Quantum-Inspired Evolutionary Algorithm Based Minimization and Reversible/Quantum Synthesis of Ternary Galois Field Sum of Products Expressions

Mozammel H. A. Khan
Department of Computer Science and Engineering, East West University, 43 Mohakhali, Dhaka 1212, Bangladesh.
mhakhan@ewubd.edu

Abstract

Reversible/quantum multiple-valued logic circuit has several advantages over reversible/quantum binary logic circuit. Galois field sum of products (GFSOP) based synthesis of multiple-valued logic function is more promising and practical than other approaches, since any multiple-valued logic function with many inputs can be represented as GFSOP expression and the GFSOP expression can be implemented as cascade of reversible/quantum gates. Moreover, GFSOP based multiple-valued reversible/quantum logic circuit is highly testable. In this paper, we have developed 39 new ternary Galois field expansions (TGFE) in addition to our previously published 16 TGFEs resulting into a total of 55 TGFEs. We have proposed a quantum-inspired evolutionary algorithm (QEA) for minimization of non-reversible ternary logic function as Galois field sum of products (GFSOP) expression, which determines the best combination of TGFEs for the function variables producing the minimized GFSOP expression. We have also proposed a method of reversible/quantum realization of ternary GFSOP expression as cascade of ternary 1-qudit, Muthukrishnan-Stroud (M-S), Feynman, and Toffoli gates. For this purpose we have proposed realization of macro-level ternary Feynman and Toffoli gates on the top of primitive 1-qudit and M-S gates, which require no ancilla input constant in the realization. Experimental results with 20 ternary logic functions having up to 11 inputs and two outputs show that the proposed QEA for ternary GFSOP minimization works very effectively for finding the minimum solutions. The proposed QEA also produces better minimization for two functions than our previous heuristic minimization technique.

Keywords: Feynman gate realization, Galois field expansion, GFSOP expression, GFSOP minimization, GFSOP synthesis, multiple-valued logic, quantum-inspired evolutionary algorithm, quantum logic, reversible logic, ternary logic, Toffoli gate realization

1. Introduction

Landauer [1] proved that binary logic circuits built using classical irreversible gates lead to inevitable heat dissipation, regardless of the technology used to realize the gates. Zhirnov *et al.* [2] showed that heat dissipation in any future CMOS technology will lead to impossible heat removal and thus the speeding-up of CMOS devices will be impossible at some point of time. Bennett [3] proved that for heat not to be dissipated in a binary logic circuit, it is necessary that the circuit be built from reversible gates. *A gate (or circuit) is reversible if there is a one-to-one and onto (bijective) mapping between inputs and outputs.* Thus all output patterns are just permutations of input patterns. Bennett's theorem suggests that every future binary technology will have to use some kind of reversible gates in order to reduce heat dissipation. This is also true for multiple-valued logics, which by themselves demonstrate several potential advantages over binary technology.

Quantum technology is inherently reversible and is one of the most promising technologies for future computing systems [4]. Quantum algorithms allow solving problems much more efficiently than in classical computing. For instance, while classical algorithm needs N steps to search an unstructured database, a quantum Grover algorithm [5] needs only $\sqrt{N}$ steps. Another example is Shor's quantum algorithm [6] for prime factoring, which is exponentially faster than any known classical algorithm. All quantum algorithms are made up of quantum logic circuits that use superposition, entanglement, and interference of the quantum states to solve the problem. A quantum logic circuit is made up of quantum gates, which are inherently reversible in nature. These quantum logic circuits are designed using reversible logic synthesis methods [4]. Thus any development of the reversible logic synthesis method adds values to both classical reversible logic circuit synthesis and quantum logic circuit synthesis.

Multiple-valued quantum gates are realizable using existing quantum technologies [7-14]. The quantum technologies naturally provide *qudits* (quantum digits – the basic unit of multiple-valued quantum information) as quanta. *Qubits* (quantum bits – the basic unit of binary quantum information) are obtained by restricting the dynamics of the technology to just two of these quanta [9]. Thus, qudit realization and qubit realization in the existing quantum technologies are of the similar cost. A major obstacle in the quantum technologies is the limit on the number of coupled qubits (or qudits for multiple-valued case) that can be achieved. For a Hilbert space of N dimension, a binary quantum system, or qubit system, requires $n_2 = \log_2 N$ qubits, whereas a d-valued quantum system, or qudit system, requires $n_d = \log_d N = \frac{\log_2 N}{\log_2 d}$ qudits. Therefore, we have

$$\frac{n_d}{n_2} = \frac{\frac{\log_2 N}{\log_2 d}}{\log_2 N} = \frac{1}{\log_2 d},$$

which implies that the d-valued quantum system requires $1/\log_2 d$ times fewer qudits than the corresponding number of qubits. For d = 3, 4, and 5, the reduction of the quantum register widths are $1/1.58$, $1/2$, and $1/2.32$ times, respectively. Thus, the use of d-valued quantum system enables a much more compact and efficient information encoding than for binary quantum system. These advantages of multiple-valued quantum system open avenues for developing multiple-valued quantum algorithms. Moreover, as the realizations of both qubit and qudit are of the similar technological complexity, the d-valued encoded realization of binary quantum logic circuit is more compact and efficient than the binary quantum logic circuit. This advantage also provides possibility to use multiple-valued quantum logic circuit internally in the binary quantum algorithms.

A good number of works have been reported on ternary ($d = 3$) reversible/quantum logic synthesis [15-35]. There are also several works reported on quaternary ($d = 4$) reversible/quantum logic synthesis [36-42]. Quinary ($d = 5$) reversible/quantum logic synthesis is in the primitive state [43, 44]. These synthesis methods for ternary, quaternary, and quinary reversible/quantum logic circuits can be divided into three broad areas of approaches. In the first approach, technology dependent primitive gates are used for circuit synthesis [17, 19, 21, 23, 25, 27, 30, 34, 38]. This approach produces circuit with low cost but the design methods are generally very difficult to use for functions with many input variables. Within this approach, papers [17, 19, 21, 23, 27, 38] use deterministic methods, whereas papers [25, 30, 34] use Genetic Algorithm based methods. In the second approach, macro-level gates are used [15, 18, 20, 24, 26, 28, 29, 32, 33, 37, 39, 40, 42]. These macro-level gates are realizable on the top of technology dependent primitive gates [35, 36, 43, 44, 45, 46]. This approach is relatively easier, but the produced circuit is of higher cost. This approach is also relatively difficult to use for functions with many input variables. Within this approach, there are also two different kinds of methods. Papers [15, 18, 24, 26, 28, 29, 32, 33, 37, 39, 40, 42] use deterministic methods, whereas paper [20] uses Genetic Algorithm based method. The third approach is to synthesize reversible/quantum logic circuits as Galois field sum of products (GFSOP) circuits [16, 22, 35, 36, 41, 43, 44]. The advantage of this approach is that any multiple-valued logic function with many input variables can be represented as GFSOP expression and the GFSOP expression can be easily implemented as cascade of multiple-valued 1-qudit, Muthukrishnan-Stroud (M-S) [8], Feynman, and Toffoli gates, where the macro-level Feynman and Toffoli gates are realizable on the top of technology dependent primitive gates like 1-qudit and M-S gates [35, 36, 45, 46, 47, 48]. This approach has additional advantage that GFSOP based circuits are highly testable [49]. This approach extensively uses Feynman and multiple-input Toffoli gates for GFSOP realization. The most remarkable drawback of this approach was that realization of macro-level Feynman and Toffoli gates on the top of primitive gates required a large number of ancilla input constants [36, 45, 46], which made the synthesized circuit having a very large number of ancilla input constants. Recently, ancilla input free realization of Feynman and Toffoli gates on the top of 1-qudit and M-S gates is reported in [48]. Thus, the GFSOP based synthesis of reversible/quantum multiple-valued logic circuit has become very promising and practical for four reasons: (i) any multiple-valued logic function with many inputs can be easily represented as minimized GFSOP expression, (ii) GFSOP expressions can be easily realized as cascade of 1-qudit, M-S, Feynman, and Toffoli gates, (iii) the realizations of Feynman and Toffoli gates on the top of technology dependent primitive gates now do not require any ancilla input constant, which makes the width of the synthesized quantum circuit minimum, and (iv) the GFSOP based quantum circuits are highly testable.

Quantum-inspired evolutionary algorithm (QEA) has been proposed by Han and Kim [50] and experimentally shown that the QEA is better than classical genetic algorithms for solving 0/1 knapsack problem. Latter, in [51], they have proposed extension of the basic QEA such as termination criterion, a modified version of the variation operator, and a two-phase scheme to improve the performance of the QEA. Besides, applications of the QEA in different application areas such as disk allocation problem [52], multiobjective 0/1 knapsack problem [53], face detection [54], and multiple-case outlier detection in least-squares regression model [55] have been reported. The basic QEA structure presented in [50] is based on the concept and principle of quantum computing. As the QEA is found to be effective for other combinatorial problems, it is also likely that it will be very suitable for ternary GFSOP minimization, since the selection of TGFEs for the function variables for minimization is combinatorial in nature.

In our earlier work of [22], we have developed 16 ternary Galois field expansions (TGFEs) and proposed a heuristic method (semi-exhaustive) for ternary GFSOP minimization using these TGFEs. In this paper, we have developed 39 new TGFEs making a total of 55 TGFEs. If the given ternary logic function has n variables, then there are 55^n possible ways of choosing TGFEs for the variables for ternary GFSOP minimization. For example, if $n = 11$, then there are $55^{11} = 1.393123392 \times 10^{19}$ ways of choosing TGFEs for variables. Exhaustive search of this huge search space is absolutely impossible. To overcome this difficulty, we have proposed a QEA for ternary GFSOP minimization, which determines the optimal combination of TGFEs to produce the best minimization possible. We have also proposed a method for ternary GFSOP synthesis that is different from that of [22]. The proposed GFSOP based synthesis of reversible/quantum ternary logic circuit has two distinct jobs, one is to represent the function as minimized GFSOP expression and the other is to realize the GFSOP expression as cascade of reversible/quantum 1-qudit, M-S, Feynman, and Toffoli gates. For the purpose of reversible/quantum synthesis of ternary GFSOP expression as cascade of reversible/quantum 1-qudit, M-S, Feynman, and Toffoli gates, we have proposed realization of macro-level ternary Feynman and Toffoli gates on the top of primitive 1-qudit and M-S gates, which require no ancilla input constant in the realization. The presented methods of this paper make possible synthesizing non-reversible ternary multiple-output logic functions as a reversible/quantum circuits.

The rest of the paper is organized as follows. In Section 2, we have discussed the concept of GF(3). We have introduced the concept of ternary Galois field sum of products (GFSOP) expression in Section 3. In Section 4, we have developed ternary Galois field expansions (GFE) using which ternary logic functions can be expressed as GFSOP expressions. We have shown the technique of minimizing a non-reversible ternary logic function expressed as output vector as ternary GFSOP expression by the applications of TGFEs on the function variables in Section 5. In Section 6, we have proposed a QEA for minimization of ternary GFSOP expression. We have introduced technology dependant ternary primitive gates like 1-qudit gates and Muthukrishnan-Stroud (M-S) gates [8] in Section 7. In Section 8, we have shown realization of ternary 1-reduced Post literals (1-RPL) and reversible literals using 1-qudit and M-S gates. We have developed method for realization of ternary Feynman gates on the top of 1-qudit and M-S gates that does not require any ancilla input constant in the realization in Section 9. In Section 10, we have developed method for realization of ternary Toffoli gates on the top of 1-qudit and M-S gates that do not need any ancilla input constant in the realization. We have developed heuristic technique for synthesis of ternary multiple-output GFSOP expressions as cascade of 1-qudit, M-S, Feynman, and Toffoli gates that minimizes the number of 1-qudit gates as well as the width of the realized circuit in Section 11. In Section 12, we have shown experimental results for minimization of ternary GFSOP expressions. Finally, we have concluded the chapter in Section 13.

