Online Testable Ternary Reversible Circuit

Md. R. Rahman
Dept. of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
Email: md.rahman7@uleth.ca

J. E. Rice
Dept. of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
Email: j.rice@uleth.ca

***Abstract*— In recent years ternary reversible logic has gained more attention because of its enormous potential in different fields, in particular quantum computing. In order for any future technologies to be reliable they must be testable, thus developing testing techniques in this area is of great importance. In this work we propose a new approach for creating an online testable ternary reversible circuit. To our knowledge this is the first such circuit of its type to be proposed. The proposed testable block can be used to implement almost all of the ternary logic operations and is capable of testing the reversible ternary network in real time (online). The error detection unit is also constructed in a reversible manner, and thus the entire reversible testable block can be used to construct complete reversible circuits.**

Keywords: ternary logic, reversible logic, online testable circuits, two pair two rail checker

I. Introduction

The area of fault detection for reversible circuits is fairly new although works such as [1], [2] and [3] are beginning to address this area. A great deal of effort has focused on finding ways to design and build ternary reversible gates and circuits. However the areas of modeling faults of ternary reversible circuits and detection of the faults in real time have yet to be addressed, despite the clear significance of these areas. There is a great deal of motivation to do so, in particular since quantum computing is rapidly emerging as an area of importance. Since quantum logic operations are reversible, studying how to build online testable ternary reversible circuits may assist in the development of test methods for quantum circuits.

In this work we present designs for two ternary blocks: one to implement basic ternary logic functions and one to implement online testability to detect any single bit error. In order to check for flaws in a number of such blocks in a circuit a two pair two rail checker circuit is also designed using basic ternary gates.

II. Background & Motivation

A. Ternary Reversible Logic

The primary motivation towards reversible computing is based on the fact that all irreversible logic operations produce a fundamental amount of waste energy. This leads to the scenario that the improvement of computer performance will soon reach its limit, and so reducing energy dissipation is a major concern. According to Frank [4],

> ...computers based mainly on reversible logic operations can reuse a fraction of the signal energy that theoretically can approach arbitrarily near to 100% as the quality of the hardware is improved...

Bennett's work suggests that all future technologies must adopt reversibility to reduce energy dissipation [5], and this phenomenon holds true whether the logic model used is two-valued or multi-valued.

Circuits can sometimes show better characteristics if multiple-valued logic (MVL) concepts are used instead of traditional binary (two-valued) concepts [6]. [7] introduces ternary reversible/quantum research as a very new research area, but with motivation based on both quantum and multiple-valued computing. [8] discusses the fact that ternary functions can be easily expressed using Ternary Galois Field Sum of Product (TGFSOP) expressions, regardless of the number of input variables. Quantum 1-qudit and 2-qudit gates are also realizable using existing quantum technology [8].

We introduce the ternary reversible gates used in this work in Sections III and V. We note that [9] provides an excellent overview of the notation used in this area as well as further background and explanation.

B. Fault Models

Recognizing and modeling the behaviour of possible defects in a circuit is known as fault modeling. [10] discusses how different levels of abstraction in circuit design can result in different fault models. We briefly describe these varying models as discussed in [10].

- Behavioural fault models. Viewing a system from a behavioral point of view results in a variety of behavioural fault models. These types of fault models stem from the constructs used in a at a behavioural description, such as a language like VHDL.
- Functional fault models. These focus on the faults in functional blocks of the system, with the goal of ensuring fault-free behaviour of the blocks. In RAM, for instance, a multiple write fault is an example of a type of functional fault.
- Structural fault models. These deal with faults that occur at the interconnections of a design. The single stuck-at fault is a commonly used fault model in this category.
- Switch level fault models. These models focus on the transistor level of a circuit. Two popular fault models in

this category are stuck-open and stuck-on fault models. A permanently non-conducting faulty transistor generates a stuck-open fault while a permanently conducting faulty transistor results in stuck-on fault.

- Geometric fault models. These models identify the flaws in the layout of a chip, or manufacturer defects. Shorted lines in a chip layout are modeled by the bridging fault model in this category.

[11] characterizes the type of fault that the proposed circuit will identify as a *single bit error*. A single bit error occurs whenever the value of an output has been changed because of an internal circuit error. This is referring to a block of output values, within which the change on any one output line can be identified. The design detects any single bit error occurs in the gates within a particular block and propagate the result to the outputs. This type of fault falls best into the functional fault model category.

The ternary online testing blocks we propose in this paper are also designed to identify single bit errors within the testable blocks in a method similar to that proposed for the binary case by [11].

C. *Online Testing*

Online testability is the ability of a circuit to test a portion of the circuit while the circuit is operating [11]. Detecting a fault in operation, the point of occurrence and in some cases, attempt to recover from fault are the major focus of research into online testing [10]. Similar processes for binary reversible logic have been discussed in [10] and [12]. In [10] the authors discuss built-in self-test (BIST) for digital circuits. BIST refers to the design of a circuit to test itself, but not necessarily online, *i.e.* while operating.

