A New Approach to Online Testing of TGFSOP-based Ternary Toffoli Circuits

N. M. Nayeem and J. E. Rice
Dept. of Math & Computer Science
University of Lethbridge
Lethbridge, Canada
Email: {*noor.nayeem, j.rice*}*@uleth.ca*

***Abstract*—Previous work presented a simple technique for converting an ESOP-based Boolean reversible circuit into an online testable circuit. This paper builds on the previous work by extending its application to circuits consisting of cascades of ternary Toffoli gates. The technique is applied to an existing cascade of ternary Toffoli gates, and requires that a single additional line be added to facilitate the propagation of any faults that are identified, as well as modification and/or addition of some gates. The modified circuit can detect online any single-bit errors that occur within the circuit. Experimental results compare very favourably to the only other known approach.**

I. Introduction

Reversible logic is becoming a popular area of research as power dissipation in the form of heat becomes problematic in today's computer chips. The International Technology Roadmap for Semiconductors (ITRS) [1] has indicated that this may become an insurmountable problem if we rely solely on existing technologies and approaches. Reversible logic, however, may provide a solution. Frank [2] states that computers based mainly on reversible logic operations could reuse up to 100% of the signal energies, and in 1973 Bennett showed that for power not to be dissipated it is necessary that a circuit be constructed from reversible gates [3]. Research on reversible computing has been shown to have uses in areas such as quantum computing [4], low power CMOS design [5], optical computing [6], and nanotechnology [7]. Many of these areas rely on a multiple-valued model, thus our work in the ternary case can be viewed as a stepping stone towards testable designs in these areas.

While some work such as [8], [9], [10] has been carried out in the area of reversible logic testing, there is little in the literature on reversible logic with online testability. The work we are aware of in this area includes [11], [12] and [13]. There is even less work on online testability of ternary reversible circuits, with the only work we are aware of being [14] (based on [15]). The motivation for ternary reversible circuits lies in the relationship between reversible logic and quantum computing.

Our paper builds on the work in [16] and [17] and extends the approach from Boolean (binary) to ternary reversible logic. Since the results in the Boolean case are so promising, we wish to determine if similar results can be obtained in the ternary case. Compared to work in [14] and [15], our approach leads to online testable circuits with significantly fewer costs, which is the goal of this work. The work in [16] modified a circuit generated by the synthesis approach in [18]. [16] required the addition of a parity line and a small number of gates to ensure that any single-bit error would be propagated along the parity line to the end of the circuit for detection. The approach we present here uses the same underlying idea, with modifications as necessary for use with ternary reversible gates.

II. Background

A. Ternary Galois Field Logic

Ternary Galois field (TGF) consists of {0, 1, 2} and two operations, addition modulo 3 and multiplication modulo 3. In this paper, the addition modulo 3 operation is denoted by $\oplus$ and the multiplication modulo 3 operation is denoted by the absence of any operator. Then for a ternary variable Y we have $Y = Y \oplus 3$ and $YYY = Y$.

According to [19], a ternary variable Y has six basic literals: Y, $Y^{+1} = Y \oplus 1$, $Y^{+2} = Y \oplus 2$, $Y^{12} = 2Y$, $Y^{01} = 2Y \oplus 1$, and $Y^{02} = 2Y \oplus 2$. A composite literal consists of the modulo 3 multiplication of two basic literals, for example, YY and $Y^{+2}Y^{12}$. In place of $\overline{Y}$ as used in Boolean logic we use $Y \oplus 1$ or $Y \oplus 2$ in this paper.

A product term is formed by applying the multiplication modulo 3 operations to constants, basic literals, and composite literals of ternary variables [19]. For example, given two variables X and Y, a product term can be $2X^{01}YY^{+1}$. A ternary Galois Field sum of products (TGFSOP) is an expression in which the addition modulo 3 operations are applied to product terms [19]. For example given two product terms, $2X^{01}YY^{+1}$ and XY^{12}, a TGFSOP expression can be written as $2X^{01}YY^{+1} \oplus XY^{12}$.

B. Reversible Logic

[20] provides the following definitions:

Definition 1: A gate is reversible if the (Boolean) function it computes is bijective.

Definition 2: A well-formed reversible logic circuit is an acyclic combinational logic circuit in which all gates are reversible, and are interconnected without fanout. This definition assumes that the circuit is strictly combinational;

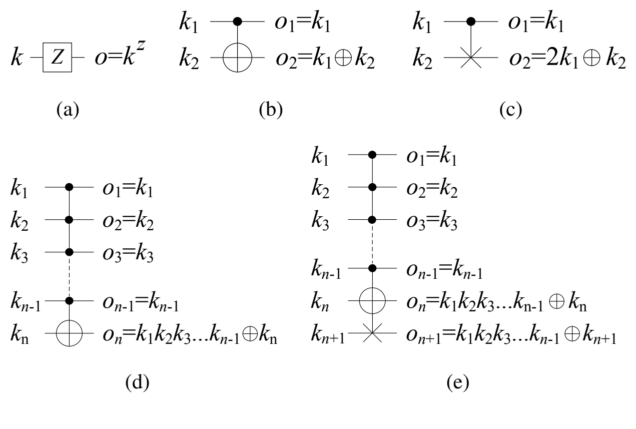
considerations for sequential logic are addressed in works such as [21].