2. Galois Field GF(3)

A field is a set F with two binary operations - addition (denoted by +) and multiplication (denoted by $\cdot$ or juxtaposition), are defined, which satisfies the following axioms:

(A1) $a + (b + c) = (a + b) + c$ (associative law for addition)

(A2) $a + b = b + a$ (commutative law for addition)

(A3) There is an element 0 (zero) such that $a + 0 = a$ for all a

(A4) For any a, there is an element ($-a$) such that $a + (-a) = 0$

(M1) $a\cdot(b\cdot c)=(a\cdot b)\cdot c$ (associative law for multiplication)

(M2) $a\cdot b=b\cdot a$ (commutative law for multiplication)

(M3) There is an element 1 (not equal to 0) such that $a\cdot 1=a$ for all a

(M4) For any $a\neq 0$, there is an element a^{-1} such that $a\cdot a^{-1}=1$

(D) $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$ (distributive law)

If p is a prime integer, then the *integers mod* p form a Galois field (also known as finite field): its elements are the congruence classes of integers mod p, with addition and multiplication induced from integer mod operations. The Galois field with p elements is abbreviated as GF(p). GF(3) addition and multiplication tables are shown in Tables 1 and 2, respectively. The readers can verify that the operations are integer mod 3 operations.

Table 1. GF(3) addition table.

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

Table 2. GF(3) multiplication table.

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

In this paper, unless otherwise explicitly stated, all addition (+) and multiplication (·) operations are GF(3) operations.

3. Ternary Galois Field Sum of Products Expression

For expressing a ternary logic function as Galois field sum of products (GFSOP) expression, we will use two types of literals – 1-reduced Post literals (1-RPLs) and reversible literals as defined below.

Definition 1. A ternary *1-reduced Post literal (1-RPL)* represents a mapping form a variable x to a transformed value x^i, $x^i:\{0,1,2\}\rightarrow\{0,1\}$, such that

$$(\forall i\in\{0,1,2\})\ x^i=\begin{cases}1 & \text{if } x=i\\ 0 & \text{otherwise}\end{cases}.$$

The ternary 1-RPLs are x^0, x^1, and x^2. The ternary 1-RPLs can be expressed using the following GF(3) polynomial expressions:

$$x^0=2x^2+1 \tag{1}$$

$$x^1=2x^2+2x \tag{2}$$

$$x^2=2x^2+x \tag{3}$$

This sort of GF(3) polynomial representation of ternary 1-RPLs will be useful in developing ternary Galois field expansions (TGFE) in Section 4. In Section 8, we will see that ternary 1-RPLs can be realized on the top of technology dependent primitive 1-qudit and M-S gates.

Definition 2. A ternary *reversible literal* represents a bijective mapping from a variable x to a transformed value $x^{<q>}$, $x^{<q>}:\{0,1,2\}\rightarrow\{0,1,2\}$, where $<q>$ is a string of length two from the alphabet $\{+,0,1,2\}$. There are $3!=6$ ternary reversible literals.

The six ternary reversible literals can be grouped into two categories. Let us assume that $a, b \in \{0,1,2\}$ and $a \neq b$. Then, the two groups of ternary reversible literals can be described as follows:

(i) A reversible literal of the form x^{+a} represents a shift of the variable x by a, i.e., $x^{+a} = x + a$. There are three such reversible literals (readers should note that $x^{+0} = x$).
(ii) A reversible literal of the form x^{ab} represents a cyclic change of the value of the variable x, i.e., a becomes b, and b becomes a. There are three such reversible literals.

The six ternary reversible literals can be expressed using the following GF(3) polynomial expressions:

$$x^{+0} = x + 0$$
$$x^{+1} = x + 1$$
$$x^{+2} = x + 2$$
$$x^{12} = 2x$$
$$x^{01} = 2x + 1$$
$$x^{02} = 2x + 2$$

The GF(3) polynomial expressions for ternary reversible literals will be useful for developing ternary Galois field expansions (TGFE) in Section 4. In Section 8, we will see that ternary reversible literals can be realized using technology dependent primitive 1-qudit gates.

Readers should note that the length of the superscript for ternary 1-RPLs is one and that of the ternary reversible literals is two. Thus, the length of the superscript differentiates which one is a 1-RPL and which one is a reversible literal.

In Section 5, we will see that for writing ternary GFSOP expressions, reversible literals are needed to be multiplied by constant 2. Such multiplication of a reversible literal by a constant 2 is equivalent to another reversible literal and can be determined using the GF(3) polynomial expressions of the reversible literals. Consider the following example:

$$2x^{01} = 2(2x + 1) = x + 2 = x^{+2}.$$

This can also be done using truth table as shown in Table 3. In Table 3, the first column lists all possible values of the variable x, the second column lists the corresponding values produced by the literal x^{01}, and the third column lists the values resulted by multiplying the values of the second column by 2. Inspection of the third column reveals that the values are $x + 2$ making the conclusion that $2x^{01} = x^{+2}$. Equivalent ternary reversible literals for product of constant 2 and ternary reversible literals are given below:

$$2x = x^{12}$$
$$2x^{+1} = x^{02}$$
$$2x^{+2} = x^{01}$$
$$2x^{12} = x$$
$$2x^{01} = x^{+2}$$
$$2x^{02} = x^{+1}$$

Table 3. Determination of equivalent reversible literal for $2x^{01}$.

x	x^{01}	$2x^{01}$
0	1	2
1	0	0
2	2	1

From the definition of reversible literals, it is clear that reversible literals represent 1-qudit permutation operations.

Galois field sum of products (GFSOP) expression is defined using the following definitions.

Definition 3. Product of ternary 1-RPLs and reversible literals are known as *ternary Galois field product (TGFP)*. For example, $x^1 y^{02} z^{+2}$ is a TGFP.

Definition 4. Sum of TGFPs is known as ternary *GFSOP expression*. For example, $x^1 y^{02} z^{+2} + x^{02} y^{+1} z^{12} + xz^1$ is a ternary GFSOP expression.

Definition 5. If a ternary GFSOP expression contains only 1-RPLs and each TGFP contains all variables, then we call that expression a *canonical GFSOP expression*. For example, $x^1 y^2 z^0 + x^2 y^2 z^1 + x^2 y^1 z^0$ is a ternary canonical GFSOP expression.

The truth table of ternary full-adder is shown in Table 4. Readers should note that, in ternary full-adder, carry inputs are either 0 or 1. Therefore, for carry input $C = 2$, outputs are don't cares as shown in Table 4. In this paper, we have not considered the problem of don't care assignments, and, therefore, we have assumed 0s for all don't care outputs in the ternary GFSOP minimization process. Canonical ternary GFSOP expression for a function can be written directly from the truth table. In Table 4, the first non-zero output of the sum output S is 1 and the corresponding input combination is $ABC = 001$. When $A = 0$, then $A^0 = 1$; when $B = 0$, then $B^0 = 1$; and when $C = 1$, then $C^1 = 1$. Therefore, when $ABC = 001$, then $A^0 B^0 C^1 = 1$. Thus, the TGFP corresponding to the input combination $ABC = 001$ is $A^0 B^0 C^1$. If we multiply the TGFP $A^0 B^0 C^1$ by the corresponding output 1, then the resulting TGFP is $A^0 B^0 C^1$, which produces the desired output 1. The third non-zero output of the sum output S is 2 and the corresponding input combination is $ABC = 011$. The TGFP for this input combination is $A^0 B^1 C^1$. When $ABC = 011$, then $A^0 B^1 C^1 = 1$. If this TGFP is multiplied by the corresponding output 2, then we get the desired output 2. Therefore, the resulting TGFP for this output is $2A^0 B^1 C^1$. Similarly, TGFPs for all non-zero outputs (don't care outputs are assumed to be 0s) are determined and summed up to write the ternary canonical GFSOP expression. The ternary canonical GFSOP expressions for the sum output S and the carry output C_o are given in (4) and (5), respectively. The sum output requires 12 TGFPs and the carry output requires nine TGFEs resulting into a total of 21 TGFEs in canonical GFSOP expressions for ternary full-adder function.

$$\begin{aligned} S(A,B,C) = {} & A^0B^0C^1 + A^0B^1C^0 + 2A^0B^1C^1 + 2A^0B^2C^0 + A^1B^0C^0 + 2A^1B^0C^1 + \\ & 2A^1B^1C^0 + A^1B^2C^1 + 2A^2B^0C^0 + A^2B^1C^1 + A^2B^2C^0 + 2A^2B^2C^1 \end{aligned} \tag{4}$$

$$C_o(A,B,C) = A^0B^2C^1 + A^1B^1C^1 + A^1B^2C^0 + A^1B^2C^1 + A^2B^0C^1 + A^2B^1C^0 + A^2B^1C^1 + A^2B^2C^0 + A^2B^2C^1 \tag{5}$$

4. Ternary Galois Field Expansions

In this section, we will develop ternary Galois field expansions (TGFE) using which any ternary logic function can be expressed as ternary GFSOP expression.

Definition 6. The *cofactors* of an n-variable ternary logic function f with respect to the variable x_i are defined as follows:

$$(\forall j \in \{0,1,2\}) \; f_j = f(x_1, \cdots, x_{i-1}, j, x_{i+1}, \cdots, x_n)$$

The three ternary cofactors are

$$\begin{aligned} f_0 &= f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots x_n) \\ f_1 &= f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots x_n) \\ f_2 &= f(x_1, \cdots, x_{i-1}, 2, x_{i+1}, \cdots x_n) \end{aligned}$$

These cofactors are the fundamental cofactors based on which composite cofactors are defined as follows.

Definition 7. A *composite cofactor* is defined as

$$f_{(a_0\cdot 0)(a_1\cdot 1)(a_2\cdot 2)} = a_0 f_0 + a_1 f_1 + a_2 f_2$$

where, $(\forall i \in \{0,1,2\})\ a_i \in \{0,1,2\}$; $a_i = 1$ is not explicitly written and in that case the braces are omitted.

Based on all possible values of a_0, a_1, and a_2, the all possible ternary composite cofactors are given in Table 5.

Table 4. Truth table of ternary full-adder.

ABC	C_oS
000	00
001	01
002	xx
010	01
011	02
012	xx
020	02
021	10
022	xx
100	01
101	02
102	xx
110	02
111	10
112	xx
120	10
121	11
122	xx
200	02
201	10
202	xx
210	10
211	11
212	xx
220	11
221	12
222	xx

The fundamental TGFE is given in the following theorem. In the following theorem, we will use the symbol x to represent the variable x_i in an n-variable ternary logic function f to make the expressions more readable.

Theorem 1. Any n-variable ternary logic function f can be expanded with respect to the variable x using the following fundamental ternary Galois field expansion (TGFE):

TGFE 0: $f = x^0 f_0 + x^1 f_1 + x^2 f_2$ (cost = 6) (6)

Proof. We will prove the theorem using perfect induction.
If $x = 0$, then $f = x^0 f_0 + x^1 f_1 + x^2 f_2 = 1\cdot f_0 + 0\cdot f_1 + 0\cdot f_2 = f_0$.

If $x = 1$, then $f = x^0 f_0 + x^1 f_1 + x^2 f_2 = 0 \cdot f_0 + 1 \cdot f_1 + 0 \cdot f_2 = f_1$.
If $x = 2$, then $f = x^0 f_0 + x^1 f_1 + x^2 f_2 = 0 \cdot f_0 + 0 \cdot f_1 + 1 \cdot f_2 = f_2$.
Thus, we have the theorem.

[End of Proof]

Table 5. Ternary composite cofactors.