In [12] the author discusses the concept of self checking logic and the necessary categories [13]. It proposes three categories for self testable circuits: fault secure, self-testing and totally self testing digital circuits. According to the author, a fault secure digital circuit should posses the characteristic that the output will not be affected by any single bit fault. A self-testing circuit refers to a digital circuit which will result in invalid output(s) for any fault that occurs inside the circuit. A totally self-testing circuit must be both fault secure and self testing [12]. The circuit we propose in this work falls into the self-testing digital circuit category.

D. *Related Work*

In [11] the authors propose online testable logic blocks for binary reversible logic. They propose two gates called R1 and R2, to be used in pairs cascaded together for the design of testable reversible logic circuits. The R1 gate is designed to implement arbitrary Boolean functions. During normal operation the input P is set to 0. The OR operation can be implemented by setting the inputs $A = P = 0$ which produce outputs $V = B + C$ and $W = B \oplus C$. By setting the inputs $C = 1$ and $P = 0$, the EXNOR and NAND functions can be obtained in the outputs such that $U = A', V = (AB)'$ and $W = (A \oplus B)'$. The NOR and AND operations can be obtained by cascading two R1 gates in configurations which are shown and discussed in [11].

The R2 gate is designed to have the online testability features integrated into it. Beside duplicating inputs, the R2 gate also generates the parity of the input pattern in S. It generates the complement of the input R at S only if the inputs remain unchanged. The input R is 1 during normal operation. Hence, if $R = S$ in any situation, it represents the presence of a fault in the circuit and thus detects the flaw. Details are given in [11], but the basic idea is that the R1 gate is used to implement binary functions and the R2 gate is used to detect faults in gates R1 and R2 itself. Figure 1 shows the gates proposed in [11].

[11] also proposes a rail checker circuit to detect flaws in testable blocks in a larger circuit. The rail checker takes inputs from multiple R2 blocks and combines the results so that many parts of the circuit can be checked for faults. Details are given in [11].

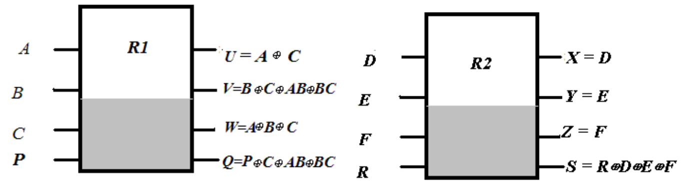


Fig. 1. The two gates proposed in [11] for online testing of reversible circuits.

In another related work, authors of [14] propose a universal reversible logic gate that can be used to construct online testable circuits. A principle similar to that used in [11] and [15] to detect the faults in logic blocks is also used in this approach. The authors propose two steps. In the first step, a new $(n + 1) * (n + 1)$ reversible gate called D RG(G) is constructed from every reversible gate G. This new gate must have the the functionality of the original gate G. In the second step, a testable version of G is produced by cascading D RG(G) with an identity gate. This identity gate is an n*n reversible gate which is *deduced* by mapping all inputs identically to the outputs. The resultant $(n + 2) * (n + 2)$ block is called a testable reversible cell (TRC) [14]. These gates are shown in Figure 2. The TRC has two fault detection properties, aimed at detecting a single bit error. The first is that if $P_{ia} = P_{ib}$ the output indicates an error if output P_{oa} and P_{ob} are complementary. The other property is that if $P_{ia} = \overline{P}_{ib}$ then $P_{oa} = P_{ob}$ indicates an error.

We propose using a similar principle to design online testable ternary reversible logic blocks.

III. Proposed Design of Online Testable Ternary Reversible Logic Block

The testable blocks that we are proposing are designed using ternary reversible building blocks, thus the resultant blocks are also reversible. We do not show the garbage outputs in order to reduce complexity of the figures and since they do not affect the operation. In this paper both basic ternary Toffoli

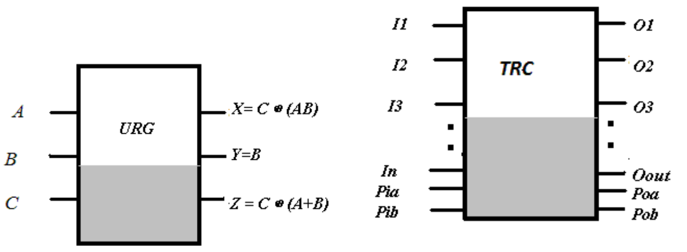


Fig. 2. The gates proposed in [14] for online testing of reversible circuits.

and Feynman gates as well as Muthukrishnan-Stroud (M-S) gates are used. Quantum realizations for the ternary Toffoli and Feynman gates using M-S Gates are discussed in [9], as is background for the M-S gate.

Briefly, the M-S gate is a 2-qudit multivalued gate which can be realized using ion-trap technology. The M-S gate can be represented by the symbol shown in Figure 3. The value of the input $X1$ controls the value of $Y2$, which is the Z transform of $X2$ whenever $X1 = 2$, where $Z \in \{+1, +2, 12, 01, 10\}$; otherwise $Y2 = X2$ [9].

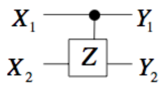


Fig. 3. Symbol for ternary Muthukrishnan-Stroud (M-S) Gate.

Other than the fact that our work is based on ternary reversible logic and the previous proposals were for binary reversible logic, there are two major differences between the design proposed in [11] and our ternary reversible logic blocks. The first is that in order to extend from the binary case to ternary the successor operation is used instead of complement, as will be shown in Section III-C. The second is that, as shown in III-A our components consist of multiple gates rather than individual gates.