Commonly used Boolean reversible gates include the Toffoli gate. An n-bit Toffoli gate maps the input vector $[k_1, k_2, \ldots, k_n]$ to the output vector $[o_1, o_2, \ldots, o_n]$ where $o_j = k_j$ for $j = 1, 2, \ldots, n-1$ and $o_n = k_1 \cdot k_2 \cdot \ldots \cdot k_{n-1} \oplus k_n$. The first $n-1$ bits are known as controls while the last bit is the target which is toggled only if all of the control lines are 1. Another name for a 2-bit (that is, $n = 2$) Toffoli gate is the CNOT gate. In this work we concentrate on ternary reversible gates. The reader is directed to, for example, [20] for descriptions of other Boolean reversible gates.

C. Reversible Ternary Gates

There are several ternary reversible gates. We define here the gates which are required for this paper. A 1-qutrit permutative gate [22] is defined as $\{k\} \rightarrow \{o = k^Z\}$ where $Z \in$ {+1, +2, 12, 01, 02} as shown in Figure 1(a). For example, if Z = +1, then $o = k^{+1} = k \oplus 1$.

An n-qutrit Toffoli gate is defined as mapping the input vector $[k_1, k_2, \ldots, k_{n-1}, k_n]$ to the output vector $[o_1, o_2, \ldots, o_{n-1}, o_n]$ where $o_j = k_j$ for $j = 1, 2, \ldots, n-1$, $o_n = f \oplus k_n$, and $f = k_1 k_2 \cdots k_{n-1}$. As in Boolean reversible logic the first $n-1$ inputs are known as controlling inputs and the last input is known as the controlled input. When $n = 2$ this gate becomes a 2-qutrit Toffoli gate with the input vector $[k_1, k_2]$ and output vector $[o_1 = k_1,\ o_2 = k_1 \oplus k_2]$. This gate is also known as the Feynman gate. A 2-qutrit modified Toffoli gate discussed in [15] is very similar to the 2-qutrit Toffoli gate, with the exception that $o_2 = 2k_1 \oplus k_2$. A 2-qutrit Toffoli gate, its modified version, and an n-qutrit Toffoli gate are shown in Figure 1(b)-(d).



(a)
(b)
(c)
(d)
(e)
k Z $o=k^z$
k_1 $o_1=k_1$
k_2 $o_2=k_1 \oplus k_2$
$o_2=2k_1 \oplus k_2$
$o_n=k_1k_2k_3...k_{n-1} \oplus k_n$
$o_{n+1}=k_1k_2k_3...k_{n-1} \oplus k_{n+1}$

Figure 1. (a) A 1-qutrit permutative gate, (b) a 2-qutrit Toffoli gate, (c) a 2-qutrit modified Toffoli gate, (d) an n-qutrit Toffoli gate, and (e) an $(n+1)$-qutrit TGC-2.

Khan, Perkowski and Kerntopf [23] proposed a Toffoli gate with m controlled inputs (TGC-m). If m = 1 this gate is the Toffoli gate as earlier defined. An $(n+1)$-qutrit TGC-2 is shown in Figure 1(e). In this figure we see that the gate maps the input vector $[k_1,\ k_2,\ \ldots,\ k_{n-1},\ k_n,\ k_{n+1}]$ to the output vector $[o_1,\ o_2,\ \ldots,\ o_{n-1},\ o_n,\ o_{n+1}]$ where $o_j = k_j$ for $j = 1, 2, \ldots, n-1$, $o_n = f \oplus k_n$, $o_{n+1} = f \oplus k_{n+1}$, and $f = k_1 k_2 \cdots k_{n-1}$. Here, the first $n-2$ inputs are the controlling inputs and the last two inputs are the controlled inputs.

D. Synthesis of Ternary Reversible Circuit

There are many approaches for synthesis of ternary reversible circuits including [23], [24], [25], [26], [27]. We briefly discuss the TGFSOP-based synthesis of ternary Toffoli circuits proposed in [23].