$a_0 a_1 a_2$	Composite Cofactor	Comments
000		No cofactor
001	f_2	Fundamental cofactor
002	$f_{(2\cdot 2)} = 2f_2$	Composite of one cofactor
010	f_1	Fundamental cofactor
011	$f_{12} = f_1 + f_2$	Composite of two cofactors
012	$f_{1(2\cdot 2)} = f_1 + 2f_2$	Composite of two cofactors
020	$f_{(2\cdot 1)} = 2f_1$	Composite of one cofactor
021	$f_{(2\cdot 1)2} = 2f_1 + f_2$	Composite of two cofactors
022	$f_{(2\cdot 1)(2\cdot 2)} = 2f_1 + 2f_2$	Composite of two cofactors
100	f_0	Fundamental cofactor
101	$f_{02} = f_0 + f_2$	Composite of two cofactors
102	$f_{0(2\cdot 2)} = f_0 + 2f_2$	Composite of two cofactors
110	$f_{01} = f_0 + f_1$	Composite of two cofactors
111	$f_{012} = f_0 + f_1 + f_2$	Composite of three cofactors
112	$f_{01(2\cdot 2)} = f_0 + f_1 + 2f_2$	Composite of three cofactors
120	$f_{0(2\cdot 1)} = f_0 + 2f_1$	Composite of two cofactors
121	$f_{0(2\cdot 1)2} = f_0 + 2f_1 + f_2$	Composite of three cofactors
122	$f_{0(2\cdot 1)(2\cdot 2)} = f_0 + 2f_1 + 2f_2$	Composite of three cofactors
200	$f_{(2\cdot 0)} = 2f_0$	Composite of one cofactor
201	$f_{(2\cdot 0)2} = 2f_0 + f_2$	Composite of two cofactors
202	$f_{(2\cdot 0)(2\cdot 2)} + 2f_0 + 2f_2$	Composite of two cofactors
210	$f_{(2\cdot 0)1} = 2f_0 + f_1$	Composite of two cofactors
211	$f_{(2\cdot 0)12} = 2f_0 + f_1 + f_2$	Composite of three cofactors
212	$f_{(2\cdot 0)1(2\cdot 2)} = 2f_0 + f_1 + 2f_2$	Composite of three cofactors
220	$f_{(2\cdot 0)(2\cdot 1)} = 2f_0 + 2f_1$	Composite of two cofactors
221	$f_{(2\cdot 0)(2\cdot 1)2} = 2f_0 + 2f_1 + f_2$	Composite of three cofactors
222	$f_{(2\cdot 0)(2\cdot 1)(2\cdot 2)} = 2f_0 + 2f_1 + 2f_2$	Composite of three cofactors

For the QEA for ternary GFSOP minimization, we have assigned a cost factor to every TGFE as defined below:

Cost of TGFE = Total number of reversible literals + 2 (Total number of 1-RPLs) in the TGFE.

In Section 11, we will see that a reversible literal is realized along the input variable line using a 1-qudit gate, but an 1-RPLs is realized using an M-S gate that requires an additional input constant 0 line. For this reason, we have assigned double cost to 1-RPLs than reversible literals.

The fundamental TGFE contains only 1-RPLs. Application of the fundamental TGFE on all variables of a ternary logic function produces canonical ternary GFSOP expression.

Based on TGFE 0 of (6), we have developed 54 TGFEs using ternary 1-RPLs, reversible literals, and composite cofactors as given in the following theorem. Among these 54 TGFEs, fifteen TGFEs such as TGFEs 1, 2, 9, 10, 13, 14, 25, 26, 27, 28, 29, 30, 31, 32, and 33 are reported in [22] in different notations and are rewritten here using the general notation used in this paper.

Theorem 2. Any n-variable ternary logic function f can be expanded with respect to the variable x using any of the following 54 TGFEs:

TGFE 1: $f = x^{0} f_{0} + x^{1} f_{12} + x^{12} f_{2}$ (cost = 5)
TGFE 2: $f = x^{0} f_{0} + x f_{1} + x^{2} f_{12}$ (cost = 5)
TGFE 3: $f = x^{0} f_{0} + x^{1} f_{1(2\cdot 2)} + xx f_{2}$ (cost = 6)
TGFE 4: $f = x^{0} f_{0} + xx f_{1} + xx^{+2} f_{1(2\cdot 2)}$ (cost = 6)
TGFE 5: $f = x^{0} f_{0} + xx^{+1} f_{(2\cdot 1)2} + xx f_{2}$ (cost = 6)
TGFE 6: $f = x^{0} f_{0} + xx f_{1} + x^{2} f_{(2\cdot 1)2}$ (cost = 6)
TGFE 7: $f = x^{0} f_{0} + xx^{+1} f_{(2\cdot 1)(2\cdot 2)} + x^{12} f_{2}$ (cost = 5)
TGFE 8: $f = x^{0} f_{0} + x f_{1} + xx^{+2} f_{(2\cdot 1)(2\cdot 2)}$ (cost = 5)
TGFE 9: $f = x^{0} f_{02} + x^{1} f_{1} + x^{+2} f_{2}$ (cost = 5)
TGFE 10: $f = x^{01} f_{0} + x^{1} f_{1} + x^{2} f_{02}$ (cost = 5)
TGFE 11: $f = x^{0} f_{0(2\cdot 2)} + x^{1} f_{1} + x^{+2} x^{+2} f_{2}$ (cost = 6)
TGFE 12: $f = x^{+2} x^{+2} f_{0} + x^{1} f_{1} + xx^{+2} f_{0(2\cdot 2)}$ (cost = 6)
TGFE 13: $f = x^{0} f_{01} + x^{02} f_{1} + x^{2} f_{2}$ (cost = 5)
TGFE 14: $f = x^{+1} f_{0} + x^{1} f_{01} + x^{2} f_{2}$ (cost = 5)
TGFE 15: $f = x^{0} f_{0(2\cdot 1)} + x^{+1} x^{+1} f_{1} + x^{2} f_{2}$ (cost = 6)
TGFE 16: $f = x^{+1} x^{+1} f_{0} + xx^{+1} f_{0(2\cdot 1)} + x^{2} f_{2}$ (cost = 6)
TGFE 17: $f = x^{+1} x^{+2} f_{(2\cdot 0)2} + x^{1} f_{1} + x^{+2} x^{+2} f_{2}$ (cost = 6)
TGFE 18: $f = x^{+2} x^{+2} f_{0} + x^{1} f_{1} + x^{2} f_{(2\cdot 0)2}$ (cost = 6)
TGFE 19: $f = x^{+1} x^{+2} f_{(2\cdot 0)(2\cdot 2)} + x^{1} f_{1} + x^{+2} f_{2}$ (cost = 5)
TGFE 20: $f = x^{01} f_{0} + x^{1} f_{1} + xx^{+2} f_{(2\cdot 0)(2\cdot 2)}$ (cost = 5)
TGFE 21: $f = x^{+1} x^{+2} f_{(2\cdot 0)1} + x^{+1} x^{+1} f_{1} + x^{2} f_{2}$ (cost = 6)
TGFE 22: $f = x^{+1} x^{+1} f_{0} + x^{+1} x^{12} f_{(2\cdot 0)1} + x^{2} f_{2}$ (cost = 6)
TGFE 23: $f = x^{+1} x^{+2} f_{(2\cdot 0)(2\cdot 1)} + x^{02} f_{1} + x^{2} f_{2}$ (cost = 5)
TGFE 24: $f = x^{+1} f_{0} + xx^{+1} f_{(2\cdot 0)(2\cdot 1)} + x^{2} f_{2}$ (cost = 5)
TGFE 25: $f = f_{0} + xx^{12} f_{012} + x f_{(2\cdot 1)2}$ (cost = 3)
TGFE 26: $f = f_{0} + x^{12} f_{1(2\cdot 2)} + xx^{12} f_{012}$ (cost = 3)
TGFE 27: $f = x^{+2} x^{01} f_{012} + f_{1} + x^{01} f_{(2\cdot 0)2}$ (cost = 3)
TGFE 28: $f = x^{+2} f_{0(2\cdot 2)} + f_{1} + x^{+2} x^{01} f_{012}$ (cost = 3)
TGFE 29: $f = x^{+1} x^{02} f_{012} + x^{+1} f_{(2\cdot 0)1} + f_{2}$ (cost = 3)
TGFE 30: $f = x^{02} f_{0(2\cdot 1)} + x^{+1} x^{02} f_{012} + f_{2}$ (cost = 3)

TGFE 31: $f = x^{0} f_{012} + x^{02} f_{1} + x^{+2} f_{2}$ (cost = 4)
TGFE 32: $f = x^{+1} f_{0} + x^{1} f_{012} + x^{12} f_{2}$ (cost = 4)
TGFE 33: $f = x^{01} f_{0} + xf_{1} + x^{2} f_{012}$ (cost = 4)
TGFE 34: $f = x^{0} f_{01(2\cdot2)} + x^{02} f_{1} + x^{+2}x^{+2} f_{2}$ (cost = 5)
TGFE 35: $f = x^{+1} f_{0} + x^{1} f_{01(2\cdot2)} + xxf_{2}$ (cost = 5)
TGFE 36: $f = x^{+2}x^{+2} f_{0} + xxf_{1} + xx^{+2} f_{01(2\cdot2)}$ (cost = 6)
TGFE 37: $f = x^{0} f_{0(2\cdot1)2} + x^{+1}x^{+1} f_{1} + x^{+2} f_{2}$ (cost = 5)
TGFE 38: $f = x^{+1}x^{+1} f_{0} + xx^{+1} f_{0(2\cdot1)2} + xxf_{2}$ (cost = 6)
TGFE 39: $f = x^{01} f_{0} + xxf_{1} + x^{2} f_{0(2\cdot1)2}$ (cost = 5)
TGFE 40: $f = x^{0} f_{0(2\cdot1)(2\cdot2)} + x^{+1}x^{+1} f_{1} + x^{+2}x^{+2} f_{2}$ (cost = 6)
TGFE 41: $f = x^{+1}x^{+1} f_{0} + xx^{+1} f_{0(2\cdot1)(2\cdot2)} + x^{12} f_{2}$ (cost = 5)
TGFE 42: $f = x^{+2}x^{+2} f_{0} + xf_{1} + xx^{+2} f_{0(2\cdot1)(2\cdot2)}$ (cost = 5)
TGFE 43: $f = x^{+1}x^{+2} f_{(2\cdot0)12} + x^{+1}x^{+1} f_{1} + x^{+2}x^{+2} f_{2}$ (cost = 6)
TGFE 44: $f = x^{+1}x^{+1} f_{0} + x^{1} f_{(2\cdot0)12} + x^{12} f_{2}$ (cost = 5)
TGFE 45: $f = x^{+2}x^{+2} f_{0} + xf_{1} + x^{2} f_{(2\cdot0)12}$ (cost = 5)
TGFE 46: $f = x^{+1}x^{+2} f_{(2\cdot0)1(2\cdot2)} + x^{+1}x^{+1} f_{1} + x^{+2} f_{2}$ (cost = 5)
TGFE 47: $f = x^{+1}x^{+1} f_{0} + x^{1} f_{(2\cdot0)1(2\cdot2)} + xxf_{2}$ (cost = 6)
TGFE 48: $f = x^{01} f_{0} + xxf_{1} + xx^{+2} f_{(2\cdot0)1(2\cdot2)}$ (cost = 5)
TGFE 59: $f = x^{+1}x^{+2} f_{(2\cdot0)(2\cdot1)2} + x^{02} f_{1} + x^{+2} f_{2}$ (cost = 4)
TGFE 50: $f = x^{+1} f_{0} + xx^{+1} f_{(2\cdot0)(2\cdot1)2} + xxf_{2}$ (cost = 5)
TGFE 51: $f = x^{+2}x^{+2} f_{0} + xxf_{1} + x^{2} f_{(2\cdot0)(2\cdot1)2}$ (cost = 6)
TGFE 52: $f = x^{+1}x^{+2} f_{(2\cdot0)(2\cdot1)(2\cdot2)} + x^{02} f_{1} + x^{+2} f_{2}$ (cost = 4)
TGFE 53: $f = x^{+1} f_{0} + xx^{+1} f_{(2\cdot0)(2\cdot1)(2\cdot2)} + x^{12} f_{2}$ (cost = 4)
TGFE 54: $f = x^{01} f_{0} + xf_{1} + xx^{+2} f_{(2\cdot0)(2\cdot1)(2\cdot2)}$ (cost = 4)

Proof. Substituting (1), (2), and (3) in (6), we have

$$\begin{aligned} f &= (2x^{2}+1) f_{0} + (2x^{2}+2x) f_{1} + (2x^{2}+x) f_{2} \\ &= (2x^{2}+1) f_{0} + (2x^{2}+2x) f_{1} + [(2x^{2}+2x)+2x] f_{2} \\ &= (2x^{2}+1) f_{0} + (2x^{2}+2x)(f_{1}+f_{2}) + 2xf_{2} \\ &= x^{0} f_{0} + x^{1} f_{12} + x^{12} f_{2} \end{aligned}$$

Thus, we have TGFE 1. We can also prove TGFE 1 using perfect induction as shown below:

If $x = 0$, then $f = x^{0} f_{0} + x^{1} f_{12} + x^{12} f_{2} = 1\cdot f_{0} + 0\cdot f_{12} + 0\cdot f_{2} = f_{0}$.
If $x = 1$, then $f = x^{0} f_{0} + x^{1} f_{12} + x^{12} f_{2} = 0\cdot f_{0} + 1\cdot f_{12} + 2\cdot f_{2} = f_{1} + f_{2} + 2\cdot f_{2} = f_{1}$.
If $x = 2$, then $f = x^{0} f_{0} + x^{1} f_{12} + x^{12} f_{2} = 0\cdot f_{0} + 0\cdot f_{12} + 1\cdot f_{2} = f_{2}$.