A. The TR1 Block

We first describe the TR1 block. This is the ternary reversible testable block that is used to implement the logic needed for the functionality of the circuit.

a) Design of the TR1 Block: The 4*4 TR1 block can be defined as $I = (A, B, C, P)$ and $O = (L = AB \oplus C, M = A \oplus B, N = 2AB, Q = P \oplus A \oplus B \oplus C)$, where I and O are input and output sets respectively. The block is shown in Figure 4. This block is used to implement the five basic functions of two-place and unary ternary operators.

b) Use of the TR1 Block: The TR1 block can be used to implement any of the basic ternary operations: AND, OR (EXOR), successor, negation/complement and mod-difference. The unary operation called successor ($\overrightarrow{x}$) is defined as $(x + 1) mod_p$ [6] where x is the input and p is the cardinality of the logic value (in this case, 3).

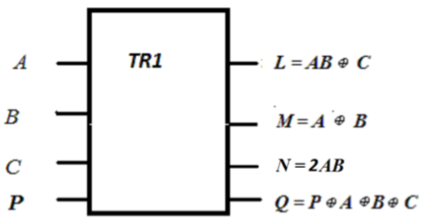


Fig. 4. Configuration of the TR1 Gate.

To implement the successor operation, inputs C and P are set to 0 and A and B are set as the operands of the successor operation *i.e.* $A = x$ and $B = 1$. M provides the desired output. For ternary, where $p = 3$, the truth table of successor is as illustrated in Table I. Figure 5 shows the configuration of the TR1 block to implement the successor operation.

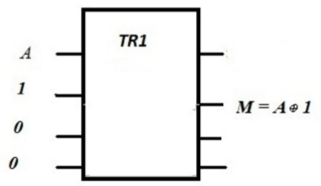


Fig. 5. The successor operation in the TR1 block.

TABLE I
THE SUCCESSOR OPERATION IN THE TR1 BLOCK.

x	$\overrightarrow{x} = x \oplus 1$
0	1
1	2
2	0

c) Example: As an example, let $x = 1$ and assume we wish to find $\overrightarrow{x}$. Setting inputs $A = 1, B = 1, C = 0, P = 0$ in TR1 produces the desired result, $M = A \oplus B = 2$.

B. The TR2 Block

The TR2 block incorporates the online testing features. The $4 * 4$ TR2 block can be defined as $I = (D, E, F, R)$ and $O = (U = D, V = D \oplus E, W = F, S = R \oplus D \oplus E \oplus F)$, where I and O are input and output sets respectively. Outputs U and W are the copies of inputs D and F, can be used in the circuit if necessary. Figure 6 shows the block diagram of the TR2 block. Output S is the EXOR of the inputs of TR2 block. This output is used to detect any single bit error when TR2 is cascaded with a TR1 block to form an online testable circuit. In an online testable design the TR2 block receives the first three outputs of the TR1 block as its inputs.

C. The Online Testable Block

To construct an online testable ternary reversible block (TR), the TR1 and TR2 blocks are cascaded together. When the TR1 and TR2 blocks are used to construct an online testable block, input P of the TR1 block and input R of the TR2 block are set

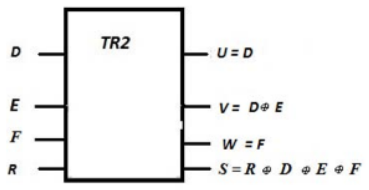


Fig. 6. The TR2 block.

so that $P = \overrightarrow{R}$. For normal operation we simply set $P = 0$ and $R = 1$. Figure 7 shows the configuration and Figure 8 shows the block diagram. TR1 takes ternary logic values at

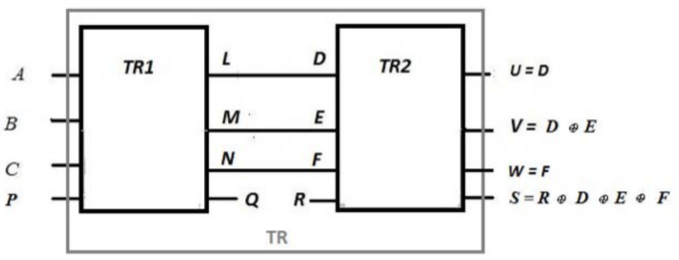


Fig. 7. Configuration of the online testable ternary reversible block TR.

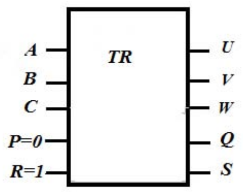


Fig. 8. Online testable ternary reversible block TR.

A, B, C as its inputs and we set $P = 0$. The inputs A, B and C are set depending on the required operation as discussed in Section III-A. At its Q output, TR1 explicitly generates the value of the expression $P \oplus A \oplus B \oplus C$ where $P = 0$. $A \oplus B \oplus C$ should be equal to $L \oplus M \oplus N$ *only if no flaw occurs inside TR1 and the inputs remain unchanged.* TR2 is used to transfer the input values D, E, F to outputs U, V and W where $D = L$, $E = M$, $F = N$ as well as generate the error detecting bit at the output S which should be the successor of Q if no flaw occurs in TR2. S is the EXOR of the inputs of TR2 when $R = 1$.