Consider a TGFSOP form of a function with u input variables and v output variables. An empty cascade with $2u + 1$ input lines and v output lines is created. The basic literals of an input variable can be realized using 1-qutrit permutative gates along the corresponding input line. However, for realizing the composite literal of a single input variable, two copies of that variable are required which can be managed by using a 2-qutrit Toffoli gate. Thus for each input variable, two input lines are needed. One constant input line is required in the cascade to realize the product term of the form $2X^{+1}X^{+1}$, where X is an input variable. Thus a circuit requires at most $2u + 1$ input lines and exactly v output lines. Now for each product term of each output variable in the TGFSOP function, a Toffoli gate is added to the circuit with controlling inputs connected to the input lines and controlled input connected to the output line. Unused input lines are removed from the circuit. Appropriate 1-qutrit permutative gates and 2-qutrit Toffoli gates are added on the input lines to restore the initial values. This approach ensures that the initial value and the final value of each input line are the same. For example, given a 3-input (I_1, I_2, I_3), 2-output (I_4, I_5) function in the TGFSOP form, $I_4 = I_1 I_2 \oplus I_1^{+1} I_3^{12}$ and $I_5 = 1 \oplus I_1 I_2^{01}$, a Toffoli cascade generated by this approach is shown in Figure 2. In the circuit, input lines are I_1, I_2, I_3, and output lines are I_4 and I_5. Toffoli gates are labeled as t_1 through t_3. At the end of circuit, two 1-qutrit permutative gates are added on lines I_2 and I_3 to restore the initial values.

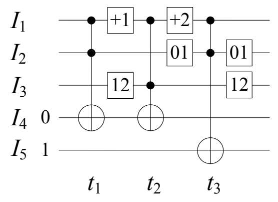


Figure 2. A ternary Toffoli circuit.

The synthesis approach that we have described here is slightly different from the original approach proposed in [23] since we have considered Toffoli gates whereas the original approach uses the TGC-m.

E. *Cost Metrics*

It is customary to use some cost metric for comparison among differing synthesis approaches. In order for accurate comparison it is necessary to clearly define how these costs are measured. The cost of a ternary circuit is the total cost of the gates used to realize the circuit. According to [22], the cost of a 1-qutrit permutative gate and a 2-qutrit Toffoli gate are 1 and 4, respectively. The cost of the modified 2-qutrit Toffoli gate is the same as that of a 2-qutrit Toffoli gate. [28] presents the lowest cost implementation of the n-qutrit ternary Toffoli gates with $n > 2$. A 3-qutrit Toffoli gate and a 4-qutrit Toffoli gate have costs of 16 and 40. For an n-qutrit ($n > 4$) Toffoli gate, the cost is defined as eight plus twice the cost of an ($n-1$)-qutrit Toffoli gate. We have calculated the cost of an ($n+1$)-qutrit ($n > 3$) TGC-2, which is six plus the cost of an n-qutrit Toffoli gate. A 3-qutrit TGC-2 and a 4-qutrit TGC-2 have costs of 6 and 20, respectively. These costs are summarized in Table I.

Table I
COSTS OF TERNARY GATES.

Gate	Cost
1-qutrit permutative gate	1
2-qutrit Toffoli gate	4
2-qutrit modified Toffoli gate	4
3-qutrit Toffoli gate	16
4-qutrit Toffoli gate	40
n-qutrit ($n > 4$) Toffoli gate	8 + 2*(cost of an ($n-1$) -qutrit Toffoli gate)
3-qutrit TGC-2	6
4-qutrit TGC-2	20
($n+1$)-qutrit ($n > 3$) TGC-2	6 + cost of an n-qutrit Toffoli gate

F. *Testing and Fault Models*

We briefly explain the difference between *offline* and *online* testing. In offline testing a test vector consisting of inputs identified to be useful in detecting errors is applied to the circuit. This requires that the circuit be taken out of operation for some time, and that the outputs resulting from the tests be compared with a set of known correct outputs. In contrast, online testing is carried out while the circuit is being used for normal operations, and additional circuitry is used to identify if a fault has occurred. It is the latter approach that we focus on in this work.

There are a variety of fault models of use with reversible logic, including the missing gate fault model, repeated gate fault model, and reduced gate fault model [29] as well as the crosspoint fault model [30]. Other works such as [11] have suggested the use of a bit fault model. In this model, a fault in a gate changes the behavior of its outputs. A single bit fault is reflected on exactly one output of a gate, changing the correct value of the output to a faulty value.

Our work uses the single bit fault model, although we note that the use of the term "bit" is not entirely accurate for ternary logic. However the concept behind the original model is still appropriate for this work, in that we are identifying the situation when a fault is reflected on exactly one output of a gate. While we acknowledge the inaccuracy of the term we refer to the model as a single-bit model through-out this paper primarily for the sake of consistency with other work.