Similarly, either using GF(3) polynomial expressions or using perfect induction, we can prove TGFE 2 through TGFE 54.

[End of Proof]

5. Minimization of Ternary Galois Field Sum of Products Expression

In this section, we will discuss minimization of non-reversible ternary logic function expressed as output vector as ternary GFSOP expression by the applications of TGFEs on function variables.

Applications of TGFE 0 on all variables of a ternary logic function produce a canonical GFSOP expression. However, applications of TGFEs 0 through 54 on variables of a ternary logic function produce a non-canonical ternary GFSOP expression that leads to minimization of ternary GFSOP expression. This minimization technique is illustrated in Figure 1 for the sum output S of ternary full-adder function from Table 4, where the don't care outputs are assumed to be 0s. For the variable A, the cofactor f_0 is the part of output vector corresponding to $A = 0$, the cofactor f_1 is the part of output vector corresponding to $A = 1$, and the cofactor f_2 is the part of output vector corresponding to $A = 2$ as shown in Figure 1. Thus, $f_0 = [010120200]^T$, $f_1 = [120200010]^T$, and $f_2 = [200010120]^T$. For the application of TGFE 29 on the variable A, we replace f_0 by $f_{012} = [000000000]^T$, f_1 by $f_{(2\cdot0)1} = [110110110]^T$, and f_2 by f_2 as shown in Figure 1. These replacements produce the transformed vector of column 3 in Figure 1. For the variable B, the cofactor f_0 is the part of transformed vector of column 3 corresponding to $B = 0$, the cofactor f_1 is the part of transformed vector of column 3 corresponding to $B = 1$, and the cofactor f_2 is the part of transformed vector of column 3 corresponding to $B = 2$ as shown in Figure 1. The application of TGFE 30 on the variable B produces the transformed vector of column 4. For the variable C, the cofactor f_0 is the part of transformed vector of column 4 corresponding to $C = 0$, the cofactor f_1 is the part of transformed vector of column 4 corresponding to $C = 1$, and the cofactor f_2 is the part of transformed vector of column 4 corresponding to $C = 2$ as shown in Figure 1. The application of TGFE 29 on the variable C produces the final transformed vector of column 5. The original output vector has 12 non-zero elements, whereas the final transformed vector has only three non-zero elements resulting into three TGFPs in the minimized ternary GFSOP expression. Thus, the ternary GFSOP minimization problem can be considered as selection of TGFEs for the variables that will produce the minimized ternary GFSOP expression for a given ternary logic function. For the sum S output of ternary full-adder function, there are $55^3 = 166{,}375$ possibilities of selecting three TGFEs for three variables and some of them such as (29, 30, 29) will produce the minimized GFSOP expression. If the number of variables of a given ternary logic function is 11, then the number of possibilities of choosing TGFEs for the variables is $55^{11} = 1.393123392 \times 10^{19}$. Thus, ternary GFSOP minimization is an NP-hard problem that requires some heuristic algorithm to find the optimal solution.

Now, we will see writing ternary GFSOP expression for sum output from the final transformed vector of column 5 of Figure 1. The first non-zero element of the final vector is 2, whose input combination is $ABC = 120$. As the value of the variable A is 1, we take the product corresponding to the 1st location of TGFE 29 (corresponding to the cofactor $f_{(2.0)1}$), which is A^{+1}. As the value of the variable B is 2, we take the product corresponding to the 2nd location of TGFE 30 (corresponding to the cofactor f_2), which is 1. As the value of the variable C is 0, we take the product corresponding to the 0th location of TGFE 29, which is $C^{+1}C^{02}$. Then these three products are multiplied with the output constant 2 resulting into the product $2 \cdot A^{+1} \cdot 1 \cdot C^{+1}C^{02} = 2A^{+1}C^{+1}C^{02}$. The constant 2 can be eliminated by multiplying it with any reversible literal of the product term such as $2A^{+1} = A_{02}$ resulting into the product term $A^{02}C^{+1}C^{02}$. Similarly, determining product terms for all non-zero elements and then summing them up, we find the minimized GFSOP expression as follows:

$$S(A,B,C) = A^{02}C^{+1}C^{02} + B^{02}C^{+1}C^{01} + C^{+1} \tag{7}$$

Applications of TGFEs 8, 42, and 32 on variables A, B, and C, respectively, of the carry output C_o of ternary full-adder function from Table 4 produce the following ternary GFSOP expression:

$$C_o(A,B,C) = A^{0}B^{12}B^{+2}C^{1} + ABC^{1} + A^{12}BB^{+2}C^{+1} + A^{12}A^{+2}B^{+2}B^{+2}C^{1} + A^{12}A^{+2}BC^{+1} \tag{8}$$

Input ABC	Output S	Expansion on A using TGFE 29		Expansion on B using TGFE 30		Expansion on C using TGFE 29	
000	0		0		0	$f_0 \Rightarrow f_{012}$	0
001	1		0	$f_0 \Rightarrow f_{0(2\cdot1)}$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
002	0		0		0	$f_2 \Rightarrow f_2$	0
010	1		0		0	$f_0 \Rightarrow f_{012}$	0
011	2	$f_0 \Rightarrow f_{012}$	0	$f_1 \Rightarrow f_{012}$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
012	0		0		0	$f_2 \Rightarrow f_2$	0
020	2		0		0	$f_0 \Rightarrow f_{012}$	0
021	0		0	$f_2 \Rightarrow f_2$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
022	0		0		0	$f_2 \Rightarrow f_2$	0
100	1		1		0	$f_0 \Rightarrow f_{012}$	0
101	2		1	$f_0 \Rightarrow f_{0(2\cdot1)}$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
102	0		0		0	$f_2 \Rightarrow f_2$	0
110	2	$f_1 \Rightarrow f_{(2\cdot0)1}$	1		0	$f_0 \Rightarrow f_{012}$	0
111	0		1	$f_1 \Rightarrow f_{012}$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
112	0		0		0	$f_2 \Rightarrow f_2$	0
120	0		1		1	$f_0 \Rightarrow f_{012}$	2
121	1		1	$f_2 \Rightarrow f_2$	1	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
122	0		0		0	$f_2 \Rightarrow f_2$	0
200	2		2		2	$f_0 \Rightarrow f_{012}$	1
201	0		0	$f_0 \Rightarrow f_{0(2\cdot1)}$	2	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
202	0		0		0	$f_2 \Rightarrow f_2$	0
210	0	$f_2 \Rightarrow f_2$	0		0	$f_0 \Rightarrow f_{012}$	0
211	1		1	$f_1 \Rightarrow f_{012}$	0	$f_1 \Rightarrow f_{(2\cdot0)1}$	0
212	0		0		0	$f_2 \Rightarrow f_2$	0
220	1		1		1	$f_0 \Rightarrow f_{012}$	0
221	2		2	$f_2 \Rightarrow f_2$	2	$f_1 \Rightarrow f_{(2\cdot0)1}$	1
222	0		0		0	$f_2 \Rightarrow f_2$	0

Figure 1. Applications of TGFEs 29, 30, and 29 on variables A, B, and C, respectively, of the sum output S of ternary full-adder function for ternary GFSOP minimization.

In Section 11, we will see realization of minimized ternary GFSOP expressions of (7) and (8).

6. Quantum-Inspired Evolutionary Algorithm for Minimization of Ternary Galois Field Sum of Products Expressions

As the ternary GFSOP minimization is an NP-hard problem, finding optimal solution requires some sort of heuristic algorithm. Moreover, the selection of TGFEs for the function variables for ternary GFSOP minimization is combinatorial in nature. As quantum-inspired evolutionary algorithm (QEA) is found to be very promising for combinatorial problems, it is expected that QEA will also be very suitable for ternary GFSOP minimization.

The proposed QEA for ternary GFSOP minimization maintains three data structures as discussed below:

Population of Q-bit individuals: A Q-bit in a QEA is a classical equivalent of a qubit in a quantum system. A Q-bit is represented as a tuple (α, β) or as a column vector $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ such that $\alpha^2 + \beta^2 = 1$, where α^2 is the probability that the Q-bit is in state 0 and β^2 is the probability that the Q-bit is in state 1. A Q-bit can be visualized as a two-dimensional unit vector, where α is the projection of the unit vector on the x-axis and β is the projection of the unit vector on the y-axis. When the unit vector lies on the x-axis, that is, $\alpha^2 = 1$ and $\beta^2 = 0$, it represents the state 0 and when it lies on the y-axis, that is $\beta^2 = 1$ and $\alpha^2 = 0$, it represents the state 1. The probabilities of a Q-bit being in state 0 or 1 can be changed by rotating the Q-bit. If a Q-bit is rotated towards y-axis, it converges towards state 1 and if it is rotated towards x-axis, it converges towards state 0. The rotation operator that rotates the Q-bit θ radian in the anticlockwise direction is

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

If the Q-bit $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ is rotated using the rotation operator $R(\theta)$, then the new Q-bit is formed as follows:

$$R(\theta)\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha\cos\theta - \beta\sin\theta \\ \alpha\sin\theta + \beta\cos\theta \end{bmatrix} = \begin{bmatrix} \alpha' \\ \beta' \end{bmatrix}. \tag{9}$$

If the Q-bit lies on the x-axis representing the state 0, then $\sin\theta = 0$ and $\cos\theta = \pm 1$. In this situation $R(\theta)\begin{bmatrix} \pm 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm 1 \\ 0 \end{bmatrix}$ and the Q-bit represents the state 0, that is, the rotation operator fails to change the probabilities of the states any more. If the Q-bit lies on the y-axis representing the state 1, then $\sin\theta = \pm 1$ and $\cos\theta = 0$. In this situation $R(\theta)\begin{bmatrix} 0 \\ \pm 1 \end{bmatrix} = \begin{bmatrix} \pm 1 \\ 0 \end{bmatrix}$ and the Q-bit represents the state 0, that is, the rotation operator just flips the state from 1 to 0 and subsequently fails to change the probabilities of the states any more. These facts are very important to choose the value of the rotation angle θ in the Q-bit update process discussed later.

A Q-bit individual of length N is represented as

$$q = [(\alpha_1, \beta_1)(\alpha_2, \beta_2)\cdots(\alpha_N, \beta_N)]. \tag{10}$$

Representation of a decimal number in the range 0 to 54 into binary form requires six bits. If the given ternary logic function has n variables, then the length of the Q-bit individual will be $N = 6 \cdot n$, where the first six Q-bits are for the first variable, the next six Q-bits are for the second variable, and so on. A Q-bit individual of length N eventually represents 2^N binary individuals in superposition. The population of Q-bit individuals at generation t is denoted by $Q(t)$. Readers can see [50] for more details of Q-bit individuals.

