A Modular Approach to Designing an Online Testable Ternary Reversible Circuit

J. E. Rice *[1], R. Rahman [2]

University of Lethbridge, Lethbridge, AB, Canada
*[1]j.rice@uleth.ca; [2]md.rahman7@uleth.ca

Abstract

Energy inefficiency in irreversible logic circuits is creating an obstruction on the path towards continued advancements in complexity and reductions in the size of today's computer systems. Designing the component circuits in a reversible manner may offer a possible solution to this crisis, allowing significant reductions in power consumption and heat dissipation requirements. Multi-valued (MV) reversible logic can provide further advantages over binary reversible logic, such as better performance or reducing wiring congestion. The current literature, however, contains very little work on testability of such designs. This paper describes the design of an online testable block for ternary reversible logic. This block implements most ternary logic operations and provides online testability for a reversible ternary network composed of several of these blocks. The testable block is composed of reversible building blocks, and thus is itself reversible. Multiple such blocks can be combined to construct complex and complete, testable, ternary reversible circuits.

Keywords

Reversible Logic; Multiple-valued Logic (MVL); Online (Concurrent) Testability; Design for Testability

Introduction

Because of the irreversible nature of today's circuits information is lost as processing takes place, and as information is lost energy is lost in the form of heat dissipation. The use of reversible transformations can preserve energy by conserving information (Frank 2005), and in fact as early as 1961 researchers had proven that reversible computing could offer a solution to the problem of information loss and heat dissipation (Landauer 1961; Bennett 1973).
Testing and reliability of computer systems is very important; however the area of fault detection in reversible circuits is fairly new. Recent works such as (Kole, Rahman, Das, and Bhattacharya 2010; Polian, Fiehn, Becker, and Hayes 2005) and (Zhong and Muzio 2006) have focused on this area. The principle motivation behind work on ternary online testing lies in quantum technologies. Quantum technologies can be both binary and ternary, however ternary logic provides many advantages over binary such as high computational speed (Miller and Thornton 2008). Quantum logic operations are also reversible (Khan and Perkowski 2007), hence research on online reversible testability may assist the development of test methods for quantum circuits.

In this paper we introduce an online testable block for use in the construction of ternary reversible circuits. We discuss the limitations of previously proposed designs (Rahman and Rice 2011b; Rahman and Rice 2011a), and introduce an upgrade. We also propose additional approaches to reduce the quantum cost and provide comparisons.

Background

We begin with some background information in order to aid the reader in their understanding of this work.

Ternary Operations

A multi-valued logic (MVL) is a logic system which utilises variables that can take on a discrete and finite set of values (Miller and Thornton 2008). For example, the binary logic system deals with values 0 and 1, whereas a three-valued, or ternary system is a MVL that deals with values 0, 1 and 2. A finite set of values including two identity elements and a set of operators are the basic elements of the algebra for a MVL. These identity elements and operators are well defined over the finite set in one-place and two-place functions. We use the term one-place to refer to the functions that have one operand and the term two-place to refer functions that have two operands. The finite sets of logic values generally have cardinality $p \geq 3$ in a MVL system.

The commonly used operators in a ternary system are given in Table 1. We note that the $\cdot$ symbol for the product operator is often omitted, as in regular multiplication. This practice is followed later on in *e.g.* Figure 6(a).

TABLE 1. SOME COMMON TERNARY OPERATORS.

operator	behaviour			
mod-sum	$\oplus$	0	1	2
$x \oplus y = (x+y)\ mod\ p$	0	0	1	2
	1	1	2	0
	2	2	0	1
mod-difference	$\ominus$	0	1	2
$x \ominus y = (x-y)\ mod\ p$	0	0	2	1
	1	1	0	2
	2	2	1	0
product	$\cdot$	0	1	2
$x \cdot y = (x \cdot y)\ mod\ p$	0	0	0	0
	1	0	1	2
	2	0	2	1
successor	x	$\overrightarrow{x}$		
$\overrightarrow{x} = (x+1)\ mod\ p$	0	1		
	1	2		
	2	0		

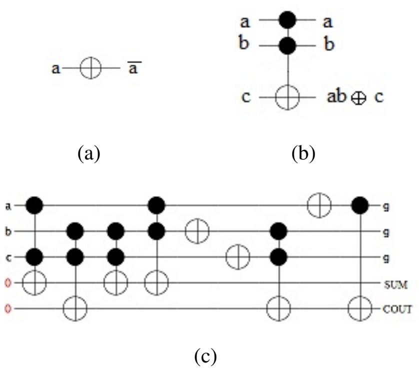


FIGURE 1. (A) THE NOT GATE, (B) THE TOFFOLI GATE, AND (C) A REVERSIBLE FULL ADDER.

Reversible Logic

The following definitions are fundamental to the area of reversible logic (Shende, Prasad, Markov, and Hayes 2003):

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.*

Table 2 defines the behaviour of some of the more commonly-used binary reversible gates.

TABLE 2. A SELECTION OF BINARY REVERSIBLE LOGIC GATES.

gate	behaviour
NOT	$(x) \rightarrow (x \oplus 1)$
Feynman	$(x,y) \rightarrow (x, x \oplus y)$
Toffoli	$(x,y,z) \rightarrow (x,y,xy \oplus z)$
Fredkin	$(x,y,z) \rightarrow (x,z,y)$ *iff* $x = 1$

Binary reversible gates are cascaded together in order to construct binary reversible circuits. For example Figure 1 illustrates a binary full adder design using NOT and Toffoli gates.
The presence of garbage outputs is sometimes necessary in a reversible circuit to maintain the reversibility. Garbage can be defined as as the number of unutilized outputs required to convert an irreversible function to a reversible one (Maslov and Dueck 2003). Some authors do not consider primary inputs (input variables) or their complemented forms to be garbage outputs (Chowdhury 2006; Khan and Perkowski 2003), and a similar assumption is made here.

Ternary Reversible Logic

We provide here descriptions for the ternary gates that are used in this work. A note on terminology is relevant here; the lines or variables operated on in MV reversible computing are often assumed to have some quantum implementation, and thus the commonly used term in binary quantum reversible logic for a variable is *qubit*, while the extension to ternary is *qutrit* and in general for MV logic the term used is *qudit*.