III. OUR APPROACH

Given a TGFSOP representation of a ternary function, a ternary Toffoli circuit is generated as discussed in Section II-D. The resulting circuit will consist of only 1-qutrit permutative gates and n-qutrit ($n > 1$) Toffoli gates. Let the number of input lines be p and the number of output lines be q in the Toffoli circuit. If the initial and final values of any input line are not the same, then appropriate gates (*e.g.* 1-qutrit permutative gates and 2-qutrit Toffoli gates) are added to restore the initial value at the end of the corresponding input line. An approach to convert such a circuit into an online testable circuit is proposed below.

The proposed approach requires a parity line L which is initialized with a zero. For each input and output line in the given Toffoli circuit, this approach inserts a 2-qutrit Toffoli gate and a 2-qutrit modified Toffoli gate at the beginning and at the end of the circuit, respectively. Note that that for each such gate, the controlled input is connected to the line L. This step requires a total of $2p+2q$ gates. All 1-qutrit permutative gates found in the given circuit are retained. Then each n-qutrit ($n > 1$) Toffoli gate is replaced by an ($n+1$)-qutrit ($n > 1$) TGC-2. The connections of the first n qutrits of the ($n+1$)-qutrit ($n > 1$) TGC-2 remain the same as that of n-qutrit Toffoli gate. The second controlled input (*i.e.* the last qutrit) is connected to L. The converted circuit is now online testable. If a single fault occurs on any input line, output line, or line L, then the value of L changes to 1 or 2. If no fault occurs, L remains 0. The following example describes this approach.

For a given 3-input (I_1, I_2, I_3), 2-output (I_4, I_5) Toffoli circuit shown in Figure 2, the proposed approach generates an online testable circuit as shown in Figure 3. In the testable circuit, a parity line L is added and the Toffoli gates (t_1, t_2, and t_3) are replaced by TGC-2s (e_1, e_2, and e_3). In addition five 2-qutrit Toffoli gates c_1 through c_5 and five 2-qutrit modified Toffoli gates c_6 through c_{10} are added.

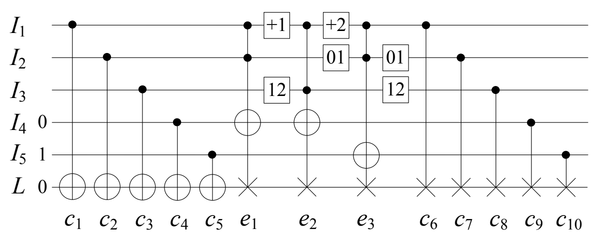

I_1
I_2
I_3
I_4 0
I_5 1
L 0
+1
12
+2
01
01
12
c_1 c_2 c_3 c_4 c_5 e_1 e_2 e_3 c_6 c_7 c_8 c_9 c_{10}

Figure 3. An online testable reversible circuit.

IV. Analysis

In our proposed testable design, a fault on an input line can propagate to the output lines and the parity line L by the TGC-2s. However, a fault on an output line or L cannot propagate to other lines since controlling inputs of the TGC-2s are not connected to the output lines or L. Lemma 1 proves that the last two outputs of the TGC-2 compute the same function no matter whether the faults occur in controlling inputs, controlled inputs or both. Lemma 2 proves that the proposed testable design can detect a single bit fault even though it propagates to multiple lines by TGC-2s, causing multiple faults.

Lemma 1. Consider an $(n+1)$-qutrit TGC-2, mapping the input vector $[k_1, k_2, \ldots, k_{n-1}, k_n, k_{n+1}]$ to the output vector $[o_1, o_2, \ldots, o_{n-1}, o_n, o_{n+1}]$, where $o_j = k_j$ (for j = 1, 2, …, $n-1$), $o_n = f \oplus k_n$, $o_{n+1} = f \oplus k_{n+1}$, and $f = k_1k_2\cdots k_{n-1}$.

a) If faults occur in controlling inputs and affect the gate, then both o_n and o_{n+1} compute $\overline{f}$; otherwise both outputs compute f.
b) If faults occur in controlled inputs of the gate, then both o_n and o_{n+1} compute f.
c) If faults occur in controlled inputs and controlling inputs which affect the gate, then both o_n and o_{n+1} compute $\overline{f}$; otherwise both outputs compute f. ◁

Proof.