The error detection principle used by the online testable block is relatively simple. The values at output S should be the successor of Q's value if all other inputs of the block are unchanged.

d) Example: Let us say $X = A \oplus B \oplus C$ and $Y = D \oplus E \oplus F$. Then $Q = P \oplus X = 0 \oplus X$ and $S = R \oplus Y = 1 \oplus Y$ since $P = 0$ and $R = 1$ during normal operation. If the inputs of the block remain unchanged all through the operations, X would be equal to Y, *i.e.* $X = Y$, which results in $S = \overrightarrow{Q}$. Otherwise, if any single bit error has occurred in the inputs, it results in $S \neq \overrightarrow{Q}$ which in turns detects the existence of flaws in the circuit.

If the fault occurs to any of the inputs within the TR1 block, the values of the inputs will be different in calculating L, M,N than Q. Because of the internal structure of the TR1 block, L, M and N will use the flawed input value, whereas Q will use the original value provided at the very beginning, or vice versa. The result will be that the outputs Q and S are not successor, indicating a flaw. The relevant portion of the truth tables of the TR1 and TR2 are shown in Table II.

TABLE II
A SUBSET OF THE TRUTH TABLES FOR (A) TR1 BLOCK AND (B) TR2 BLOCK.

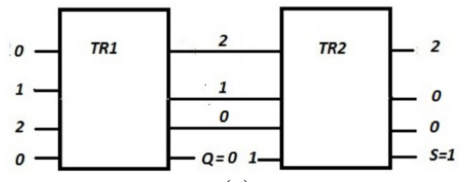
Input					Output			
A	B	C	P		L	M	N	Q
0	0	0	0		0	0	0	0
0	0	1	0		1	0	0	1
0	0	2	0		2	0	0	2
0	1	0	0		0	1	0	1
0	1	1	0		1	1	0	2
0	1	2	0		2	1	0	0
0	2	0	0		0	2	0	2
0	2	1	0		1	2	0	0
0	2	2	0		2	2	0	1
1	0	0	0		0	1	0	1
1	0	1	0		1	1	0	2
1	0	2	0		2	1	0	0
2	0	0	0		0	2	0	2
2	0	1	0		1	2	0	0
2	0	2	0		2	2	0	1
1	1	0	0		1	2	1	0
1	2	0	0		2	0	1	0
1	1	1	0		2	2	2	0
1	1	2	0		0	2	2	1
1	2	1	0		0	0	1	1
1	2	2	0		1	0	1	2
2	1	0	0		2	0	1	0
2	1	1	0		0	0	1	1
2	1	2	0		1	0	1	2
2	2	0	0		1	1	2	1
2	2	1	0		2	1	2	2
2	2	2	0		0	1	2	0

(a)

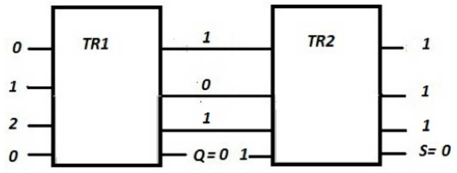
Input					Output			
D	E	F	R		U	V	W	S
0	0	0	1		0	0	0	1
1	0	0	1		1	1	0	2
2	0	0	1		2	2	0	0
0	1	0	1		0	1	0	2
1	1	0	1		1	2	0	1
2	1	0	1		2	0	0	1
0	2	0	1		0	2	0	0
1	2	0	1		1	0	0	1
2	2	0	1		2	1	0	2
0	1	0	1		0	1	0	2
1	1	0	1		1	2	0	0
2	1	0	1		2	0	0	1
0	2	0	1		0	2	0	0
1	2	0	1		1	0	0	1
2	2	0	1		2	1	0	2
1	2	1	1		1	0	1	1
2	0	1	1		2	2	1	1
2	2	2	1		2	1	2	1
0	2	2	1		0	2	2	2
0	0	1	1		0	0	1	2
1	0	1	1		1	1	1	0
2	0	1	1		2	2	1	1
0	0	1	1		0	0	1	2
1	0	1	1		1	1	1	0
1	1	2	1		1	2	2	2
2	1	2	1		2	0	2	0
0	1	2	1		0	1	2	1

(b)

e) Example: Let $A = 0$, $B = 1$ and $C = 2$. The outputs of TR1 would be $L = AB \oplus C = 2$, $M = A \oplus B = 1$, $N = 2AB = 0$ and $Q = P \oplus A \oplus B \oplus C = 0$. Since TR2 receives the outputs L, M, N of TR1 as its inputs *i.e.* $D = L = 2$, $E = M = 1$, $F = N = 0$ and, the output S would be $S = R \oplus D \oplus E \oplus F = 1$. Therefore, S is the successor of Q since it is assumed there is no change in the inputs anywhere in the middle of operation. Let's say that, in the middle of operation the TR1 input A is changed to 2 due to an error. Input values for computing L, M, N will be changed to $A = 2$, $B = 1$ and $C = 2$ whereas for Q, it is still $A = 0$, $B = 1$ and $C = 2$. Therefore, the outputs of TR1 become $L = 1$, $M = 0$, $N = 1$ and $Q = 0$. Since TR2 will take the flawed L, M, and N outputs as its inputs, it will generate $S = 0$ which violates the condition of the error free circuit $S = \overrightarrow{Q}$ since here S is not a successor of Q. Hence the presence of a fault is assumed. The inputs and outputs for the testable block in both cases is shown in Figure 9.