Population of observed binary solutions: The selection of TGFEs for the variables for ternary GFSOP minimization can be encoded as a binary string of length $N = 6 \cdot n$

$$x = (x_1 x_2 \cdots x_N), \tag{11}$$

where n is the number of variables of the given function. Decimal equivalent of the first six bits is the TGFE for the first variable, decimal equivalent of the next six bits is the TGFE for the second variable, and so on. The population of binary solutions at generation t is denoted by $X(t)$. The sizes of the population of Q-bit individuals and the population of binary solutions are same. The population of binary solutions $X(t)$ is generated by probabilistic observation of the population of Q-bit individuals $Q(t-1)$. In the probabilistic observation process, the binary value of x_i of an observed binary solution is set to 1 if and only if $\text{random}[0\cdots1] < \beta_i^2$ for the corresponding Q-bit individual, otherwise it is set to 0.

The best binary solution: The best binary solution is denoted by $b = (b_1 b_2 \cdots b_N)$, where $N = 6 \cdot n$ and n is the number of variables of the given function, which stores the best binary solution so far generated.

The structure of the QEA for ternary GFSOP minimization is given below:

```
Procedure QEA for ternary GFSOP minimization
begin
            $t \leftarrow 0$
(i)         initialize $Q(t)$
(ii)        make $X(t)$ by observing $Q(t)$
(iii)       repair $X(t)$
(iv)        evaluate $X(t)$
(v)         store best solution among $X(t)$ into $b$
            while (t < MAX_GEN) do
            begin
                $t \leftarrow t+1$
(vi)            make $X(t)$ by observing $Q(t-1)$
(vii)           repair $X(t)$
(viii)          evaluate $X(t)$
(ix)            update $Q(t)$
(x)             store the best solution among $X(t)$ and $b$ into $b$
            end
end
```

The procedure is discussed below step wise:

Step (i): The Q-bits of all Q-bit individuals of $Q(0)$ are initialized to $\alpha = \beta = 1/\sqrt{2}$, which implies that the probabilities of a Q-bit being in state 0 or 1 are equal.

Step (ii): The population of binary solutions $X(0)$ is generated by probabilistic observation of the population of Q-bit individuals $Q(0)$.

Step (iii): Using six bits, decimal numbers 0 through 63 can be represented. The TGFE numbers are 0 through 54. If the decimal equivalent of a six-bit segment of a binary solution is greater than 54, then randomly selected 1s of that segment are flipped to 0s until the decimal equivalent becomes less than or equal to 54. In this way, each six-bit segment of all binary solutions is repaired.

Step (iv): The evaluation of a binary solution is the most important part of the QEA for ternary GFSOP minimization. The given ternary logic function expressed as output vector is expanded using the TGFEs represented by the binary solution applying the technique discussed in Section 5. The number of 0s in the produced final transformed vector is considered as the first fitness function $f_1(x)$. The sum of the costs of the TGFEs represented by the binary solution is considered as the second fitness function $f_2(x)$. The objective of the QEA is to maximize $f_1(x)$ so that the number of non-zero elements in the final transformed vector is minimized resulting into minimization of number of product terms in the resulting ternary GFSOP expression. In case of tie of $f_1(x)$, the second objective is to minimize $f_2(x)$ so that the resulting circuit requires less number of input lines.

Step (v): The binary solution x among $X(0)$ having the highest $f_1(x)$ value is stored into b. Any tie is broken using the lowest value of $f_2(x)$.

The while loop runs the subsequent generations of the QEA.

Step (vi): The population of binary solutions $X(t)$ is generated by probabilistic observation of the population of Q-bit individuals $Q(t-1)$.

Step (vii): The binary solutions in $X(t)$ are repaired as in Step (iii).

Step (viii): The repaired binary solutions in $X(t)$ are evaluated as in Step (iv).

Step (ix): In this step, the Q-bit individuals are updated to converge towards the stored best solution b by rotating each Q-bit θ radian. If $f_1(x) > f_1(b)$ or $[f_1(x) = f_1(b)] \wedge [f_2(x) < f_2(b)]$ for a binary solution x, then the Q-bits of the corresponding Q-bit individual are not rotated, i.e., rotated 0 radians. If $f_1(x) < f_1(b)$ or $[f_1(x) = f_1(b)] \wedge [f_2(x) \geq f_2(b)]$ for a binary solution x, then there are three situations: (i) if $x_i = b_i$, then the corresponding Q-bit is not rotated, (ii) if $x_i = 0$ and $b_i = 1$, then the corresponding Q-bit is rotated θ radians towards y-axis to converge the Q-bit towards 1, and (iii) if $x_i = 1$ and $b_i = 0$, then the corresponding Q-bit is rotated θ radians towards x-axis to converge the Q-bit towards 0. We have used $\theta = 0.011\pi$ radians as rotation angle for our experiments. This selection of θ ensures that the Q-bit will never lie on any axis. Readers can see [50, 51] for more details of updating Q-bit individuals.

Step (x): The binary solution x among $X(t)$ and b having the highest $f_1(x)$ value is stored into b. Any tie is broken using the lowest value of $f_2(x)$.

7. Some Ternary Primitive Reversible/Quantum Gates

Any unitary operation can theoretically be realized using quantum technologies [4]. Specifically, any d-valued ($d \geq 3$) unitary operation represented by a $d \times d$ unitary matrix, such as ternary unitary operation represented by a 3×3 unitary matrix, can be realized using liquid ion-trap quantum technology [8] as 1-qudit gate. We will represent a ternary 1-qudit gate using the symbol of Figure 2, where A is the input variable, z is any unitary operation, and A^z is the output of the gate after unitary operation z is applied on the input A. The unitary operation z includes any possible unitary operation represented by a 3×3 unitary matrix including the permutation operations corresponding to six ternary reversible literals discussed in Section 3.

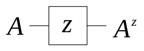

Figure 2. Symbol of ternary 1-qudit gate.

Muthukrishnan and Stroud [8] proposed a liquid ion-trap realizable 2-qudit d-valued ($d \geq 3$) controlled quantum gate family, where a unitary operation is applied on the controlled input when the value of the controlling input is $d-1$. We will refer to this family of gates as Muthukrishnan-Stroud (M-S) gate family. The symbol of ternary M-S gate is shown in Figure 3. The input A is the controlling input and the input B is the controlled input. The output P is exactly equal to the input A (pass through output). The output Q is equal to B^z when $A = 2$ and equal to B when $A \neq 2$, where z is any possible 3×3 unitary operation including permutation operations corresponding to six ternary reversible literals.

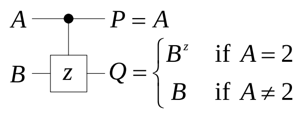
A
$P = A$
B
z
$Q = \begin{cases} B^z & \text{if } A = 2 \\ B & \text{if } A \neq 2 \end{cases}$

Figure 3. Symbol of ternary Muthukrishnan-Stroud (M-S) gate family.

8. Realization of Ternary Literals Using Primitive Quantum Gates

The six ternary reversible literals discussed in Section 3 are permutation transforms of input values to output values, which are unitary in nature. These reversible literals can be realized using 1-qudit gates.

We have proposed here a method for realization of ternary 1-RPLs using 1-qudit and M-S gates as shown in Figure 4. The value of a is chosen in such a way that when the value of $A = i$, then the controlling value of the M-S gate is $x = i + a = 2$ to produce $A^i = 1$. The value of b is chosen in such a way that $a + b = 0$ to restore the value of A at the output. The values of a and b for realization of ternary 1-RPLs are given in Table 6. The realizations of ternary1-RPLs require three primitive gates and one ancilla input constant (*any additional input other than the primary input used in the design is called ancilla input*). However, the reader can readily verify that the ternary 1-RPL A^2 can be realized using only one M-S gate.

Table 6. Values of a and b for realization of ternary 1-RPLs.

$A = i$	a	b
0	2	1
1	1	2
2	0	0

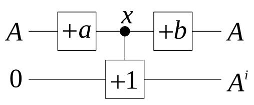

Figure 4. Realization of ternary1-RPLs using quantum gates.

9. Ternary Feynman Gate

The symbol of ternary Feynman gate is shown in Figure 5, where A, B are inputs and $P = A$, $Q = A + B$ are outputs. Feynman gate is a macro-level gate and needs to be realized on the top of 1-qudit and M-S gates.

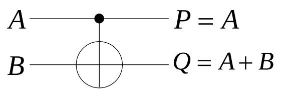

Figure 5. Symbol of ternary Feynman gate.

For realization of ternary Feynman gate on the top of 1-qudit and M-S gates, we propose the architecture of Figure 6. The architecture does not require any ancilla input constant. When $A = 0$, the output $Q = 0 + B = B$ and we need not to apply any operation on the input B. When $A = i \in \{1,2\}$, we need to apply $+i$ operation on B using M-S gate to make the output $Q = i + B = A + B$. For doing this, we need to make the corresponding controlling value of the M-S gate equal to 2 by providing an effective shift of $+x_i$ to the input $A = i$ at the controlling point. This implies that

$$(\forall i \in \{1,2\})\ i + x_i = 2\,. \tag{12}$$

This effective shift to the input $A = i$ is made by cascaded 1-qudit gates, which requires that

$$(\forall i \in \{1,2\})\ x_i = x_{i-1} + y_i\,. \tag{13}$$

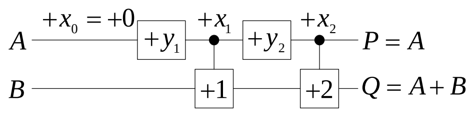

Figure 6. Architecture for realization of ternary Feynman gate.

Solving (12) and (13), we have the values of y_i and x_i as given in Table 7. Using these values of y_i and x_i, the realizations of ternary Feynman gate is shown in Figure 7. Realization of ternary Feynman gate requires four primitive gates.

Table 7. Values of y_i and x_i for realization of ternary Feynman gate using architecture of Figure 6.

$A=i$	y_i	x_i
0		0
1	1	1
2	2	0

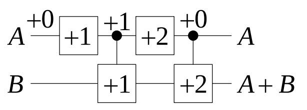

Figure 7. Realization of ternary Feynman gate.

10. Ternary Toffoli Gate

The symbol of ternary 3-qudit Toffoli gate is shown in Figure 8, where A, B, C are inputs and $P=A$, $Q=B$, $R=C+AB$ are outputs. Toffoli gate is a macro-level gate and needs to be realized on the top of 1-qudit and M-S gates.

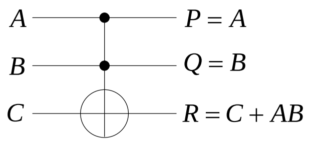

Figure 8. Symbol of ternary 3-qudit Toffoli gate.

For realization of ternary 3-qudit Toffoli gate on the top of 1-qudit and M-S gates, we propose the architecture of Figure 9. The proposed architecture does not require any ancilla input constant. In the architecture of Figure 9, the inputs A and B are used as the controlling line and the controlled line, respectively, for circuits 1, 2, and 3. The first circuit produces $(\forall A)\ X=xB$, where $x\in\{1,2\}$. The second circuit produces $(\forall A)\ Y=yX=xyB$, where $y\in\{1,2\}$. The third circuit produces $(\forall A)\ Q=zY=xyzB$, where $z\in\{1,2\}$. The condition $(\forall A)\ Q=xyzB=B$ requires that

$$(\forall A\in\{0,1,2\})\ xyz=1\,. \tag{14}$$

The two Feynman gates in the architecture of Figure 9 produce $R=C+X+Y=C+(x+xy)B=C+AB$, which requires that

$$(\forall A\in\{0,1,2\})\ x+xy=A\,. \tag{15}$$

Solving (14) and (15), we get the values of x, y, and z as given in Table 8. Based on these values, the realization of ternary 3-qudit Toffoli gate is shown in Figure 10. There are some 1s in Table 8. Multiplications by these 1s are not required, since they do not change the values. The other multiplications are done using M-S gates. For this purpose, we need to provide controlling value of 2 at the controlling points for all values of A by using cascaded 1-qudit gates similar to that done for Feynman gate realization. Ternary $2\times$ operation can be realized using 12 permutation operation. Realization of ternary 3-qudit Toffoli gate requires 8 + 2×4 = 16 primitive gates and no ancilla input constant.

