valued Logic*, Portland, Oregon, USA, May 23-25, 2000, pp. 233-238.

[16] T. Sasao, "Ternary Decision Diagrams: Survey", *Proc. 27th IEEE Int. Symposium on Multiple-Valued Logic*, Antigonish, Nova Scotia, Canada, May 28-30, 1997, pp. 241-250.

[17] T. Sasao and J.T. Butler, "Planar Multiple-Valued Decision Diagrams", *Proc. 25th IEEE Int. Symposium on Multiple-Valued Logic*, Bloomington, Indiana, USA, May 23 - 25, 1995, pp. 28-35.

[18] T. Sasao and J.T. Butler, "A Method to Represent Multiple-Output Switching Functions by Using Multi-Valued Decision Diagrams", *Proc. 26th IEEE Int. Symposium on Multiple-Valued Logic*, Santiago de Compostela, Spain, May 19-31, 1996, pp. 248- 254.

[19] R.S. Stankovic, "Functional Decision Diagrams for Multiple-Valued Functions", *Proc. 25th IEEE Int. Symposium on Multiple-Valued Logic*, Bloomington, Indiana, USA, May 23 - 25, 1995, pp. 284-289.

[20] R.S. Stankovic and T. Sasao, "Decision Diagrams for Discrete Functions: Classification and Unified Interpretation", *Proc. Asia and South Pacific Design Automation Conference*, Yokohama, Japan, February 10-13, 1998, pp. 439-446.

[21] F. Schmiedle, W. Günther, and R. Drechsler, "Dynamic Re-Encoding During MDD Minimization", *Multiple- Valued Logic – An International Journal*, 2002, pp. 625-643.

[22] P. Kerntopf, "Multiple-Valued Decision Diagrams based on generalized Shannon expansion", *Booklet of 12th Workshop on Post-Binary Ultra-Large-Scale Integration Systems (ULSI)*, Tokyo, May 16, 2003, pp. 9-16.

[23] L. Macchiarulo and P. Civera, "Ternary Decision Diagrams with Inverted Edges and Cofactors – an Application to Discrete Neural Networks Synthesis", *Proc. 28th IEEE Int. Symp. on Multiple-Valued Logic*, Fukuoka, Japan, 27-29 May, 1998, pp. 58-63.

[24] E. Dubrova and J.C. Muzio, "Generalized Reed-Muller canonical form of a multiple-valued algebra", *Multiple-Valued Logic - An International Journal,* vol. 1, 1996, pp. 104 -109.

[25] D.M. Miller and G.W. Dueck, "On the Size of Multiple-Valued Decision Diagrams", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 235-240.

[26] D.V. Popel and R. Drechsler, "Efficient Minimization of Multiple-Valued Decision Diagrams for Incompletely Specified Function", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 241-246.

[27] S. Nagayama and T. Sasao, "Compact Representations of Logic Functions using Heterogeneous MDDs", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 247-252.

[28] D.M. Miller and R. Drechsler, "Augmented Sifting of Multiple-Valued Decision Diagrams", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 375-382.

[29] J.T. Butler and T. Sasao, "On the Average Path Length in Decision Diagrams of Multiple-Valued Functions", *Proc. 33rd IEEE Int. Symp. On Multiple-Valued Logic*, Tokyo, May 16-19, 2003, pp. 383-390.

[30] A. Muthukrishnan, and C.R. Stroud, Jr., "Multivalued logic gates for quantum computation," *Phys. Rev. A.* Vol. 62, 052309, 2000.

[31] A.V. Burlakov, M.V. Chekhova, O.V. Karabutova, D.N. Klyshko, and S.P. Kulik, "Polarization state of a biphoton: quantum ternary." *Phys. Rev. A 60,* R4209, 1999.

[32] A. Sarabi, P. F. Ho, K. Iravani, W. R. Daasch, M. A. Perkowski, "Minimal Multi-Level Realization of Switching Functions Based on Kronecker Functional Decision Diagrams," *Proc. of IEEE International Workshop on Logic Synthesis, IWLS '93,* Tahoe City, CA, pp. P3a-1 - P3a-6, May 1993.

[33] http://www.ee.pdx.edu/polo/

10. Appendix: Ternary benchmark functions created

prod*n*: input $x_0\ x_1 \cdots x_{n-1}$; output $y = (x_0 x_1 \cdots x_{n-1}) \bmod 3$. [Output is the GF3 product of *n* input variables.]

sum*n*: input $x_0\ x_1 \cdots x_{n-1}$; output $y = (x_0 + x_2 + \cdots + x_n) \bmod 3$. [Output is the GF3 sum of *n* input variables.]

***n*cyr**: input $x_0\ x_1 \cdots x_{n-1}$; output $y = \left[\sum_{i=0}^{n-1} + \left(\prod_{j=0}^{r-1} x_{(i+j) \bmod n}\right)\right] \bmod 3$. [A ternary GFSOP function of *n* input variables, where the products consist of *r* input variables in cyclic order. Example: For 3cy2, $y(a,b,c) = ab + bc + ca$.]

sqsum*n*: input $x_0\ x_1 \cdots x_{n-1}$; output $y = \left(x_0^2 + x_1^2 + \cdots + x_{n-1}^2\right) \bmod 3$. [Output is the GF3 sum of squares of *n* input variables]

avg*n*: input $x_0\ x_1 \cdots x_{n-1}$; output $y = \mathrm{int}[(x_0 + x_1 + \cdots + x_{n-1})/n] \bmod 3$. [Output is the integer part of the average of *n* input variables expressed as mod 3 value.]

a2bcc: input a, b, c; output $y = (a^2 + bc + c) \bmod 3$. [An arbitrary function]

thadd: input *a b*; output $c = \mathrm{int}[(a+b)/3]$, $s = (a+b) \bmod 3$. [Ternary half-adder]

tfadd: input *a b c*; output $y = \mathrm{int}[(a+b+c)/3]$, $s = (a+b+c) \bmod 3$. [Ternary full-adder]

mul2: input *a b*; output $c = \mathrm{int}[ab/3]$, $m = ab \bmod 3$. [2-trit ternary multiplier]

mul3: input *a b c*; output $c = \mathrm{int}[abc/3]$, $m = abc \bmod 3$. [3-trit ternary multiplier]

mami4: input *a b c d*; output $y = \max(a,b)$, $z = \min(c,d)$. [The output *y* is the maximum of the inputs *a* and *b*; the output *z* is the minimum of the inputs *c* and *d*.]