(a)



(b)

Fig. 9. (a) Values on the TR1 and TR2 lines during fault free operation. (b) Values on the TR1 and TR2 lines during faulty operation, *i.e.* in the presence of an error.

f) Single bit error detection policy: Any single bit error in the input combination must generate a result of different pattern in outputs from the pattern than would be generated for the combination of inputs before the error occurs. The result/output (Q) for the single bit error can never be the same as the result/output (Q) for the error-free inputs, since otherwise the fault cannot be detected. This condition is the core of the single bit error detection policy.

From the above truth tables we can see that in the patterns listed a single bit change in any input pattern in TR1 will result into a different pattern which in turn generates a new result in Q that is different from the original one. For example, let us set inputs $A = 1$, $B = 1$ and $C = 1$. The output is $Q = 0$ for this combination. If a single bit error changes A to 2 then the new combination is $A = 2$, $B = 1$ and $C = 1$. For this input combination $Q = 1$ which is different from the original output $Q = 0$. This feature is important to generate an output on S in TR2 that is not a successor of Q to detect a flaw.

It is worth noting that in [11] where the same principle has been followed to detect a single bit error in the logical block, it appears that the above-mentioned feature was not maintained for some combinations. This results in the failure to detect faults for those input combinations.

IV. Internal Designs

A. TR1

Basic Toffoli and Feynman gates are used to design the internal block of TR1. 2 and 4 input Feynman gates are used to implement the blocks. The 2 input Feynman gate was discussed in [16]. The notation/block diagram used in the circuit design and realization using M-S gates is shown in Figure 10. The TR1 block is implemented using the Feynman gates illustrated in Figure 10 and basic 3-input Toffoli gates. The internal architecture of the TR1 block is shown in Figure 11.

B. TR2

Three 2-input Feynman gates are used to implement the TR2 block as shown in Figure 12.

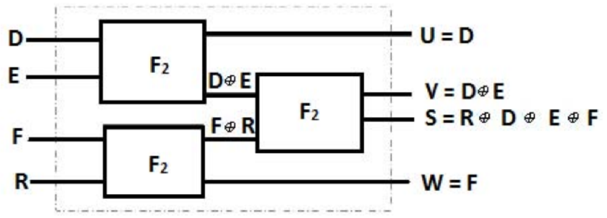


Fig. 12. Internal diagram of the TR2 Block.

V. Two Pair Two rail Checker Circuit

As shown in [17], error checking and correcting is often implemented using one of two main techniques: parity codes and two rail checkers. Two rail checkers compare the outputs from more than one identical system. This process is also used to reduce the error detector output. The two rail checker receives a predetermined pattern of inputs as a subset of the entire input set during fault free operation and generates a predetermined output to represent the fault free situation. A two pair two rail checker receives two pairs of inputs from two identical systems and detects if any flaw exists in those systems [17]. This concept has also been used in the reversible context by other authors such as in [18] and [11].

We have also designed a two pair two rail checker circuit to detect flaws in a large circuit by cascading our TR block and rail checker circuit. The rail checker circuit is designed using ternary 1-qudit permutative gates and ternary controlled-controlled gates. These gates are discussed in the following sections.

A. 1-qutrit Permutative Gate

1-qutrit unitary permutative gates are discussed in [7] and [16]. These works state that if any transformation of the qutrit state can be represented by a 3*3 unitary matrix, then a 1-qudit ternary gate can be specified in a similar way [7]. This transformation, known as the Z-transformation, shifts/permutes the input states to produce the desired output states. For example, the Z(+1) transformation shifts the input qutrit states by 1. Table III shows the truth table of 1-qutrit permutative gates and the symbol for the ternary 1-qutrit permutative gate is shown in Figure 13(a) [7].

TABLE III
Operations of 1-qutrit permutative gates as shown in [7].

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

Two 1-qutrit ternary gates act as another 1-qutrit ternary gate if they are cascaded together [7]. The cascading gates have a serial effect on the input which produces a resultant output that is similar to the single output of another 1-qutrit gate.