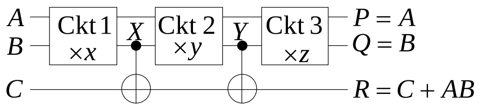

Figure 9. Architecture for realization of ternary 3-qudit Toffoli gate.

Table 8. Values of x, y, and z for realization of ternary Toffoli gates using the architecture of Figure 9.

A	x	y	z
0	1	2	2
1	2	1	2
2	1	1	1

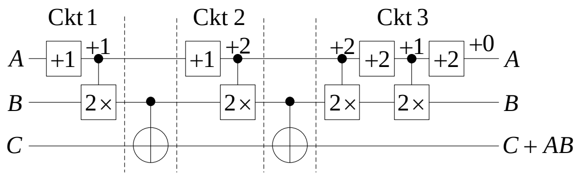

Figure 10. Realization of ternary 3-qudit Toffoli gate.

Ternary Toffoli gates with more than three qudits are often used in the ternary GFSOP synthesis. The symbol of ternary 4-qudit Toffoli gate is shown in Figure 11. For realization of ternary 4-qudit Toffoli gate on the top of 1-qudit and M-S gates, we propose the architecture of Figure 12. In the architecture of Figure 12, the input A is used as the controlling line and the input C is used as the controlled line for circuits 1, 2, and 3. The input B remains unchanged in all three circuits. The first circuit produces $(\forall A)\ X = xC$, where $x \in \{1,2\}$. The second circuit produces $(\forall A)\ Y = xyC$, where $y \in \{1,2\}$. The third circuit produces $(\forall A)\ R = xyzC$, where $z \in \{1,2\}$. The condition $(\forall A)\ R = xyzC = C$ requires that

$$(\forall A \in \{0,1,2)\ xyz = 1\,. \tag{16}$$

The two 3-qudit Toffoli gates in Figure 12 produce $S = D + BxC + BxyC = D + (x + xy)BC = D + ABC$, which requires that

$$(\forall A \in \{0,1,2\})\ x + xy = A\,. \tag{17}$$

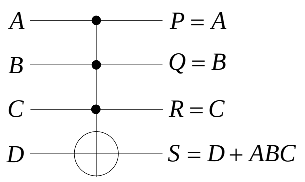

Figure 11. Symbol of ternary 4-qudit Toffoli gate.

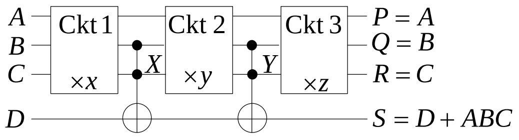

Figure 12. Architecture for realization of ternary 4-qudit Toffoli gate.

Equations (16) and (17) are exactly the same as (14) and (15) and their solutions are as shown in Table 8. Based on these values, the realization of ternary 4-qudit Toffoli gate is shown in Figure 13. Realization of ternary 4-qudit Toffoli gate requires 8 + 2×16 = 40 primitive gates

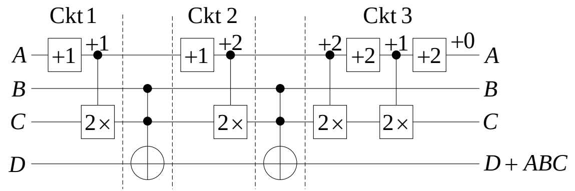

Figure 13. Realization of ternary 4-qudit Toffoli gate.

The architecture of Figure 12 can be generalized for realization of ternary n-qudit ($n \geq 3$) Toffoli gate using $(n-1)$-qudit Toffoli gates as shown in Figure 14. Readers should note that the Feynman gate can be considered as a 2-qudit Toffoli gate. The input A_1 is the controlling input and the input A_{n-1} is the controlled input in all three circuits. The inputs A_2 through A_{n-2} are unchanged. The two Toffoli gates are $(n-1)$-qudit Toffoli gates. The values of x, y, and z are as shown in Table 8. *Realization of ternary* n*-qudit* ($n \geq 3$) *Toffoli gate requires 8 + 2×[number of primitive gates for ternary* $(n-1)$ *-qudit Toffoli gate] primitive gates.*

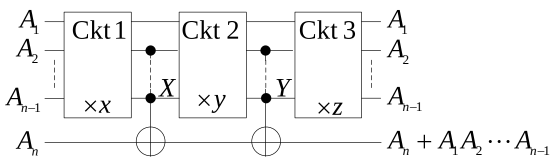

Figure 14. Architecture for realization of ternary n-qudit ($n \geq 3$) Toffoli gate.

11. Realization of Ternary Multiple-Output GFSOP Expressions

Multiple-output ternary GFSOP expressions can be realized as a cascade of 1-qudit, M-S, Feynman, and Toffoli gates. In the GFSOP realizations, the 1-RPLs are realized using 1-qudit and M-S gates and the reversible literals are realized using 1-qudit gates as discussed in Section 8. The product terms are generated and summed with other product terms using Toffoli gates. For realization of literals, necessary copies of the concerned input is made using Feynman gate. To reduce the number of additional copies of the input, we will generate the literals along the same copy of the input. To reduce the number of 1-qudit gates, we will use a literal as many times as possible before changing it to another literal. For this purpose, we have developed heuristic technique for literal and product ordering as discussed below using the GFSOP expressions of ternary full-adder function as given in (7) and (8):

Step 1: Determine the literal counts for each variable in the given GFSOP expressions. Order the literals for each variable in the decreasing order of count. Break the tie arbitrarily. The literal counts and their orders for GFSOP expressions of (7) and (8) are given in Table 9. If any product term has more than one literals of the same variable, then rearrange the literals in the given GFSOP expressions in the increasing order. The product terms of the GFSOP expressions of (7) and (8) with literal rearrangements are shown in Table 10. This rearrangement of the literals will make the most frequent literal appears first in the product term.

Step 2: Determine the score for each product term of the given GFSOP expressions as sum of its literal counts. Order the product terms in the decreasing order of scores. Break the tie using decreasing number of matches with the preceding product term. The scores and the orders of the product terms of the GFSOP expressions of (7) and (8) are given in Table 10. The product terms $A^{12}A^{+2}B^{+2}B^{+2}C^{1}$, $A^{12}B^{+2}BC^{+1}$, $A^{12}A^{+2}BC^{+1}$, and $A^{0}B^{+2}B^{12}C^{1}$ have scores 16, 15, 13, and 9, respectively, and have been assigned orders 1, 2, 3, and 4, respectively, according to their decreasing order of scores. The product terms $A^{02}C^{+1}C^{02}$, $B^{02}C^{+1}C^{01}$, and ABC^{1} have the same score 7. The numbers of matches of the product terms $A^{02}C^{+1}C^{02}$, $B^{02}C^{+1}C^{01}$, and ABC^{1} with the preceding product term $A^{0}B^{+2}B^{12}C^{1}$ (order

4) are 0, 0, and 1, respectively. Therefore, the product term ABC^{1} has been assigned order 5 and the product terms $A^{02}C^{+1}C^{02}$ and $B^{02}C^{+1}C^{01}$ have been assigned orders 6 and 7, respectively. Finally, the product term C^{+1} having score 5 has been assigned order 8. In the GFSOP realization, the product terms are implemented in the increasing order as discussed below.

Table 9. Literal ordering for each variable of the GFSOP expressions of (7) and (8).

Variable A			Variable B			Variable C		
Literal	Count	Order	Literal	Count	Order	Literal	Count	Order
A^{02}	1	3	B^{02}	1	3	C^{+1}	5	1
A^{0}	1	4	B^{12}	1	4	C^{02}	1	3
A	1	5	B^{+2}	4	1	C^{01}	1	4
A^{12}	3	1	B	3	2	C^{1}	3	2
A^{+2}	2	2						

Table 10. Product term ordering of the GFSOP expressions of (7) and (8).

Product Term	Function	Score	Order
$A^{02}C^{+1}C^{02}$	S	7	6
$B^{02}C^{+1}C^{01}$	S	7	7
C^{+1}	S	5	8
$A^{0}B^{+2}B^{12}C^{1}$	C_o	9	4
ABC^{1}	C_o	7	5
$A^{12}B^{+2}BC^{+1}$	C_o	15	2
$A^{12}A^{+2}B^{+2}B^{+2}C^{1}$	C_o	16	1
$A^{12}A^{+2}BC^{+1}$	C_o	13	3

This sort of literal rearrangement and product term ordering will ensure that most frequent literals appear in consecutive product terms and the literals can be realized once and used for generating the concerned product terms. This will help reduce number of 1-qudit gates needed to generate the literals as discussed in GFSOP synthesis method given below.

We have developed a method for synthesis of ternary multiple-output GFSOP expressions using cascade of 1-qudit, M-S, Feynman, and Toffoli gate as discussed below:

Step 1: Count the maximum number of literals of each variable appearing in any product term of the GFSOP expressions. If these numbers are more than one, then create additional copies of the variables using Feynman gate. In the GFSOP expressions of (7) and (8), the maximum number of literals of all variables appearing in any product term is two and, therefore, an additional copy of each variable is created as shown in Figure 15. If any variable has 1-RPLs in the GFSOP expression, then add an input constant 0 line with that variable as shown in Figure 15 for variable A and C. If constant 2 cannot be eliminated from the GFSOP expressions, then add constant 2 line. For this specific example, such constant 2 input line is not needed. For realizing the outputs, add constant 0 lines for each output as shown in Figure 15 for sum output S and carry output C_o.

Step 2: Realize the product terms according to their increasing order as determined in Table 10. For the first product term $A^{12}A^{+2}B^{+2}B^{+2}C^{1}$, generate the reversible literals A^{12}, A^{+2}, B^{+2}, and B^{+2} by using 12, +2, +2, and +2 1-qudit gates, respectively, along the two A input lines and two B input lines. Generate the 1-RPL C^{1} using the method discussed in Section 8, that is, generate the effective shift +1 of the input C by using a +1 1-qudit gate along the first C input line and then add an M-S gate with +1 operation between the controlling line C and the constant 0 input line dedicated for generating 1-RPLs as shown in Figure 15. The five literals are then multiplied by a 6-qudit Toffoli gate and added to the constant 0 line dedicated for the carry output C_o as shown in Figure 15. For the second product term $A^{12}B^{+2}BC^{+1}$, the reversible literals A^{12}, B^{+2}, and C^{+1} (during generation of 1-RPL C^{1}) are already generated. Generate the reversible literal B along the second B input line using a +1 1-qudit gate in cascade with the previous +2 1-qudit gate. The new 1-qudit gate can be determined as shown in Table 11. In Table 11, the first column lists all possible values of input B, the second column lists the values produced by the previous reversible literal B^{+2}, and the third column lists the values to be produced by the needed reversible literal B. Now, we need a 1-qudit gate that will generate the output of the third column taking input of the second column. A closer inspection of the second and third columns shows that the required 1-qudit gate is +1. The already generated three reversible literals A^{12}, B^{+2}, C^{+1} and the newly generated reversible literal B are then multiplied to produce the product term $A^{12}B^{+2}BC^{+1}$ and added this to the previous product term $A^{12}A^{+2}B^{+2}B^{+2}C^{1}$ using another 5-qudit Toffoli gate as shown in Figure 15. The other product terms of the GFSOP expressions are generated and summed in the similar ways as shown in Figure 15.

Table 11. Generation of the reversible literal B from the reversible literal B^{+2}.