1) 1-qutrit Permutative Gates

Any transformation of the qutrit state can be represented by a 3×3 unitary matrix (Khan, Perkowski, and Khan 2004; Khan 2008). This transformation is known as a Z-transformation. A Z-transformation shifts or permutes the input states to generate the desired output states. For example, the Z(+1) transformation shifts the input states by 1. There are numerous Z-transformations that can be defined by varying the 3×3 matrices, but the most useful transformations are shown in Table 3. There are five ternary one-place operations corresponding to the permutation of ternary values 0,1 and 2 and each of these can be constructed as ternary reversible gates. These gates are known as ternary 1-qutrit permutative gates. Table 3 shows the operations of these 1-qutrit permutative gates as defined by Khan (2008). Two 1-qutrit ternary gates act as another 1-qutrit ternary gate if they are cascaded together (Khan 2008). Table 4 shows the resultant 1-qutrit gates for two cascaded 1-qutrit gates as defined by in (Khan 2008).

TABLE 3. OPERATIONS OF 1-QUTRIT PERMUTATIVE GATES.

Input	Output of z-transformation				
	Z(+1)	Z(+2)	Z(12)	Z(01)	Z(02)
0	1	2	0	1	2
1	2	0	2	0	1
2	0	1	1	2	0

TABLE 4. OPERATIONS RESULTING FROM CASCADING TWO 1-QUTRIT GATES.

First 1-qutrit gate	+1	+2	12	01	02
+1	+2	+0	02	12	01
+2	+0	+1	01	02	12
12	01	02	+0	+1	+2
01	02	12	+2	+0	+1
02	12	01	+1	+2	+0

2) M-S Gates

The Muthukrishnan-Stroud, or M-S gate is a 2-qutrit MV gate which can be realized using ion-trap technology (Khan and Perkowski 2007). The M-S gate is generally represented as shown in Figure 2(a). The value of output Q is controlled by the value of the input A. Q is the Z-transformation of the input B whenever $A = 2$, where Z$\in \{+1, +2, 12, 01, 10\}$. If $A \neq 2$ then input B is passed unchanged as $Q = B$. The Z transforms are described in Table 3 (Khan and Perkowski 2007).

3) Ternary Toffoli and Feynman Gates

Toffoli gates and 2×2 Feynman gates are used extensively in this work. The notations used in this paper are shown in Figure 2. Quantum realizations using M-S gates to implement ternary Toffoli and Feynman gates are discussed by Khan and Perkowski (2007) as well as in a later work (Khan 2009).

4) Generalized Toffoli Gates (GTGs)

In a generalized Toffoli gate (GTG), whenever the values of the two controlling inputs are 2 a Z-transformation is applied on the controlled input to generate the output. For all other combinations of the controlling inputs the controlled input is passed unchanged. However to make the operation more generalized, *i.e.* to allow the Z-transformation to be activated for other combinations of the controlling inputs (other than 2, 2), (Khan and Perkowski 2007) proposed a more general version of this Toffoli gate. The symbol for Khan's GTG is shown in Figure 2(d), where x and y indicate the controlling values for each line.

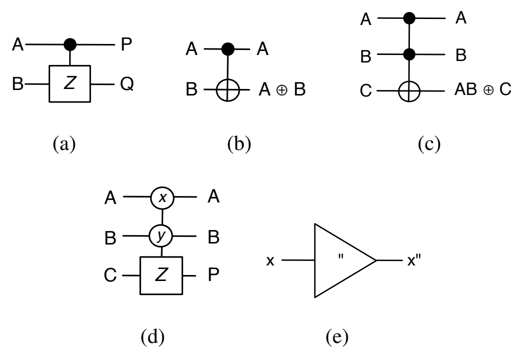


FIGURE 2. (A) THE M-S GATE, (B) THE 2-INPUT FEYNMAN GATE, (C) THE 3-INPUT TERNARY TOFFOLI GATE, (D) THE GENERALIZED TOFFOLI GATE (GTG), AND (E) THE DUAL SHIFT GATE.

5) Dual Shift Gates

Six 1-qutrit ternary shift gates based on GF3 operations were proposed in (Khan 2004). In this work we utilise one of these, the dual shift gate. The dual shift gate has the functionality $x \rightarrow (x+2) mod 3$ and is shown in Figure 2(e).

6) Cost Metrics

One simple way to compare reversible circuits is by counting the gates required in a circuit's implementation. However, gate count does not consider the complexity of the gates, and thus is often an inaccurate measure of the size of a reversible circuit. For example, if a circuit can be constructed either using five Toffoli gates with 2 control lines, or two Toffoli gates each with 10 control lines, the gate count metric will consider the second approach to be better since it uses only two gates to realize the circuit. However, this is inaccurate since a 10-bit Toffoli gate has higher complexity, and requires additional quantum operations when compared to a 2-bit Toffoli gate (Nayeem 2011).

(Maslov and Dueck 2004) define the quantum cost of a reversible gate as the number of quantum operations required to realize that gate. The M-S gate is considered to be an elementary quantum building block which Khan and Perkowski (2007) define as having a cost of 1, and this is used as the cost metric in this work.

Fault Models

Fault modeling refers to the detection and modeling of the behaviour of possible defects in a circuit. There are various traditional models of faults such as stuck-at faults and bridging faults (Jha and Gupta 2003), as well as models suggested specifically for reversible implementations such as missing gate faults (Polian, Fiehn, Becker, and Hayes 2005) and cross point faults (Zhong and Muzio 2006). The fault model used in this work is the single-bit fault used in (Vasudevan, Lala, Di, and Parkerson 2006). Whenever an internal circuit error changes the value of an output value, a single bit error occurs. The error is reflected in any one of the output values of the block. Single bit errors are very similar to stuck-at faults; however, stuck-at faults behave independently of the inputs whereas the behaviour of a single bit error is dependent on the initial input values.