a) Consider faults in $k_i, k_j, \ldots, k_l$ such that $\{i, j, \ldots, l\} \subseteq \{1, 2, \ldots, n-1\}$. These faults affect the gate only if such faults cause the function f to be changed to $\overline{f}$. As an example, consider a TGC-2 with $f = k_1k_2k_3$. Let, the values of k_1, k_2, and k_3 be 1, 1, and 2, respectively. Thus f = 2. If a fault changes the value of k_1 to 2, then the fault has an effect since f also changes from 2 to 1.
Faults affecting the gate have impact on o_n and o_{n+1} since both outputs compute f. Thus $o_n = \overline{f} \oplus k_n$ and $o_{n+1} = \overline{f} \oplus k_{n+1}$. If the faults do not affect the calculation of f, then according to the definition, $o_n = f \oplus k_n$ and $o_{n+1} = f \oplus k_{n+1}$.
Therefore, if the faults have an effect, then o_n and o_{n+1} compute $\overline{f}$; otherwise these outputs compute f.
b) We consider three cases, depending on whether one of the controlled inputs is faulty or both controlled inputs are faulty.
Case 1. Assume a fault occurs in k_n. This fault does not propagate to o_{n+1} since o_{n+1} is independent of k_n; thus $o_{n+1} = f \oplus k_{n+1}$. However, due to the fault, $o_n = f \oplus \overline{k}_n$.
Case 2. Assume a fault occurs in k_{n+1}. The proof is similar to Case 1. We get $o_n = f \oplus k_n$ and $o_{n+1} = f \oplus \overline{k}_{n+1}$.
Case 3. Consider that faults occur in both k_n and k_{n+1}. Due to these faults, $o_n = f \oplus \overline{k}_n$ and $o_{n+1} = f \oplus \overline{k}_{n+1}$.
Hence, for any of the cases, both o_n and o_{n+1} compute f.
c) We consider two cases, depending on whether faults in controlling inputs affect the gate or not.
Case 1. First consider that faults in controlling inputs affect the gate. From (a), we can write $o_n = \overline{f} \oplus k_n$ and $o_{n+1} = \overline{f} \oplus k_{n+1}$.
Assume that faults also occur in both controlled inputs. According to (b), we can rewrite the outputs as follows: $o_n = \overline{f} \oplus \overline{k}_n$ and $o_{n+1} = \overline{f} \oplus \overline{k}_{n+1}$. Thus both outputs compute $\overline{f}$. Similarly, we can reach the same conclusion if a fault occurs in any of controlled inputs.
Case 2. For the case that faults in controlling inputs have no effect on the gate, the proof is similar to (b). ◀

Lemma 2. If any single fault occurs on any line in the testable circuit, the value of L changes from 0 to either 1 or 2, and the fault is detected. ◁

Proof. Consider an online testable ternary circuit consisting of p input lines, q output lines, and a parity line L. Let $G = \{g_1, g_2, \ldots, g_N\}$ be the set of $(n+1)$-qutrit ($n > 1$) TGC-2s used in the circuit. Let the initial values of the input lines Given an $(n+1)$-qutrit TGC-2 $g_i \in G$ and the input vector $[k_1, k_2, k_3, \ldots, k_{n-1}, k_n, k_{n+1}]$, the output vector is $[o_1 = k_1, o_2 = k_2, o_3 = k_3, \ldots, o_{n-1} = k_{n-1}, o_n = fg_i \oplus k_n, o_{n+1} = fg_i \oplus k_{n+1}]$, where $fg_i = k_1k_2\cdots k_{n-1}$, and n can be at most $p+1$. In order to prove this lemma, we consider the following three cases:

Case 1. Consider a single fault on an input line X_w (for w = 1, 2, …, p) which affects a number of gates. Let $A = \{a_1, a_2, \ldots, a_u\} \subseteq G$ be the set of gates which are affected by the fault. Let $B = \{b_1, b_2, \ldots, b_v\} = G - A$. Here, A or B can be empty.

Note that the last two outputs of a gate $g_i \in G$ compute the same function fg_i. A gate b_r in B computes fb_r, for r = 1, 2, …, v. However, from Lemma 1, due to the fault, each gate a in A computes $\overline{fa_s}$, for s = 1, 2, …, u.

At the beginning of circuit, there is a 2-qutrit Toffoli gate between X_y and L, for $\forall\ y \in \{1, 2, \ldots, p+q\}$. Thus the value of L, just before the first gate in G, is $x_1 \oplus x_2 \oplus \ldots \oplus x_w \oplus \ldots \oplus x_{p+q}$. After the last gate in G, the value of L becomes $x_1 \oplus x_2 \oplus \ldots \oplus x_w \oplus \ldots \oplus x_{p+q} \oplus fb_1 \oplus fb_2 \oplus \ldots \oplus fb_v \oplus \overline{fa_1} \oplus \overline{fa_2} \oplus \ldots \oplus \overline{fa_u}$.