Input B	Output of the previous reversible literal B^{+2}	Output of the reversible literal B to be generated
0	2	0
1	0	1
2	1	2

Step 3. Restore the inputs at the output to make the quantum circuit coherent. The 1-RPL generated for the variable A is A^0. Generation of 1-RPL A^0 is done using an M-S gate with +2 effective shift of the variable A as controlling value and a +1 operation in the M-S gate. If we add another M-S gate with +2 effective shift of the variable A as controlling value and a +2 operation of the M-S gate, then the constant 0 input dedicated for generating 1-RPL will be restored as shown in Figure 15, since +1 + 2 = 0. The constant 0 input dedicated for generating 1-RPL of C input is also restored similarly as shown in Figure 15. The last effective shift of the variable A is +2. If we add a +1 1-qudit gate along the A line, then the variable A will be restored as shown in Figure 15, since +2 + 1 = 0. Restorations of other inputs including the copies are done similarly as shown in Figure 15. The copy of the variable A is generated by adding the input A with constant 0 input using a Feynman gate, since $A+0=A$. If we add two Feynman gates between input A line and copy A line (after restoration of the copy of A) the constant 0 will be restored as shown in Figure 15, since $A+A+A=0$. Input constant 0s for creating copies of other variables are restored similarly as shown in Figure 15. Input restoration is important for making the quantum circuit coherent and to reuse the same inputs in other parts of a larger quantum circuit. For example, this sort of input reuse is done in the oracles of Grover's quantum search algorithm [4, 5].

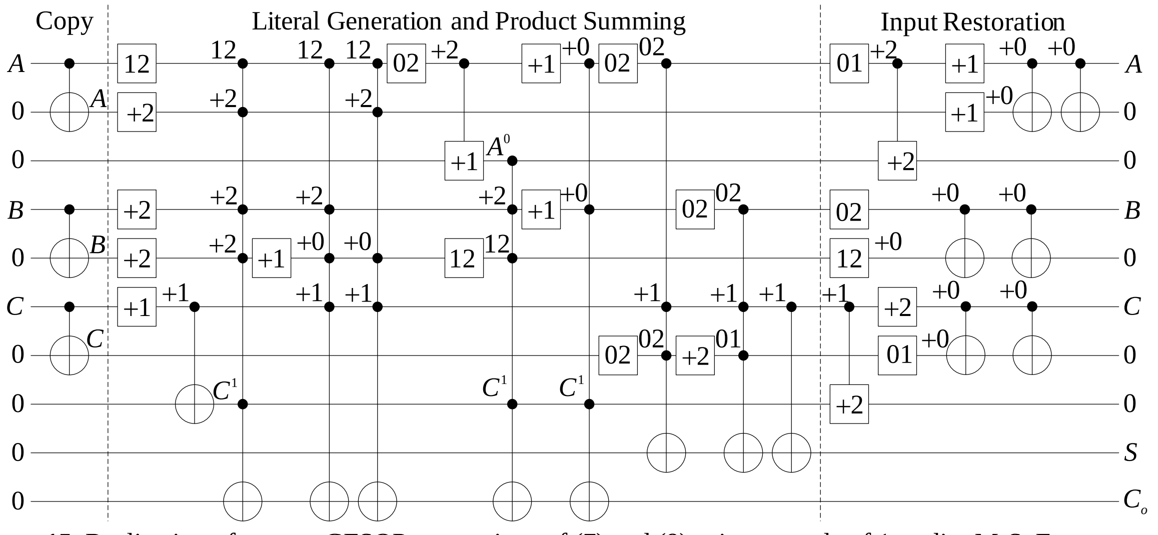

Figure 15. Realization of ternary GFSOP expressions of (7) and (8) using cascade of 1-qudit , M-S, Feynman , and Toffoli gates.

The ternary GFSOP synthesis technique proposed here ensures that the literal that is used more than once is tried to be generated once and used in concerned product terms generations. This will eventually reduce the number of 1-

qudit gates needed. Generating the literals along the same line, rather than using many copies of the variable, will reduce the width of the synthesized circuit.

12. Experimental Results

We have implemented the QEA for ternary GFSOP minimization using C language on a PC. We have experimented with the following functions:

prod*n*: input $x_0, x_1, \cdots, x_{n-1}$; output $y = (x_0 x_1 \cdots x_{n-1})$ GF(3). [Output is the GF(3) product of n input variables.]
sum*n*: input $x_0, x_1, \cdots, x_{n-1}$; output $y = (x_0 + x_1 + \cdots + x_{n-1})$ GF(3). [Output is the GF(3) sum of n input variables.]
sqsum*n*: input $x_0, x_1, \cdots, x_{n-1}$; output $y = \left(x_0^2 + x_1^2 + \cdots + x_{n-1}^2\right)$GF(3). [Output is the GF(3) sum of squares of n input variables.]
***n*cy*r*:** input $x_0, x_1, \cdots, x_{n-1}$; output $y = x_0 x_1 \cdots x_{r-1} + x_1 x_2 \cdots x_r + \cdots + x_{n-1} x_0 \cdots x_{r-2}$ GF(3). [A ternary sum of products function of n input variables expressed as GF(3) value, where the product terms consist of r input variables in cyclic order. For example, 3cy2 is $y(a,b,c) = (ab + bc + ca)$ GF(3).]
hadd: Ternary half-adder.
hsub: Ternary half-subtractor.
fadd: Ternary full-adder. For carry input $C = 1$, outputs are assumed to be 0.
fsub: Ternary full-subtractor. For borrow input $C = 1$, outputs are assumed to be 0.

In our experiments with the QEA for ternary GFSOP minimization, we arbitrarily take MAX_GEN = 500 and vary population size from 5 to 100 in 5 steps, that is, 5, 10, 15, …, 95, 100. The Q-bits are updated with probability 1. Then from 20 results corresponding to 20 population sizes, the first minimum result is taken as the minimum result.

Experimental results of 20 ternary functions with up to 11 inputs and two outputs are shown in Table 12. From Table 12, we see that functions **prod*n***, **sum*n***, **sqsum*n***, ***n*cy*r***, *S* output of **hadd**, *D* output of **hsub**, *S* output of **fadd**, and *D* output of **fsub** functions produce exact minimum results in terms of number of products. Knowledge of exact minimum solutions of C_o output of hadd, B_o output of hsub, C_o output of fadd, and B_o output of fsub are not known, and, therefore, the present results cannot be compared. In comparison with results from [22], **5cy3** and **fadd** functions produce better results than [22]. The results of other functions that are common with [22] produce the same results as [22].

From Table 12, we see that the population sizes and the generations required for finding the minimum ternary GFSOP expressions are quite small. Generation of the minimum results with these small QEA parameters shows that QEA works quite satisfactorily and effectively for finding the minimum ternary GFSOP solutions.

13. Conclusion

Multiple-valued logic has many advantages over binary logic. The major advantage is that multiple-valued encoded realization of binary logic function is more compact than original binary realization. Multiple-valued quantum gates are realizable in current quantum technologies [7-14] and their costs are almost similar to the cost of binary quantum gates [9]. As the multiple-valued encoded realization of binary function is more compact, its quantum realization cost will be lesser than that of binary realization. This advantage makes multiple-valued reversible/quantum logic synthesis very promising. Successful methods for synthesis of multiple-valued quantum logic circuit also creates possibility of developing multiple-valued quantum algorithms as well as using multiple-valued quantum circuits internally in the binary quantum algorithms.

Multiple-valued logic functions can be very easily represented as GFSOP expression and the GFSOP expression can be realized as cascade of reversible gates. This approach uses Feynman and multiple-input Toffoli gates for the cascade. Previously, multiple-input Toffoli gate realization required many ancilla constants [36, 43, 44] making the realized GFSOP based reversible/quantum logic circuit very wide. Ancilla input constant free realizations of Feynman and multiple-input Toffoli gates on the top of 1-qudit and M-S gates as presented in this paper makes this GFSOP based synthesis more promising and practical than the earlier.

Table 12. Experimental results.

Function name	# of input	# of output	# of products in canonical GFSOP	# of products in minimized GFSOP	Population size	Generation required	TGFEs (from left to right variables)	# of products in minimized GFSOP from [22]
prod5	5	1	32	1	5	127	33, 32, 25, 2, 25	1
prod10	10	1	1024	1	5	308	44, 33, 44, 53, 1, 25, 25, 26, 7, 7	1
prod11	11	1	2048	1	5	301	44, 26, 8, 2, 26, 32, 53, 54, 8, 33, 32	-
sum5	5	1	162	5	5	67	29, 27, 29, 27, 26	5
sum10	10	1	39366	10	10	112	26, 26, 28, 28, 25, 25, 28, 27, 30, 29	10
sum11	11	1	118098	11	5	237	28, 29, 25, 25, 30, 29, 26, 29, 26, 29, 27	-
sqsum5	5	1	162	5	5	202	25, 26, 26, 25, 25	5
sqsum10	10	1	39528	10	35	170	26, 25, 26, 26, 26, 25, 25, 26, 25, 26	10
sqsum11	11	1	118098	11	55	286	25, 25, 26, 25, 26, 26, 26, 25, 25, 25, 26	-
5cy2	5	1	162	5	10	75	26, 26, 26, 25, 25	5
5cy3	5	1	102	5	60	92	26, 25, 26, 26, 25	7
5cy4	5	1	102	5	10	230	26, 26, 25, 32, 33	5
10cy2	10	1	39204	10	55	197	25, 25, 25, 25, 25, 25, 26, 26, 26, 26	-
10cy3	10	1	35802	10	35	249	26, 26, 26, 26, 25, 25, 26, 26, 26, 25	-
11cy2	11	1	118098	11	60	299	25, 26, 26, 25, 25, 25, 26, 26, 25, 26, 25	-
11cy3	11	1	109782	11	40	249	26, 25, 25, 26, 26, 26, 25, 25, 25, 26, 26	-
hadd	2	2	9 (S: 6, C_o: 3)	4 (S: 2, C_o: 2)	S: 5 C_o: 5	S: 15 C_o: 36	S: 25, 25 C_o: 33, 33	4
hsub	2	2	9 (D: 6, B_o: 3)	4 (D: 2, B_o: 2)	D: 5 B_o: 5	D: 15 B_o: 2	D: 25, 25 B_o: 29, 53	-
fadd	3	2	21 (S: 12, C_o: 9)	8 (S: 3, C_o: 5)	S: 5 C_o: 10	S: 46 C_o: 218	S: 29, 30, 29 C_o: 8, 42, 32	10
fsub	3	2	21 (D: 12, B_o: 9)	8 (D: 3, B_o: 5)	D: 5 B_o: 5	D: 52 B_o: 84	D: 30, 26, 29 B_o: 49, 8, 32	-

The ternary GFSOP synthesis technique proposed here ensures that the literal that is used more than once is tried to be generated once and used in concerned product terms generations. This will eventually reduce the number of 1-qudit gates needed. Generating the literals along the same line, rather than using many copies of the variable, will reduce the width of the synthesized circuit.

Experimental results with 20 ternary logic functions having up to 11 inputs and two outputs show that the proposed QEA for ternary GFSOP minimization works very effectively for finding the minimum solutions. The proposed

QEA also produces better minimization for two functions than our previous heuristic minimization technique of [22].

The proposed ternary GFSOP synthesis method implicitly converts a non-reversible function into a reversible function for synthesis using reversible/quantum gates.

The proposed ternary GFSOP minimization method works with completely specified functions. In our experiments, we have replaced the don't care outputs by 0s. Further study can be done on value assignment for don't care outputs such that the method can take advantage of don't cares to further minimization of the incompletely specified functions.

Our primary intention was to minimize ternary GFSOP expressions using the proposed QEA. For this reason, we did not experimented with different values of the QEA parameters. Further experiments can be done by varying rotation angle, probability of Q-bit update, value of MAX_GEN, and other relevant parameters of QEA.

As the realizations of ternary Feynman and Toffoli gates do not require any ancilla input constant and the ternary GFSOP synthesis method presented here uses as minimum copies of a variable as possible, the proposed synthesis method is suitable for quantum circuit synthesis with less ancilla input constants, which is a favorable and desirable attempt, since quantum register width is a major constraint in quantum technology.