The ternary online testing blocks we propose in this paper are designed to identify single bit errors propagated to one output line within the testable blocks. This uses a method similar to that proposed for the binary case by other authors (Vasudevan, Lala, Di, and Parkerson 2006). The errors are then propagated to the circuit outputs via the use of a ternary two-pair two-rail checker. Since we are working with ternary logic, a single bit error should technically be referred to as a single qutrit error; however for the sake of simplicity we use the term “single bit error” through-out the rest of this paper.

Online Testability

Online, or concurrent testability refers to the ability of a circuit to test a portion of itself without halting the operation (Vasudevan, Lala, Di, and Parkerson 2006). Online testing includes detecting a fault, the point of occurrence and sometimes, fault recovery (Jha and Gupta 2003).

Related Work

Previous work includes the proposal of binary reversible online teatble logic blocks (Vasudevan, Lala, Di, and Parkerson 2006). The proposed blocks $R1$ and $R2$ are used in pairs and are cascaded together for the design of testable reversible logic circuits. Arbitrary binary functions are implemented by the $R1$ gate, and the $R2$ gate implements the online testability. As well as duplicating inputs, the $R2$ gate also generates the parity of its inputs, and generates the complement of its input R at S only if the inputs remain unchanged. Thus $R = S$ indicates a fault in the circuit. Figure 3 shows the proposed $R1$ and $R2$. The same work also proposes a rail checker circuit to detect and carry forward flaws in testable blocks in a larger circuit.

(Mahammad and Veezhinathan 2010) propose an approach to directly construct an online testable circuit from a given reversible circuit. The authors propose two steps for this construction. In the first step, every $n \times n$ reversible gate G in that circuit is transformed into a new $(n+1) \times (n+1)$ Deduced Reversible Gate DRG (G). This can be achieved

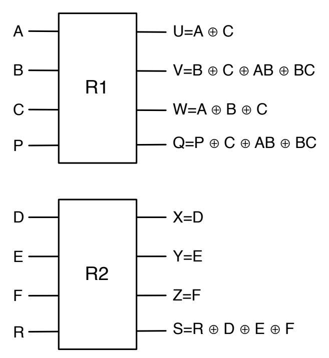


FIGURE 3. $R1$ AND $R2$ GATES (VASUDEVAN, LALA, DI, AND PARKERSON 2006).

by adding an extra input bit P_{ia} and the corresponding output bit P_{oa} to the reversible gate G, maintaining the original functionality of the gate. Figure 4 shows the conversion.

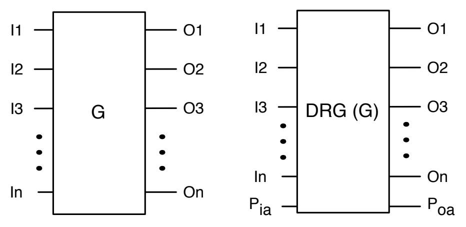


FIGURE 4. G GATE AND DEDUCED REVERSIBLE LOGIC GATE DRG(G).

In the second step, a testable reversible cell of G, TRC(G) is constructed by cascading the DRG(G) with a deduced identity gate. An identity gate is an $n \times n$ reversible gate where all the inputs are simply copied to the outputs. For instance if X is an $n \times n$ identity gate, then its deduced identity gate called DRG(X) can be easily constructed from X by adding the extra input bit P_{ib} and the corresponding output bit P_{ob}. The DRG(G) and DRG(X) are cascaded by connecting the first n outputs of DRG(G) and the first n inputs of DRG(X). The new $(n+2) \times (n+2)$ block is called a Testable Reversible Cell, TRC(G) which is the final online testable gate. The construction is shown in Figure 5.

Another work (Nayeem and Rice 2012) presents more current research related to online testing of ternary Toffoli gates. In this paper the authors discuss an online testing approach as applied to a cascade of ternary Toffoli gates. In this approach one additional line (referred to as a parity line) is added to the circuit. Each input and output line of the circuit is connected to the additional parity line by one 2-qutrit Toffoli gate and one 2-qutrit modified Toffoli gate.

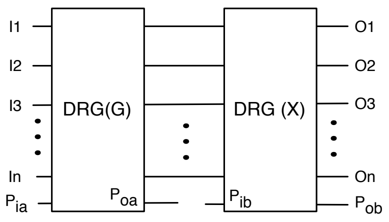


FIGURE 5. CASCADED DRGS TO FORM A TRC.

Each n-qutrit Toffoli gate of the original circuit is replaced by a $n+1$-qutrit Toffoli gate and connected to the parity line. The resultant circuit is proved to be online testable.

Testable Ternary Logic Block

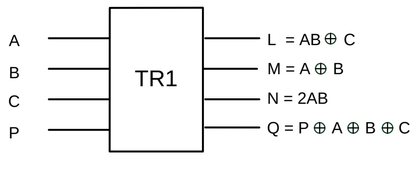
In this section we describe our design of the online testable ternary reversible logic block. The basic concept is similar to that described in (Vasudevan, Lala, Di, and Parkerson 2006), however we have extended this work to ternary and proposed different approaches to reduce the implementation cost. As first described in (Rahman and Rice 2011b) and extended in (Rahman and Rice 2011a), the online testable ternary reversible logic block (TR) is composed of two individual blocks referred to as TR1 and TR2, which are described below.

TR1 Block

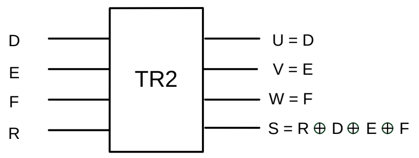
The logic needed for the functionality of the circuit is implemented using the proposed TR1 block, as shown in Figure 6(a). All operations in this figure and subsequent figures are ternary operations as defined in Table 1. The outputs L, M and N implement the ternary operations and error detection is performed using the output Q. $Q = P \oplus A \oplus B \oplus C$ should be equivalent to the sum of the outputs L, M and N, and input P. The operations are independent of the input P and P is set to an arbitrary value 0 which is used in the testability feature. The basic ternary operations such as AND, EXOR, successor, negation/complement, mod-sum and mod-difference can be implemented by the TR1 block. As an example, to find the successor $\overrightarrow{x}$ where $x = 1$, we need to set the inputs $A = 1, B = 1, C = 0$ and $P = 0$ in TR1 to produce the desired result of $M = A \oplus B = 2$.