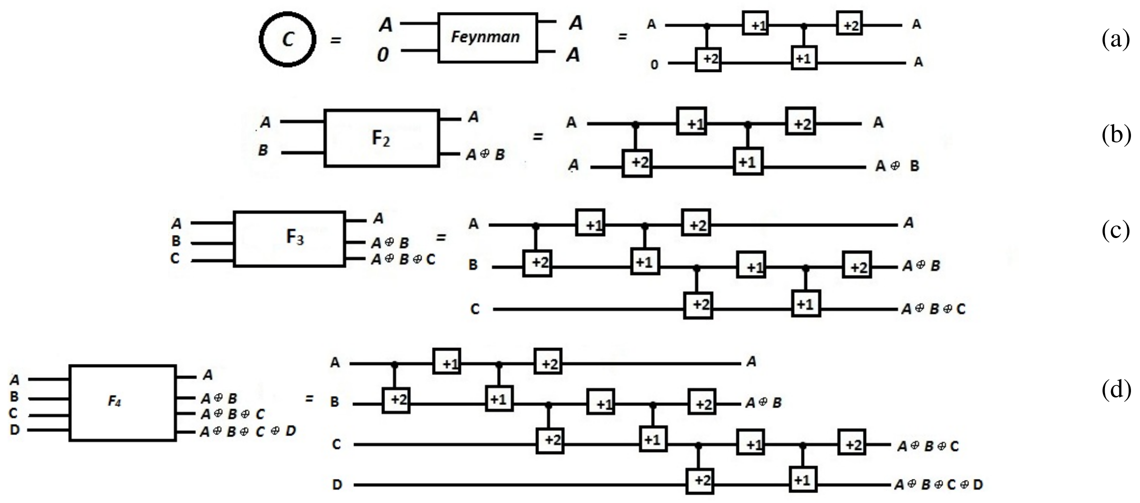


Fig. 10. (a) 2-input copy Feynman gate. (b) 2-input basic Feynman gate as shown in [16]. (c) 3-input Feynman gate. (d) 4-input Feynman gate.

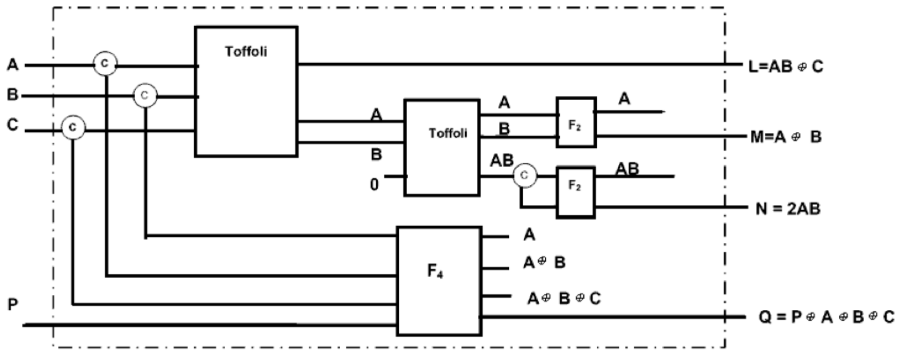
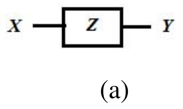


Fig. 11. Internal diagram of the TR1 block.

Table IV shows the resultant 1-qutrit gates for two cascaded 1-qutrit gates as shown in [7].

TABLE IV
RESULTANT 1-QUTRIT GATES FOR TWO CASCADED 1-QUTRIT GATES AS SHOWN IN [7].

	Second 1-qutrit gate				
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



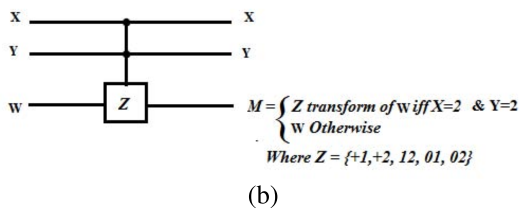

X
Y
W
Z
M = { Z transform of W iff X=2 & Y=2
W Otherwise
Where Z = {+1,+2, 12, 01, 02}
(b)

Fig. 13. (a) Symbol of 1-qutrit permutative gate. (b) Ternary controlled-controlled gate as shown in [7].

B. Ternary Controlled-controlled Gate

The ternary controlled-controlled gate was also proposed in [7]. The ternary controlled-controlled gate has two controlling inputs and one controlled output. Figure 13(b) illustrates the symbol of a ternary controlled-controlled gate. From the figure it can be seen that the Z-transform is applied on the controlled input W only when the values of both of the inputs X and Y are 2, otherwise W is passed unchanged.

C. Principle of the Rail Checker Circuit

The purpose of the rail checker is to detect if there is any flaw in the circuits connected to the rail checker by checking whether one output is the successor of the other. The rail checker circuit also generates two output values where one is successor to another if there is no flaw in the input circuits. Since these values may be cascaded to additional rail checkers, a successor is generated to represent the fault free situation. Otherwise, if the rail checker detects any flaw, the values generated can never be successors. Figure 14 shows the block diagram of a general rail checker circuit and Figure 17 shows the internal design of the rail checker circuit. From Figure 17 one can see that since error signals $X3$ and $X4$ are generated out of two physically separated modules E1 and E2, any single bit error in internal lines will affect either of the two outputs, but not both. Hence, any single error in the internal lines results in the outputs being non-successors and thus indicates the occurrence of an error.

The rail checker circuit is designed in such a way that if the inputs are successors of each other, *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 the combinations where $X4 \neq \overrightarrow{X3}$.

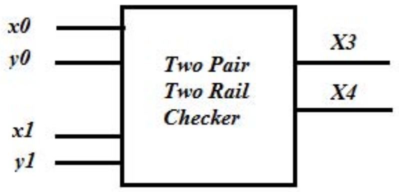


Fig. 14. Block diagram of two pair two rail checker.