References

[1] R. Landauer, "Irreversibility and heat generation in the computational process," *IBM Journal of Research and Development*, vol. 5, 1961, pp. 183-191.

[2] V.V. Zhirnov, R.K. Kavin, J.A. Hutchby, and G.I. Bourianoff, "Limits to binary logic switching – a Gedanken model," *Proceeding of the IEEE*, vol. 91, no. 11, 2003, pp. 1934-1939.

[3] C.H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, 1973, pp. 525-532.

[4] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of 28th ACM Symposium on Theory of Computing*, 1996, pp. 212-219.

[6] P.W. Shor, "Algorithm for quantum computation: discrete logarithms and factoring," *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, 1994.

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

[8] A. Muthukrishnan and C.R. Stroud Jr., "Multivalued logic gates for quantum computation," *Physical Review A.* vol. 62, 2000, 052309/1-8.

[9] S. Bartlett, D. deGuise, and B. Sanders, "Quantum encodings in spin systems and harmonic oscillators," *Physical Review A 65*, 052316, 2002.

[10] R. Das, A. Mitra, V. Kumar, and A. Kumar, "Quantum information processing by NMR: Preparation of pseudo pure states and implementation of unitary operations in a single-qutrit system," *arXiv:quant-ph/0307240v1*, 31 July 2003.

[11] A. B. Klimov, R. Guzman, J.C. Retamal, and C. Saavedra, "Qutrit quantum computer with trapped ions," *Physical Review A*, vol. 67, 062313, 2003.

[12] J. Daboul, X. Wang, and B.C. Sanders, "Quantum gates hybrid qudits," *Journal of Physics A: Mathematical and General*, vol. 36, no. 14, 2003, pp. 7063-7078.

[13] F. Kunio, "The controlled-U and unitary transformations in two-qudit," *arXiv:quant-ph/0304078*, 2003.

[14] D. McHugh and J. Twamley, "Trapped-ion qutrit spin molecule quantum computer," *http://arXiv.org/abs/quant-ph/0506031*.

[15] M. Perkowski, A. Al-Rabadi, and P. Kerntopf, "Multiple-valued quantum logic synthesis," *Proceedings of 2002 International Symposium on New Paradigm VLSI Computing*, Sendai, Japan, December 12-14, 2002, pp. 41-47.

[16] M.H.A. Khan, M.A. Perkowski, and P. Kerntopf, "Multi-output Galois field sum of products synthesis with new quantum cascades," *Proceedings of 33rd IEEE International Symposium on Multiple-Valued Logic (ISMVL 2003)*, Tokyo, Japan, 16-19 May 2003, 146-153.

[17] A.N. Al-Rabadi, "Quantum circuit synthesis using classes of GF(3) reversible fast spectral transforms," *Proceedings of 34th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2004)*, Toronto, Canada,

19-22 May 2004, pp. 87-93.

[18] E. Curtis and M. Perkowski, "A transformation based algorithm for ternary reversible logic synthesis using universally controlled ternary gates," *Proceeding of International Workshop on Logic and Synthesis (IWLS 2004)*, Tamecula, California, USA, 2-4 June 2004.

[19] N. Denler, B. Yen, M. Perkowski, and P. Kerntopf, "Synthesis of reversible circuits from a subset of Muthukrishnan-Stroud quantum multi-valued gates," *Proceeding of International Workshop on Logic and Synthesis (IWLS 2004)*, Tamecula, California, USA, 2-4 June 2004.

[20] M.H.A. Khan and M.A. Perkowski, **"**Genetic algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates," *Proceedings of 2004 IEEE Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, USA, 19-23 June 2004, pp. 2194-2201.

[21] S.S. Bullock, D.P. O'Leary, and G.K. Brennen, "Asymptotically optimal quantum circuits for d-level systems," *Physical Review Letters*, vol. 94, 230502, 2005.

[22] M.H.A. Khan, M.A. Perkowski, M. R. Khan, P. Kerntopf, "Ternary GFSOP minimization using Kronecker decision diagrams and their synthesis with quantum cascades," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 11, 2005, pp. 567-602.

[23] B. Yen, P. Tomson, and M. Perkowski, "Sum of non-disjoint cubes covering generation for multi-valued systems of base 2, for use in Muthukrishnan-Stroud quantum realizable gates: an extension of the EXOR covering problem," *Proceedings of International Workshop on Logic and Synthesis (IWLS 2005).*

[24] M.H.A. Khan and M.A. Perkowski, "Quantum realization of ternary encoder and decoder," *Proceedings of 7th International Symposium on Representations and Methodology of Future Computing Technologies (RM 2005),* Tokyo, Japan, 5-6 September, 2005.

[25] M.M.R. Khan**,** M.H.A. Khan, and M.M. Akbar, "Evolutionary algorithm based synthesis of multi-output ternary reversible circuits using quantum cascades," *Proceedings of 7th International Symposium on Representations and Methodology of Future Computing Technologies (RM2005)*, Tokyo, Japan, 5-6 September, 2005.

[26] G. Yang, X. Song, M. Perkowski, and J. Wu, "Realizing ternary quantum switching networks without ancilla bits," *arXiv:quant-ph/0509192v1*, 27 Sep 2005.

[27] F.S. Khan and M. Perkowski, "Synthesis of ternary quantum logic circuits by decomposition," *arXiv:quant-ph/0511041v1*, 4 Nov 2005.

[28] D.M. Miller, D. Maslove, and G.W. Dueck, "Synthesis of quantum multiple-valued circuits," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 12, no. 5-6, 2006.

[29] M.H.A. Khan, "Design of reversible/quantum ternary multiplexer and demultiplexer," *Engineering Letters,* vol. 13, no. 2, 2006, pp. 65-69.

[30] M.M.R. Khan, M.G.A. Sujan, and M.H.A. Khan, "Post-EA simplification of ternary reversible circuit," *Proceedings of 9th International Conference on Computer and Information Technology (ICCIT 2006),* Dhaka, Bangladesh, 21-23 December 2006.

[31] P. Kerntopf, M.A. Perkowski, and M.H.A. Khan, "On universality of general reversible multiple-valued logic gates", *Journal of Multiple-Valued Logic and Soft Computing*, vol. 12, no. 5-6, 2006, pp. 417-429.

[32] M.H.A. Khan and M.A. Perkowski, "Quantum ternary parallel adder/subtractor with partially-look-ahead carry," *Journal of System Architecture*, vol. 53, no. 7, 2007, pp. 453-464.

[33] M.H.A. Khan, "Design of reversible/quantum ternary comparator circuits," *Engineering Letters*, vol. 16, no. 2, 2008, pp. 178-184.

[34] R. Khanum, T. Kamal, and M.H.A. Khan, "Genetic algorithm based synthesis of ternary reversible/quantum circuit," *Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008)*, 25-27 December 2008, Khulna, Bangladesh, pp. 270-275.

[35] M.M.M. Khan, A.K. Biswas, S. Chowdhury, M. Hasan, and A.I. Khan, "Synthesis of GF(3) based reversible/quantum logic circuits without garbage output," *Proceedings of 39th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2009)*, Naha, Okinawa, Japan, 21-23 May 2009, pp. 98-102.

[36] M.H.A. Khan and M.A. Perkowski, "GF(4) based synthesis of quaternary reversible/quantum logic circuits," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 13, 2007, pp. 583-603.

[37] M.H.A. Khan, "Reversible realization of quaternary decoder, multiplexer, and demultiplexer circuits," *Engineering Letters*, vol. 15, no. 2, 2007, pp. 203-207.

[38] M.H.A. Khan, "Synthesis of incompletely specified multi-output quaternary function using quaternary quantum gates", *Proceedings of 10th International Conference on Computer and Information Technology (ICCIT 2007),* Dhaka, Bangladesh, 27-29 December 2007.

[39] M.H.A. Khan, "A recursive method for synthesizing quantum/reversible quaternary parallel adder/subtractor with look-ahead carry," *Journal of Systems Architecture*, vol. 54, no. 12, 2008, pp. 1113-1121.

[40] M.H.A. Khan, “Synthesis of quaternary reversible/quantum comparators,” *Journal of Systems Architecture*, vol. 54, no. 10, 2008, pp. 977-982.

[41] M.H.A. Khan, N.K. Siddika, and M.A. Perkowski, “Minimization of quaternary Galois field sum of products expression for multi-output quaternary logic function using quaternary Galois field decision diagram,” *Proceedings of 38th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2008),* 22-24 May 2008, Dallas, TX, USA, pp. 125-130.

[42] M.H.A. Khan, “Scalable architecture for synthesis of reversible quaternary multiplexer and demultiplexer circuits,” *Proceedings of 39th IEEE International Symposium on Multiple-Valued Logic (ISMVL2009)*, 21-23 May 2009, Naha, Okinawa, Japan, pp. 343-348.

[43] M.H.A. Khan, “Reversible synthesis of quinary logic function,” *18th International Workshop on Post-Binary ULSI Systems (ULSIWS2009)*, 20 May 2009, Naha, Okinawa, Japan.

[44] M.H.A. Khan and R. Hasan, “Minimized reversible synthesis of non-reversible quinary logic function,” *Proceedings of 12th International Conference on Computer and Information Technology (ICCIT 2009),* Dhaka, Bangladesh, 21-23 December 2009.

[45] M.H.A. Khan, “Quantum realization of ternary Toffoli gate using ion-trap realizable Muthukrishnan-Stroud primitive gates,” *Proceedings of 7th International Conference on Computer and Information Technology (ICCIT 2004)*, Dhaka, Bangladesh, 26-28 Dec 2004, pp. 369-371.

[46] M.H.A. Khan, “Quantum realization of quaternary Feynman and Toffoli gates,” *Proceedings of International Conference on Electrical and Computer Engineering (ICECE 2006)*, Dhaka, Bangladesh, 19-21 December, 2006, pp. 157-160.

[47] A.I. Khan, N. Nusrat, S.M. Khan, M. Hasan, and M. H. Khan, “Quantum realization of some ternary circuits using Muthukrishnan-Stroud gates”, *Proceedings of 37th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2007)*, 14-16 May 2007, Oslo, Norway.

[48] M.H.A. Khan, “Quantum realization of multiple-valued Feynman and Toffoli gates without ancilla input,” *Proceedings of 39th IEEE International Symposium on Multiple-Valued Logic (ISMVL2009)*, 21-23 May 2009, Naha, Okinawa, Japan, pp. 103-108.

[49] U. Kalay, D. Hall, and M. Perkowski, “A minimal and universal test set for multiple-valued Galois field sum-of-products circuits,” Proceedings of 7th International Workshop on Post-Binary ULSI Systems (ULSIWS 19980, Fukuoka, Japan, May 1998, pp. 50-51.

[50] K.-H. Han and J.-H. Kim, “Quantum-inspired evolutionary algorithm for a class of combinatorial optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, Dec. 2002, pp. 580-593.

[51] K.-H. Han and J.-H. Kim, “Quantum-inspired evolutionary algorithms with a new termination criterion, H_ϵ gate, and two-phase scheme,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, April 2004, pp. 156-169.

[52] K.-H. Kim, J.-Y.Hwang, K.-H. Han, J.-H. Kim, and K.-H. Park, “A quantum-inspired evolutionary computing algorithm for disk allocation method,” *IEICE Transactions on Information and Systems*, vol. E86-D, no. 3, March 2003. pp. 645-649.

[53] Y. Kim, J.-H. Kim, and K.H. Han, “Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 Knapsack problems,” *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, July 2006, pp. 9151-9156.

[54] J.-S. Jang, K.-H. Han and J.-H. Kim, “Face detection using quantum-inspired evolutionary algorithm,” *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, June 2004, pp. 2100-2106.

[55] M.H.A. Khan and S. Akter, “Multiple-case outlier detection in least-squares regression model using quantum-inspired evolutionary algorithm,” *12th International Conference on Computer and Information Technology (ICCIT 2009)*, Dhaka, Bangladesh, 21-23 Dec. 2009.