TR2 Block

The online testing is achieved by the TR2 block. Figure 6(b) shows the functionality of the TR2 block. A single bit error is detected using the output S when TR2 is cascaded with a TR1 block to create an online testable block (TR). In a



(a)



(b)

FIGURE 6. (A) THE TR1 BLOCK AND (B) THE TR2 BLOCK.

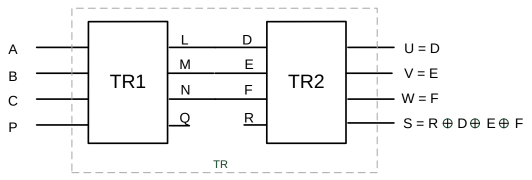
TR block the TR2 block receives the first three outputs of the TR1 block as inputs and generates copies of each, along with the error detecting output S. If any error occurs inside the TR2 block it is reflected on the output S.

Online Testable Block (TR)

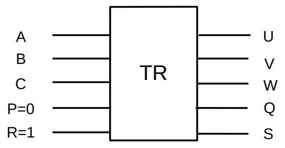
TR1 and TR2 are cascaded together to construct an online testable ternary reversible block (TR). The TR block that is formed by connecting a 4×4 TR1 to a 4×4 TR2 by their first three outputs and inputs is a 5×5 block. In an online testable block, input P of the TR1 block and input R of the TR2 block must be set in such a way that $R = \overrightarrow{P}$. Since we must set R to be a successor of P, we can set $P = 0$ and $R = 1$, $P = 1$ and $R = 2$ or $P = 2$ and $R = 0$. For regular operation we have chosen to set $P = 0$ and $R = 1$. Figure 7(a) shows the configuration and Figure 7(b) shows the block diagram of TR.
TR1 takes ternary logic values for implementing the desired functionality at A, B, C, and P is set to 0. Output Q generates $P \oplus A \oplus B \oplus C$ where $P = 0$. TR1 has been constructed so that $A \oplus B \oplus C$ can be equal to $L \oplus M \oplus N$ only if no error occurs inside TR1. TR2 transfers the input values D, E, F to outputs U, V and W, where $D = L$, $E = M$ and $F = N$. TR2 also generates the error detecting output at S. The output S will be the successor of Q if no error occurs in TR1 and TR2. If any error occurs, output S will no longer be a successor of Q.

Limitation of the TR block

In ternary GF3 logic, adding 3 to a variable leaves the variable unchanged, *e.g.*, $A \oplus 3 = A \oplus 1 \oplus 1 \oplus 1 = A$. Let us assume that a copy of the input A is required. If inputs B and C are set to 1, then we have $U = A \oplus 1$ and $V = A \oplus 1$ at the

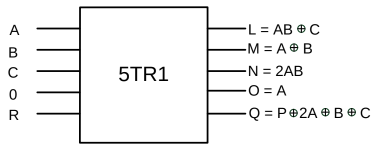


(a)

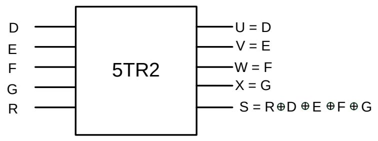


(b)

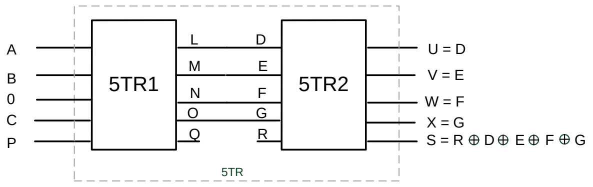
FIGURE 7. (A) INTERNAL CONFIGURATION OF THE ON-LINE TESTABLE TERNARY REVERSIBLE BLOCK (TR) AND (B) ITS BLOCK DIAGRAM.



(a)



(b)



(c)

FIGURE 8. (A) $5 \times 5TR1$ (5TR1) BLOCK, (B) $5 \times 5TR2$ (5TR2) BLOCK, AND (C) $6 \times 6TR$ (5TR) BLOCK.

outputs of a TR. If we repeat this process thrice using three TR blocks, each time providing $A = A \oplus 1$ from the previous block and $B = C = 1$, at the end of the third operation we have $U = A \oplus 1 \oplus 1 \oplus 1 = A$ and $V = A \oplus 1 \oplus 1 \oplus 1 = A$. Thus we have produced a copy of input A.

In this design five TR blocks and two rail checkers (RC) are required to generate a single copy. Each TR block requires 61 M-S gates and each RC requires 18 M-S gates. Therefore, the number of gates required for a single copy operation is $3 * 61 + 2 * 18 = 219$, which is very large. This inefficiency of the TR block can be improved in different ways, as discussed in the following sections.

Upgraded design of the TR1, TR2 and TR blocks (5TR1, 5TR2, 5TR)

One disadvantage of the original TR block is that it requires three TR blocks and two rail checkers to generate a copy of one input. This creates a large amount of overhead. Hence, the 4×4 TR1 block that is part of the TR block can be changed into a 5×5 block by adding an additional constant input 0 and an output $O = A$. The output function Q will also be changed. To distinguish between the 4×4 TR1 block and the 5×5 TR1 block in this literature they are referred as 4TR1 and 5TR1 correspondingly; similary, the 4×4 TR2 block is referred to as 4TR2. We define the new 5×5 TR1 block (5TR1) as shown in Figure 8(a) and 5×5 TR2 block (5TR2) as shown in Figure 8(b). To distinguish between the 4×4 TR2 block and 5×5 TR2 block we refer to them as 4TR2 and 5TR2 respectively.

Two-pair two-rail checker

As described in (Nikolos 1998), error checking is often implemented using one of two main techniques: parity codes or two-pair two-rail checkers. Rail checkers compare the outputs from more than one identical system. This process is also used to reduce the number of error detecting outputs (Nikolos 1998). The two-pair two-rail checker receives two pairs of inputs and generates a pair of outputs that indicates if the prior operations were fault-free or faulty. A ternary two-pair two-rail checker should receive two pairs of inputs from two TR blocks, compare them and indicate in the outputs if any flaw was identified by either of those TR blocks. This concept has also been used in the reversible context by other authors such as in (Farazmand, Zamani, and Tahoori 2010) and (Vasudevan, Lala, Di, and Parkerson 2006).