At the end, there is a 2-qutrit modified Toffoli gate between X_y and L, for $\forall\ y \in \{1, 2, \ldots, p+q\}$. As a result, the faulty value of line X_w (which is $\overline{x}_w$) propagates to L and hence L becomes $x_1 \oplus x_2 \oplus \ldots \oplus x_w \oplus \ldots \oplus x_{p+q} \oplus fb_1 \oplus fb_2 \oplus \ldots \oplus fb_v \oplus \overline{fa_1} \oplus \overline{fa_2} \oplus \ldots \oplus \overline{fa_u} \oplus 2fb_1 \oplus 2fb_2 \oplus \ldots \oplus 2fb_v \oplus 2\overline{fa_1} \oplus 2\overline{fa_2} \oplus \ldots \oplus 2\overline{fa_u} \oplus 2x_1 \oplus 2x_2 \oplus \ldots \oplus 2\overline{x}_w \oplus \ldots \oplus 2x_{p+q}$

$= x_w \oplus 2\overline{x}_w$

$= x_w \oplus 2x_w \oplus 1$ or $x_w \oplus 2x_w \oplus 2$ = 1 or 2.

Since at the end the line L contains 1 or 2, the circuit can detect the fault.

Case 2. Now consider that a single fault occurs on an output line X_{p+w} at any point, where w = 1, 2, . . ., q.

Consider a set of gates, $A = \{a_1, a_2, \ldots, a_u\} \subseteq G$ which have controlled inputs on line X_{p+w}. Consider another set of gates, $B = \{b_1, b_2, \ldots, b_v\}$ such that $B = G - A$. According to Lemma 1, each gate a_s in A computes fa_s, for s = 1, 2, . . ., u. Similarly, each gate b_r in B computes fb_r, for r = 1, 2, . . ., v.

The fault on X_{p+w} propagates to the end of the line. The value of the line X_{p+w} at the end is $x_{p+w} \oplus fa_1 \oplus fa_2 \oplus \ldots \oplus fa_u \oplus m$ where m is either 1 or 2. Because of the 2-qutrit modified Toffoli gates between X_y and L, $\forall\, y \in \{1, 2, \ldots, p{+}q\}$, the faulty value of X_{p+w} appears at L. Thus L becomes $x_1 \oplus x_2 \oplus \ldots \oplus x_{p+q} \oplus fa_1 \oplus fa_2 \oplus \ldots \oplus fa_u \oplus fb_1 \oplus fb_2 \oplus \ldots \oplus fb_v \oplus 2x_1 \oplus 2x_2 \oplus \ldots \oplus 2x_{p+q} \oplus 2fa_1 \oplus 2fa_2 \oplus \ldots \oplus 2fa_u \oplus 2fb_1 \oplus 2fb_2 \oplus \ldots \oplus 2fb_v \oplus 2m$
= $2m$ = 1 or 2.

Case 3. A fault can also occur on L. This causes L to have the value either 1 or 2.

Hence, for any of these cases, the circuit detects the fault. ◀

The following two examples illustrate the detection of single faults which occur on an input line and an output line, respectively.

Example 1: For a given online testable circuit with three input lines (I_1, I_2, I_3) and two output lines (I_4, I_5) as shown in Figure 4, consider a single fault on the input line I_2 just before the first TGC-2. For the faulty lines, output values are given in the form [fault-free value/ faulty value] after each gate. For other lines, only the fault free values are shown. Note that the fault on I_2 propagates to lines I_4 and I_5, causing multiple faults. However, the fault is detectable since the value of L changes to 2. This example illustrates that the circuit is able to detect a fault even though it propagates to several lines.

Example 2: For a given online testable circuit with three input lines (I_1, I_2, I_3) and two output lines (I_4, I_5) as shown in Figure 5, consider a single fault on the output line I_4 between the first and second TGC-2s. Note that the fault propagates to the end of I_4 and L becomes 1. Thus the fault is detected.

V. Experimental Results

For a number of benchmark functions, we have implemented the testable circuits using our proposed approach and the previous approach [15]. In order to calculate the overhead for adding the testability feature, we have also implemented the non-testable circuits as described in Section II-D. The results are summarized in Table II. The first and second columns specify the function name and the total number of input and variables in the function. The third and fourth columns show the costs of testable circuits generated by the previous approach [15] and our approach, respectively. The fifth column calculates the improvements achieved by our approach. The next column shows the costs of non-testable circuits. The last two columns analyze the overhead costs of the existing testable circuits and our testable circuits over the non-testable circuits.

It can be seen from Table II that our approach reduces the cost considerably for every circuit, compared to the previous approach. As an example, our approach requires a cost of 332 for implementing the function 6CyG3, whereas the previous approach requires 1843 cost. Improvements of our approach range from 32% to 82%, with 6 out of 11 circuits improved by more than 70%.