1) Example: Let us say $x0 = 1$, $y0 = 2$, $x1 = 0$ and $y1 = 1$. Then the two pair two rail checker will produce $X3 = 1$ and $X4 = 2$ as its output. Again let us assume $x0 = 1$, $y0 = 2$, $x1 = 2$ and $y1 = 1$. then the two pair two rail checker will produce $X3 = 1$ and $X4 = 0$ as its output. Since in this case $y1 \neq \overrightarrow{x1}$, the rail checker generates $X4 \neq \overrightarrow{X3}$.

D. Elementary E gate

To implement the operation of the two pair two rail checker circuit discussed above, an elementary gate (E) has been designed which has two controlling inputs and a controlled output. The E gate is designed based on the architecture of the 1-qutrit ternary comparator circuit proposed in [7]. The controlling inputs (x, y) are the two of the four inputs of the main rail checker circuit and the controlled output (K) is one of the two outputs of the main rail checker circuit. Two such gates/blocks are used to create the rail checker circuit. The output of the controlled input depends on whether the second input (y) is the successor of the first input (x) or not. If it is a successor, a Z-transform operation changes the controlled input constant (0) to either 1 or 2 in y depending on which output the gate/block produces, $X3$ or $X4$, hence $K = 1$ or 2. Otherwise, 0 is passed unchanged *i.e* $K = 0$. Figure 15 shows the block diagram of the E gate.

1-qutrit permutative gates and ternary controlled-controlled gates are used to design our E gate. Table V shows all possible input combinations, the desired output for that combination, and shifts required to produce that output. From Table V it can be seen that only for three input combinations, the output K is the Z-transform of input constant 0 where Z

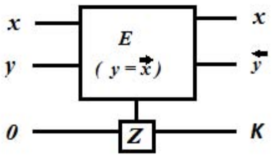


Fig. 15. E gate.

TABLE V
TRUTH TABLE AND TRANSFORMATION TABLE OF THE E GATE.

Input	Output	Shift Required
00	0	
01	1 or 2	+1,+2
02	0	
10	0	
11	0	
12	1 or 2	+1,+2
20	1 or 2	+1,+2
21	0	
22	0	

=+1,+2. For all other combinations $K = 0$. For the input combinations for which the output is 0, no transformation is applied on the 0-input and 0 is passed unchanged to the output. For the input combinations for which the output is the Z-transform of 0, the inputs are changed to 2 by applying the 1-qutrit transform (Z(+1), Z(+2)). The values are used to trigger the Z-transformation (Z(+1), Z(+2)) of the ternary controlled-controlled gate on input constant 0 to produce $K = 1$ or $K = 2$ at the output. Figure 16 shows our realization of this gate.

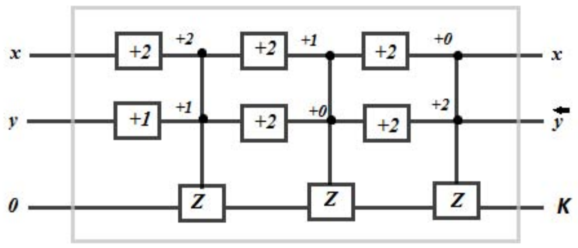


Fig. 16. Internal structure of the E gate.

For example, if the input combination is $x = 0$ and $y = 1$, transformation +2 and +1 is needed to transform/shift both the input values to 2, 2. Hence a +2 gate is placed along the x input and a +1 gate is placed along the y input. The outputs of these gates are used to trigger the Z-transform gates placed along input constant 0. Other transformations are applied by placing appropriate gates along x and y and the input constant 0 in a similar way. Since the 1-qutrit gates along input x and y are cascaded, effective transformations are shown explicitly at the controlling points.

E. Architecture of the Rail Checker Circuit

To construct the rail checker circuit, the first E gate that takes $x0$ and $y0$ as its inputs generates $K = 1$ at the output by applying Z(+1) transformation on constant input 0, but only if $y0 = \overrightarrow{x0}$. It generates $K = 0$ in all other cases.

The second E gate that takes $x1$ and $y1$ as inputs generates $K = 2$ at the output by applying Z(+2) on the 0 input, but only if $y1 = \overrightarrow{x1}$. It generates $K = 0$ in all other cases. Figure 17 shows the block diagram of the internal architecture of the rail checker circuit.

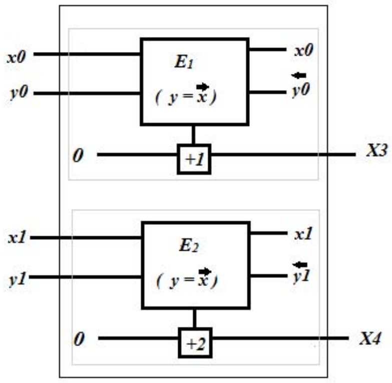


Fig. 17. Internal architecture of the two pair two rail checker circuit.

The operation of the rail checker can be easily verified from all possible input combinations of the blocks. Let us define the input state of a E block to be "True" if the inputs of the block are successors of each other and "False" otherwise. Let us also name the block of E gate with output $X3$ to be B1 and the block with output $X4$ to be B2. Table VI shows the truth table for all possible input states and the corresponding outputs for blocks B1 and B2.