The purpose of the rail checker is to detect the existence of a flaw, if there is any, in the TR blocks attached to the rail checker. This is achieved by checking whether the outputs of the blocks are successors or not. The rail checker generates two outputs where one is successor to another if the attached blocks are fault-free. Since these outputs may be used afterwards to cascade additional rail checkers, successor outputs are generated to represent the fault-free situation. Otherwise, if the rail checker detects any flaw in the attached TR blocks, the design guarantees that the outputs generated will never be the successor of each other.

The rail checker is designed in such a way that if the input pairs are successors, *i.e.* $y0 = \overrightarrow{x0}$ and $y1 = \overrightarrow{x1}$, the rail checker will generate $X3 = 1$ and $X4 = 2$, so that $X4 = \overrightarrow{X3}$, otherwise it generates outputs where $X4 \neq \overrightarrow{X3}$. There is no specific reason for choosing $X3 = 1$ and $X4 = 2$. Any two successors could have been used, for example $X3 = 0$ and $X4 = 1$ or $X3 = 2$ and $X4 = 0$.

Example Circuit

We implement the GFSOP expression $F = ab \oplus cd$ to demonstrate that the proposed blocks in this work can successfully implement a ternary GFSOP. Figure 9 shows the realization of function F using the proposed online testable ternary reversible blocks. The final outputs of the second rail checker can be used for cascading if the function needs to be further extended.

Internal Designs

5TR Block

The designs of the TR1 and TR2 block have gone through a number of evolutions as we continue to improve the design. The final versions of the 5×5 TR1 (5TR1) and 5×5 TR2 (5TR2) blocks are designed using a number of 2×2 Feynman gates and a cascade of GTGs. To reduce the number of M-S gates in the new design of 5TR1 a new gate named the Modified-Feynman (MF) gate has been proposed, as illustrated in Figure 10.

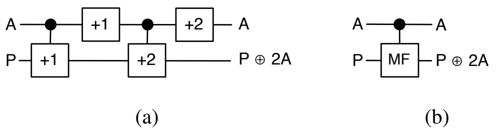


FIGURE 10. (A) INTERNAL DESIGN OF THE MODIFIED FEYNMAN (MF) GATE AND (B) SYMBOL FOR THE MF GATE.

Figure 11 shows the internal design of the 5TR1 block. This design includes four 2×2 Feynman gates, one MF gate, one dual-shift gate and four GTGs. It can be verified from the internal structures of the constructing gates that the number of M-S gates required for each gate is as follows:

- a Feynman gate requires 4 M-S gates,
- the dual-shift gate requires one M-S gate,
- the MF gate requires 4 M-S gates, and
- each GTG requires 7 M-S gates (Khan and Perkowski 2007).

Therefore, the total number of M-S gates required to implement the 5TR1 block is 49 which is 55.4% less than the first design of the 4TR1 block discussed in (Rahman and Rice 2011b). However the most significant achievement of this design is the number of garbage outputs, which is zero for this design. However it should be mentioned here that when this block is used to realize a ternary circuit, some of the outputs of the TR block may become garbage outputs if they are not used in any further operation.
Figure 11 shows the new 6×6 TR block consisting of a 5TR1 cascaded with a 5TR2 and addressed as the 5TR block. The total number of M-S gates required to construct this 5TR block is $(49 + 16) = 65$.

Two-pair two-rail checker

To implement the rail checker discussed above, an elementary gate (E) with two controlling inputs and one controlled input has been designed. The design of the E gate is based on the architecture of the 1-qutrit ternary comparator circuit proposed in (Khan 2008). Figure 12 (a) and (b) shows the block diagram of the E gates. The behavior of the Z-transformation depends on whether the second input (y) is successor to the first input (x) or not. If the second input is a successor of the first input, the Z-transformation changes the controlled input, as previously described, to either a 1 or a 2. Otherwise, 0 is passed unchanged through to K. There are actually two E-gate designs, one with an output of 1 if $x_0 = \overrightarrow{y}_0$ and one with an output of 2 if $x_1 = \overrightarrow{y}_1$ where x_0, y_0 and x_1, y_1 are outputs from two attached TR blocks. The E-gate with an output of 1 is denoted by E_a in Figure 12 (a) and the E-gate with an output of 2 is denoted by E_b in Figure 12 (b).
The controlling inputs (x, y) of the two E-gates comprise the four inputs of the rail checker circuit and the controlled outputs K_a and K_b comprise the two outputs of the rail checker. Each controlled input is a constant input set to the value zero. One each of E_a and E_b are used to construct the ternary rail checker.
The first E-gate (E_a) receives $x0$ and $y0$ as its inputs and generates $K_a = 1$. This is generated by the Z(+1) transformation on the controlled input, but only if $y0 = \overrightarrow{x0}$. The second E-gate (E_b) that takes $x1$ and $y1$ as inputs generates $K_b = 2$ at the output by applying the Z(+2) transformation on the controlled input in the case where $y1 = \overrightarrow{x1}$. Figure 12 (c) shows the block diagram of the internal architecture of the ternary rail checker. From this figure we can see that since error detecting signals $X3$ and $X4$ are generated out of two physically separated E-gates any single bit error in internal lines will affect one of the two outputs, but not both.