We have found that the overhead costs incurred by the previous approach are much higher than the costs incurred by our approach. However we can see from the last column of Table II that our costs are significantly higher for the first seven benchmarks than for the last four. Since our approach requires the addition of $2(p + q)$ gates regardless of the number of gates in the reversible circuit, this will result in a higher overhead cost for smaller circuits *i.e.* those similar to the first seven benchmarks used in our experiments. We note that the non-testable circuits of such benchmarks consist of no more than three gates, with the exception being the circuit for a2bccG which requires five gates. The remaining four non-testable circuits in Table II contain up to ten gates; thus the overhead of our approach is reduced to only 17%. As an example, for a randomly generated circuit with $p = 3$, $q = 2$ and only 3 gates in the cascade, our overhead can be expressed as 48% additional costs. However, a circuit with the same values of p and q but requiring 70 gates in the cascade has an overhead cost of only 16%. Thus our approach has the potential to minimize the overhead costs for larger circuits. Continuing work will involve the development and/or acquisition of larger benchmarks on which to test our approach. As well, further work will be carried out to estimate the overall fault coverage in terms of detectable faults.

Table II
Experimental results.

Function	No. of I/O variables	Testable circuit		Improvement	Non-testable circuit	Overhead	
		Previous approach	Our approach			Previous approach	Our approach
2CyG2	3	281	78	72%	40	603%	95%
3CyG2	4	489	92	81%	48	919%	92%
a2bccG	4	364	112	69%	56	550%	100%
ProdG2	3	65	44	32%	16	306%	175%
ProdG3	4	148	78	47%	40	270%	95%
SumG2	3	65	36	45%	8	713%	350%
SumG3	4	148	50	66%	12	1133%	317%
5CyG3	6	1536	278	82%	200	668%	39%
6CyG3	7	1843	332	82%	240	668%	38%
7CyG4	8	2923	722	75%	616	375%	17%
10CyG4	11	4219	1028	76%	880	379%	17%

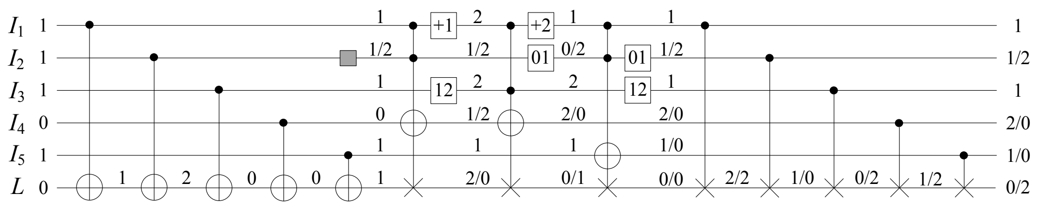


Figure 4. A single fault on input line I_2.

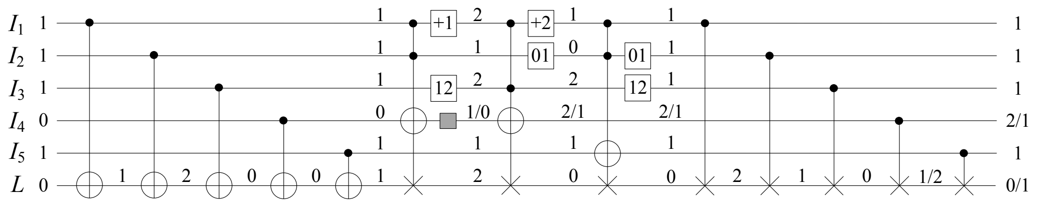


Figure 5. A single fault on output line I_4.

VI. Conclusion

This paper presents an approach for taking an existing TGFSOP-based cascade of ternary Toffoli gates and adding a single parity line plus a fixed number of gates to modify the circuit in order to provide online testability. With these small modifications we create a circuit that computes its original functionality and in addition will detect and propagate to the outputs any single-bit fault that occurs within the circuit. The process is straightforward and provable, and requires only $2(p+q)$ additional gates.

Experimental results showed that our approach far outperformed the only other approach available for comparison, and that for most of the benchmarks tested the overhead incurred over a non-testable approach was 100% or less. We highlight that the majority of the benchmarks currently available to us require a very small number of gates for their implementation, and that as the benchmark sizes grow our overhead for online testability will shrink.

Our work differs from any other in either Boolean or ternary reversible online testing techniques, in that other techniques require redesign of the entire circuit. Also, in most other techniques only small portions of the circuits can be tested, requiring a rail checker or similar structure to combine and propagate online testing results. Most importantly we add no garbage lines to the design. However, this approach currently works only for a particular type of TGFSOP-based synthesis. Future work will extend this approach for other type of synthesis methods as well as investigate online detection of multiple errors.

Acknowledgment

This research was funded by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] "International Technology Roadmap for Semiconductors (ITRS)," http://public.itrs.net, 2009 executive summary.

[2] M. P. Frank, "Introduction to reversible computing: Motivation, progress, and challenges," in *Proceedings of the 2nd Conference on Computing Frontiers*, Ischia, Italy, May 4–6 2005, pp. 385–390.