TABLE VI

TRUTH TABLE FOR THE POSSIBLE INPUT STATES AND THE CORRESPONDING OUTPUTS OF BLOCK B1 AND B2

B1	B2	X3	X4
True	True	1	2
True	False	1	0
False	True	0	2
False	False	0	0

Table VI demonstrates that the rail checker circuit produces $X3 = 1$ and $X4 = 2$, (*i.e.* $X4 = \overrightarrow{X3}$) at the output only when both of the input sets have successive ($y0 = \overrightarrow{x0}$ and $y1 = \overrightarrow{x1}$) values.

VI. Sample Design

In [11] the author implemented the NAND-NAND form of the Sum of Product (SOP) expression $F = ab + cd$ using the proposed testable blocks. In ternary a Galois Field Sum of Product (GFSOP) can be directly implemented in a similar way by a ternary reversible circuit. Thus as an example we will implement the same function, *i.e.* $F = ab + cd$ as a demonstrate that the proposed blocks in this paper can successfully implement a GFSOP expression. Figure 18 shows the implementation of F using the proposed online testable ternary reversible blocks. Use of variables more than once can be implemented by duplicating variables using duplicating ciruits which can also be implemented by the proposed TR block. The final output of the second rail checker can be used if the function needed to be further extended; *i.e.* if additional logic were to be added.

VII. Conclusion

The testable blocks proposed in this paper use basic ternary building blocks to incorporate online testing features. This has produced complex and larger circuit blocks, although gates and garbage values were of course minimized where ever possible. To keep the number of garbage values at a minimum, the garbage output of one ternary gate is used as an input to subsequent gates. Because of the fan-out limitation of reversible logic circuits, Feynman gates are used to create copies. As far as we are aware our proposed TR and dual rail checker circuit are the first such combination to be proposed for ternary reversible logic. Work is continuing in a number of areas, including improved implementations for the TR1 and TR2 blocks, determining if they can be implemented as ternary gates, and reduction of the number of garbage outputs.

References

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

[2] I. Polian, T. Fiehn, B. Becker, and J. P. Hayes, "A family of logical fault models for reversible circuits," in *Asian Test Symposium*. Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 422–427.

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

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

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

[6] D. Miller and M. Thornton, *Multiple Valued Logic: Concepts and Representations*. The Morgan and Claypool Publishers, 2008.

[7] M. Khan, "Design of reversible/quantum ternary comparator circuits," *Engineering Letters*, vol. 16:2, pp. 178–184, May 2008.

[8] M. H. A. Khan, "Quantum realization of ternary toffoli gate," in *Proceedings of the 3rd International Conference on Electrical and Computer Engineering*, 28-30 December, 2004.

[9] M. H. A. Khan and M. A. Perkowski, "Quantum ternary parallel adder/subtractor with partially-look-ahead carry," *Journal of Systems Architecture*, vol. 53, pp. 453–464, 2007.

[10] N. Jha and S. Gupta, *Testing of Digital Systems*. The Press Syndicate of the University of Cambridge, 2003.

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

[12] B. W. Johnson, *Design and Analysis of Fault-tolerant Digital Systems*. Prentice-Hall International, 1985.

[13] P. K. Lala, *Fault-Tolerant and Fault Testable Hardware Design*. Prentice-Hall International, 2003, London, UK.

[14] 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, January 2010.

[15] D. Vasudevan, P. K. Lala, J. Di, and J. P. Perkerson, "Online testable reversible logic circuit design using NAND blocks," in *Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04)*, 2004, pp. 325–330.

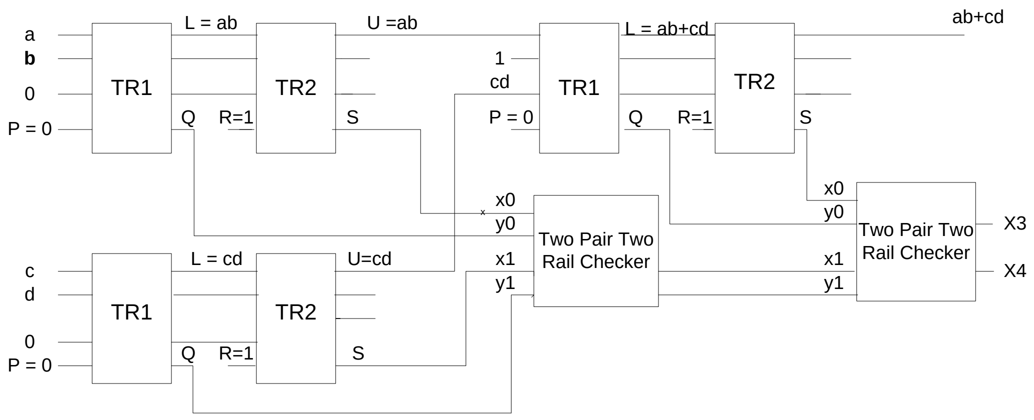


Fig. 18. Online testable ternary reversible implementation of function $F = ab + cd$.

[16] M. H. A. Khan, M. A. Perkowski, and M. R. Khan, "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*, 2004, pp. 58–67, Toronto, Canada, 19-22 May.

[17] D. Nikolos, "Self-testing embedded two-rail checkers," *Journal of Electronic Testing: Theory and Applications*, vol. 12, no. 1–2, pp. 69–79, Feb./April 1998.

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