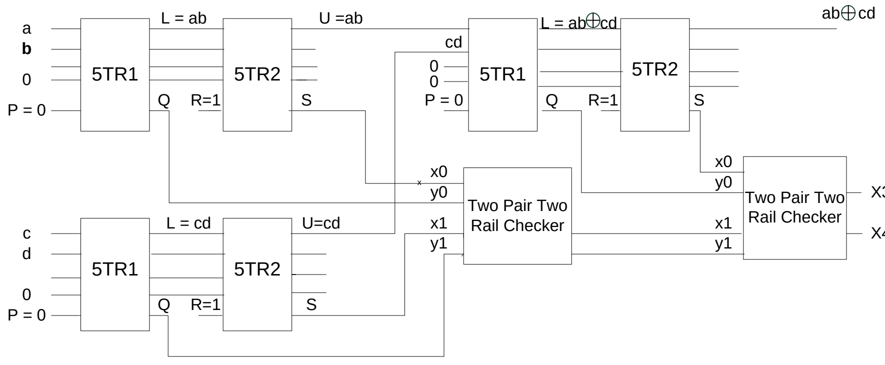


FIGURE 9. ONLINE TESTABLE TERNARY REVERSIBLE IMPLEMENTATION OF FUNCTION $F = ab \oplus cd$.

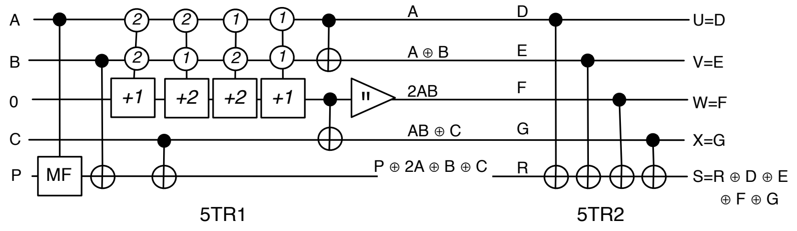


FIGURE 11. 6×6 TR (5TR) BLOCK WITH FUNCTIONALITY FOR THE 5TR1 AND 5TR2 SHOWN ON THE LEFT AND RIGHT RESPECTIVELY.

Approaches to Utilizing the Online Testable Block

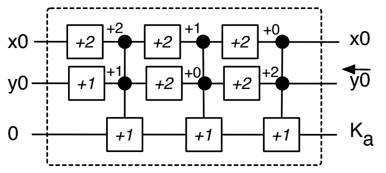
The benchmark circuits from (Denler, Yen, Perkowski, and Kerntopf 2004) can be implemented using the proposed 5TR block and the rail checker. However in reversible logic it is often necessary to "copy" input values to avoid fanout. Therefore, it is possible to reduce the cost of implementation in terms of the number of gates if the copy operation is performed intelligently rather than using multiple 5TR blocks for generating copies. In the following sections distinct blocks are proposed to implement the copy operation.

Method 1: Combination of 5TR block and online testable copy gate

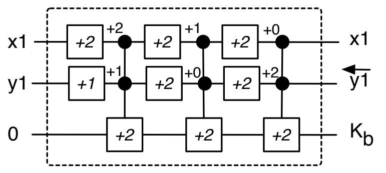
This block can be used for generating copies of a single input but will also incorporate the online testability feature. The maximum number of outputs excluding the error checking output is four in a 5×5 block. Therefore the online testable copy gate can generate four copies of a single input. However, generating four copies involves more M-S gates as well as increase the number of garbage outputs if only two copies are required, as in most of the cases. Thus, we have limited our design of TR$_{copy}$ to generate only three copies of the input, although the flexibility of designing a copy gate for four copies still exists. Figure 13(a) shows the internal design of the online testable copy gate. For the copy operation 5TR1 is replaced with the new 5×5 TR$_{copy}$ block and cascaded with a 5TR2 block for the online testability feature. The new 5TR block constructed from TR$_{copy}$ and 5TR2 is referred to as TR$_c$ which generates three copies of a single input. The error detection policy is identical to that used in both the 4TR and 5TR blocks. TR$_{copy}$ requires 8 and 5TR2 requires 12 M-S gates. Therefore, a TR$_c$ block requires 24 M-S gates in total.

Method 2: Combination of 5TR block, TR$_c$ and multicopy gate (TR$_{mc}$)

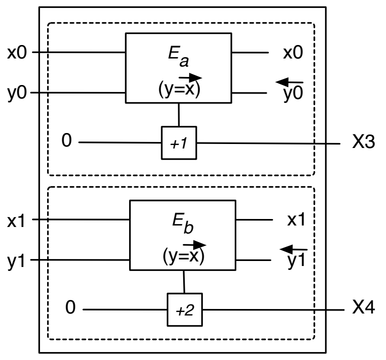
Copies of multiple variables are required for some logic operations. For example, two copies of A, two copies of B and two copies of C are required for the benchmark 3CyG2 which implements the function $ab \oplus bc \oplus ca$. Three TR$_c$ blocks are required to generate the copies in this benchmark. To avoid this situation the design of TR$_{copy}$ can be modified to generate two copies of two different variables. Figure 13(b) shows the configuration. This multi copy gate is referred as TR$_{multicopy}$. 16 M-S gates are required to realize this gate. Online testability is incorporated by cas-



(a)



(b)

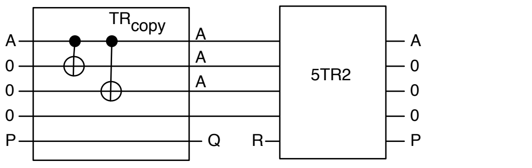


(c)

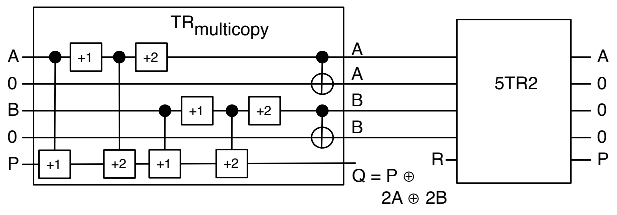
FIGURE 12. (A) THE INTERNAL STRUCTURE OF THE E_a GATE, (B) THE INTERNAL STRUCTURE OF THE E_b GATE, AND (C) THE INTERNAL ARCHITECTURE OF THE TWO-PAIR TWO-RAIL CHECKER CIRCUIT.

cading the TR$_{multicopy}$ and the 5TR2 block. The new cascaded block is referred as TR$_{mc}$. Therefore, a TR$_{mc}$ block requires 32 M-S gates in total to generate one copy for each of the two variables whereas TR$_c$ would require 48 M-S gates to perform the same operation. Although the number of M-S gates is increased in TR$_{mc}$, this block can reduce the number of M-S gates to 33% where copies for multiple variables are necessary. In the 3CyG2 benchmark function, $24*3 = 72$ M-S gates are required to implement the copy functions, whereas 56 M-S gates are required if one TR$_c$ and one TR$_{mc}$ are used. In another benchmark circuit, $4CyG2$, which implements $ab \oplus bc \oplus cd \oplus da$, $24*4 = 96$ M-S gates are required if TR$_c$ is used whereas implementation using two TR$_{mc}$ requires only 64 M-S gates. For the best result a combination of TR$_c$ and TR$_{mc}$ blocks can be used.