[3] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[4] A. N. Al-Rabadi, "New classes of Kronecker-based reversible decision trees and their group-theoretic representations," in *Proceedings of the International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Vienna, Austria, September 11-12 2004, pp. 233–243.

[5] W. C. Athas and L. J. Svensson, "Reversible logic issues in adiabatic CMOS," in *Proceedings of Workshop on Physics and Computation (PhysComp)*, Dallas, TX, 1994, pp. 111–118.

[6] P. Picton, "Optoelectronic, multivalued, conservative logic," *International Journal of Optical Computing*, vol. 2, pp. 19–29, 1991.

[7] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, no. 1, pp. 21–40, 1993.

[8] K. N. Patel, J. P. Hayes, and I. L. Markov, "Fault testing for reversible circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 8, pp. 1220–1230, 2004.

[9] M. Ibrahim, A. R. Chowdhury, and H. M. H. Babu, "Minimization of CTS of k-CNOT circuits for SSF and MSF model," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Boston, MA, 2008, pp. 290–298.

[10] H. Rahaman, D. K. Kole, D. K. Das, and B. B. Bhattacharya, "On the detection of missing-gate faults in reversible circuits by a universal test set," in *Proceedings of the 21st International Conference on VLSI Design*, 2008, pp. 163–168. [Online]. Available: http://dx.doi.org/10.1109/VLSI.2008.106

[11] D. P. Vasudevan, P. K. Lala, D. Jia, and J. P. Parkerson, "Reversible logic design with online testability," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 2, pp. 406–414, 2006.

[12] S. N. Mahammad and K. Veezhinathan, "Constructing online testable circuits using reversible logic," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 1, pp. 101–109, 2010.

[13] N. Farazmand, M. Zamani, and M. B. Tahoori, "Online fault testing of reversible logic using dual rail coding," in *Proceedings of 16th IEEE International On-Line Testing Symposium (IOLTS)*, 5-7 July 2010, pp. 204–205.

[14] M. R. Rahman and J. E. Rice, "On designing a ternary reversible circuit for online testability," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, B.C., Canada, August 2011, pp. 119–124.

[15] M. R. Rahman, "Online testing in ternary reversible logic," Master's thesis, University of Lethbridge, 2011.

[16] N. M. Nayeem and J. E. Rice, "A simple approach for designing online testable reversible circuits," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, Canada, August 2011, pp. 85–90.

[17] ——, "Online fault detection in reversible logic," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, Vancouver, Canada, October 2011.

[18] K. Fazel, M. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 22-24 Aug. 2007, pp. 206–209.

[19] M. H. A. Khan, M. Perkowski, M. R. Khan, and P. Kerntopf, "Terary GFSOP minimization using kronecker decision diagrams and their synthesis with quantum cascades," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 11, no. 5-6, pp. 567–602, 2005.

[20] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[21] J. E. Rice, "An introduction to reversible latches," *The Computer Journal*, vol. 51, no. 6, pp. 700–709, 2007.

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

[23] M. H. A. Khan, M. Perkowski, and P. Kerntopf, "Multi-output Galois field sum of products synthesis with new quantum cascades," in *Proceedings of the 33rd International Symposium on Multiple-Valued Logic*, 2003, pp. 146–153.

[24] N. Denler, B. Yen, M. Perkowski, and P. Kerntopf, "Synthesis of reversible circuits from a subset of Muthukrishnan-Stroud quantum realizable multi-valued gates," in *Proceedings of the International Workshop on Logic & Synthesis (IWLS)*, 2004.

[25] M. H. A. Khan and M. Perkowski, "Genetic algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates," in *Proceedings of Congress on Evolutionary Computation (CEC)*, vol. 2, 2004, pp. 2194–2201.

[26] D. M. Miller, D. Maslov, and G. W. Dueck, "Synthesis of quantum multiple-valued circuits," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 12, no. 5-6, pp. 431–450, 2006.

[27] M. H. A. Khan, "GFSOP-based ternary quantum logic synthesis," in *Proceedings of SPIE 7797 (Optics and Photonics for Information Processing IV)*, San Diego, California, 2010.

[28] 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," in *Proceedings of the 39th International Symposium on Multiple-Valued Logic (ISMVL)*, 2009, pp. 98–102.

[29] J. P. Hayes, I. Polian, and B. Becker, "Testing for missing-gate faults in reversible circuits," in *Proceedings of the 13th Asian Test Symposium*, 2004, pp. 100–105. [Online]. Available: http://dx.doi.org/10.1109/ATS.2004.84

[30] J. Zhong and J. C. Muzio, "Analyzing fault models for reversible logic circuits," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, BC, 2006, pp. 2422–2427.