The number of M-S gates required to construct each proposed block is shown in Table 5. This table can be used to determine the quantum cost of any circuit constructed using the proposed blocks.



(a)



(b)

FIGURE 13. INTERNAL DESIGNS OF THE (A) 6×6 TR$_c$ BLOCK AND (B) 6*6 TR$_{mc}$ BLOCK.

TABLE 5. NUMBER OF M-S GATES REQUIRED TO IMPLEMENT THE PROPOSED BLOCKS.

Blocks	Number of M-S gates	
4TR	4TR1	49
	4TR2	12
	Total (4TR)	61
5TR	5TR1	49
	5TR2	16
	Total (5TR)	65
TR$_c$	TR$_{copy}$	8
	5TR2	16
	Total (TR$_c$)	24
TR$_{mc}$	TR$_{multicopy}$	16
	5TR2	16
	Total (TR$_{mc}$)	32
RC	18	

Discussion

Comparisons of the Approaches

Table 6 presents the number of M-S gates, which is used as the cost metric in this work, required to realize the benchmark circuits provided by Denler, Yen, Perkowski, and Kerntopf (2004) using the design approaches discussed so far. The cost includes the number of M-S gates required to implement the testable blocks as well as the rail checkers.

TABLE 6. COMPARISON OF NUMBER OF M-S GATES FOR IMPLEMENTING THE TERNARY BENCHMARK CIRCUITS.

Benchmarks	Proposed Methods 5TR & RC	5TR,TR$_c$& RC	5TR,TR$_c$,TR$_{mc}$& RC	Non-testable ternary gates	Lowest overhead
2CyG2 ($2ab$)	**65**	**65**	**65**	**37**	164%
3CyG2 ($ab+bc+ca$)	**646**	**523**	**489**	**92**	531%
a2bccG (a^2+bc+c)	**480**	**398**	**364**	**96**	379%
ProdG2 (ab)	**65**	**65**	**65**	**28**	217%
ProdG3 (abc)	**148**	**148**	**148**	**56**	250%
SumG2 ($a \oplus b$)	**65**	**65**	**65**	**4**	1525%
SumG3 ($a \oplus b \oplus c$)	**148**	**148**	**148**	**8**	1750%

Table 6 shows that the fewest M-S gates are required when using the non-testable ternary gates. This is unsurprising because incorporating testability features always adds some overhead to the circuit. However testable circuits are far more robust and fault tolerant than non-testable circuits, hence the tradeoff between the overhead and testability is considered to be justifiable. It can be seen from Table 6 that the basic approach using only 5TR blocks and rail checkers requires more M-S gates than the other approaches for benchmarks 3CyG2 and a2bccG but almost equal numbers of gates for the remaining benchmarks. The approach of using TR$_c$ blocks along with 5TR blocks generates the best result whenever the circuit requires more than two copies of an input variable. The design of TR$_c$ can be easily upgraded to generate 4 copies of a single input variable, although the upgraded design would require 8 more M-S gates. If the function to be implemented consists of several variables which require 4 copies each, the upgraded design would further reduce the required number of gates in total. However the proposed design is the best for functions consisting of variables which require at most 3 copies each.

The approach using both TR$_c$ and TR$_{mc}$ blocks along with the 5TR blocks requires the lowest number of M-S gates to implement the benchmark circuits. This approach is the most efficient since it uses a best fit approach to use either of the TR$_c$ or TR$_{mc}$ block to achieve the best result. This approach is the most efficient if the design requires at most two copies of the input variables and also requires copies for a large number of variables. It is evident from Table 6 that this approach requires a lower number of M-S gates for benchmarks $2CyG2$, $2CyG3$ and $a2bccG$ and an equal number of gates as compared to the previous approach for the remaining benchmarks.

Overhead Analysis

In this work overhead can be approximated as the total number of extra elementary gates required to be added for the realization of the ternary online testable circuit as compared to the original gate count of the non-testable realization of the same circuit.

We have compared our overhead with that resulting from the implementation of a binary online reversible testing approaches (Nayeem 2011). The cost metric used in both the binary and ternary online testing approaches is the number of elementary gates required to realize a logic function. Table 7 shows the comparison of average overheads among the two binary and the proposed ternary online testing approaches.

Although the average overhead of our approach is higher than the other approaches, the following should be considered. From the last column of Table 6 it can be seen that the overhead costs for the benchmarks $SumG2$ and $SumG3$ are significantly higher than the other overheads. These costs are significant in increasing the average overhead of our approach. If the average overhead is calculated excluding the benchmarks $SumG2$ and $SumG3$, the average overhead for our approach is reduced to 308%. We note that the non-testable approaches for benchmarks $SumG2$ and $SumG3$ required less than 8 gates whereas the minimum number of gates required for a ternary online testable design is 61. Therefore, we can predict that our approach is likely to minimize the overhead costs for larger Toffoli circuits.

The approach discussed in Nayeem and Rice (2011)has very low quantum cost; however our design has some advantages over the design in that work. Their design is based on the fact that there should be an existing Toffoli gate cascade. Therefore if there is no existing gate cascade implementation for the function, or if the existing gate cascade

TABLE 7. OVERHEAD COMPARISON.

Approaches	Average overhead for testability
(Vasudevan, Lala, Di, and Parkerson 2006)	312.28% (Nayeem 2011)
(Farazmand, Zamani, and Tahoori 2010)	388.67% (Nayeem 2011)
(Nayeem 2011; Nayeem and Rice 2011)	4.21%
Our approach	688%

uses a different type of gate, their approach can not be applied. Our design, however, can realize any ternary function in online testable form as long as the function is described as a TGFSOP. However as a result of our comparisons we are motivated to continue our research to find ways to improve our technique to achieve compatible performance.

Conclusion & Future Work

In this paper we present a design for an online testable ternary reversible logic block. Multiple such blocks can be used in conjunction with two-pair two-rail checkers to implement any ternary reversible circuit. Different design approaches are suggested, with discussion as to how each approach might best be utilized.

The fault model used in this work is a single-bit fault model, meaning that any single error in a block can be detected and will be propagated to the outputs through the use of the two-pair two-rail checkers.

One of the major concerns of reversible logic synthesis is to keep the number of the input constants as few as possible. In our proposed designs, we have only one constant input. The other two major concerns of reversible logic synthesis are to keep the number of garbage outputs and the length of gate cascades as small as possible. In our designs we minimize the number of garbage outputs by using the internal gates as intelligently as possible. The length of cascaded gates is also kept at a minimum by replacing 4TR or 5TR blocks with TR_c and TR_{mc} blocks, which are designated for copy operations, whenever required. In this work we have detected and resolved the limitation of the design first proposed in (Rahman and Rice 2011b) and upgraded in (Rahman and Rice 2011a). We have also proposed some additional blocks for the copy operation, which may also result in an additional reduction in the circuit's quantum cost.

Future work will include further analysis as to the fault coverage and examination as to how the designs could be modified to work with other fault models. In addition a synthesis process must be developed with heuristics for making the best choice among the alternate blocks in order to implement circuits with the lowest cost.

Acknowledgments

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

References

Bennett, C. H. (1973). Logical reversibility of computation. *IBM Journal of Research and Development 6*, 525–532.

Chowdhury, A. R. (2006). A new approach to synthesize multiple-output functions using reversible programmable logic array. In *Proceedings of the IEEE 19th International Conference on VLSI Design*, Hyderabad, India, Jan. 2-7, pp. 311–316.

Denler, N., B. Yen, M. Perkowski, and P. Kerntopf (2004). 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)*, June, Tamecula, California, USA.

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

Frank, M. P. (2005). Introduction to reversible computing: motivation, progress, and challenges. In *Proceedings of the 2nd Conference on Computing Frontiers*, New York, NY, USA, pp. 385–390. ACM.

Jha, N. and S. Gupta (2003). *Testing of Digital Systems*. The Press Syndicate of the University of Cambridge.

Khan, M. H. A. (2004). Quantum realization of ternary Toffoli gate. In *Proceedings of the 3rd International Conference on Electrical and Computer Engineering*, pp. 264–266. 28–30 December, Dhaka, Bangladesh.

Khan, M. H. A. (2008, May). Design of reversible/quantum ternary comparator circuits. *Engineering Letters 16*(2), 178–184.

Khan, M. H. A. (2009). Quantum realization of multiple-valued Feynman and Toffoli gates without ancilla input. In *Proceedings of the 39th International Symposium on Multiple-Valued Logic (ISMVL)*, 21-23 May, Naha, Okinawa, Japan, pp. 103–108.

Khan, M. H. A. and M. A. Perkowski (2003). Multi-output ESOP synthesis with cascades of new reversible families. In *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technologies*, March, pp. 144–153.

Khan, M. H. A. and M. A. Perkowski (2007). Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *Journal of Systems Architecture 53*, 453–464.

Khan, M. H. A., M. A. Perkowski, and M. R. Khan (2004). Ternary Galois field expansions for reversible logic and Kronecker decision diagram for ternary GFSOP minimization. In *Proceedings of the 34th International Symposium on Multiple-Valued Logic (ISMVL)*, Toronto, Canada, 19-22 May, pp. 58–67.

Kole, D. K., H. Rahman, D. K. Das, and B. B. Bhattacharya (2010). Synthesis of online testable reversible circuit. In *Proceedings of the IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, Vienna, 14–16 April, pp. 277–280.

Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development 5*, 183–191.

Mahammad, S. N. and K. Veezhinathan (2010, January). Constructing online testable circuits using reversible logic. *IEEE Transactions on Instrumentation and Measurement 59*(1), 101–109.

Maslov, D. and G. W. Dueck (2003). Garbage in reversible design of multiple output functions. In *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technologies*, March, pp. 162–170.

Maslov, D. and G. W. Dueck (2004). Reversible cascades with minimal garbage. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 23, pp. 1497–1509.

Miller, D. and M. Thornton (2008). *Multiple Valued Logic: Concepts and Representations*. Morgan and Claypool Publishers.

Nayeem, N. and J. E. Rice (2011, August). A simple approach for designing online testable reversible circuits. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 274–279. (best paper award).

Nayeem, N. M. (2011). Synthesis and testing of reversible Toffoli circuits. Master's thesis, University of Lethbridge.

Nayeem, N. M. and J. E. Rice (2012). A new approach to online testing of TGFSOP-based ternary Toffoli circuits. In *Proceedings of the International Symposium on Multiple-Valued Logic (ISMVL)*, 14–16 May, Victoria, BC, Canada, pp. 315–321.

Nikolos, D. (1998, Feb./April). Self-testing embedded two-rail checkers. *Journal of Electronic Testing: Theory and Applications 12*(1–2), 69–79.

Polian, I., T. Fiehn, B. Becker, and J. P. Hayes (2005). A family of logical fault models for reversible circuits. In *Proceedings of the Asian Test Symposium (ATS)*, Los Alamitos, CA, USA, pp. 422–427. IEEE Computer Society.

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

Rahman, M. R. and J. E. Rice (2011b). Online testable ternary reversible circuit. In *Proceedings of the Reed-Muller Workshop*, May 25–26, Tuusula, Finland, pp. 71–79.

Shende, V. V., A. K. Prasad, I. L. Markov, and J. P. Hayes (2003, June). Synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22*(6), 710–722.

Vasudevan, D. P., P. K. Lala, J. Di, and J. P. Parkerson (2006, April). Reversible logic design with online testability. *IEEE Transactions on Instrumentation and Measurement 55*(2), 406–414.

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