On Designing a Ternary Reversible Circuit for Online Testability

Md. R. Rahman and J. E. Rice
Dept. of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
Email: {j.rice; md.rahman7}@uleth.ca

Abstract

Reversible logic has the potential to solve the problems of energy efficiency that are beginning to create roadblocks in the continued advancement in complexity and decrease in component size of today's computer systems. Multiple-valued versions of reversible logic can provide even further advantages, but there is currently in the literature little to no work on testability of such designs. This paper details work on designing an online testable block for ternary reversible logic. We build on earlier work that introduced the basic design, and provide some improvements and modifications. The proposed testable block can be used to implement most ternary logic operations and is capable of testing the reversible ternary network in real time (online). Since all parts of the testable block are constructed of reversible building blocks, the block is itself reversible and thus multiple such blocks can be combined to construct complete, testable ternary reversible circuits.

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

1 Introduction

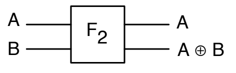
The traditional model for today's circuits is *irreversible*; that is, it is not possible to unambiguously reconstruct the inputs given a particular output. The implication behind this is that information is lost, and we observe that in the heat dissipation that must be managed. At the current rate of technological advancement, this problem may soon cause us to hit a roadblock in our increasing reduction in the size of components, as well as the simultaneous increase in complexity. In order to advance beyond this point, energy-recovering techniques will be required and eventually require an increasing amount of reversible logic in digital design. Reversible computing recovers energy by conserving information when performing logic, storage, and communication operations using reversible transformations [5]. This has been known since the 60s, when Landauer [14] and later Bennett [1] both proved that reversible logic would be necessary for lower power dissipation in circuits. More recently researchers have applied these theories to areas such as supercomputing [2]. While many of us are familiar with Boolean (binary, two-valued) logic, both in traditional (irreversible) circuit design and reversible models, there are many advantages to moving to a multiple-valued paradigm, or at least considering problems using a less restricted model. Many existing problems have simpler solutions when approached using a p-valued ($p > 2$) model [16], and existing work has shown a variety of ways to implement and model ternary reversible logic [3, 8, 9]. With all these reasons to investigate reversible multiple-valued logic design, many researchers are developing logic synthesis and optimization techniques; however there is much less work on fault detection. As anyone in this industry is aware, fault-tolerance and testing of computer systems and their components is absolutely essential. Works such as [13, 18] and [22] are beginning to address this area, however the areas of modeling faults of ternary reversible circuits and real-time online fault detection have yet to be addressed, despite the clear significance of these areas.

In this work we build on the ideas first introduced in [19]. We discuss the design of ternary reversible logic blocks that can be combined to implement most basic ternary logic functions, and have the ability to test the functions while the circuit is under operation (online testing). We present details of the designs and use in Section 3, and discuss their internal architectures in Section 4. A two-pair two-rail checker is used to implement a cascade of such testable blocks; design of this is briefly discussed in Section 4.1.

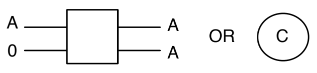
2 Background

2.1 Ternary Reversible Logic

The ternary reversible blocks in this paper are built using ternary Toffoli, Feynman, and Muthukrishnan-Stroud (M-S) gates. Quantum realizations for the ternary Toffoli and Feynman gates using M-S Gates are discussed in [10], as is background for the M-S gate. Figure 1 illustrates the basic

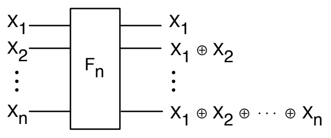


(a) General operation of the two input Feynman gate.

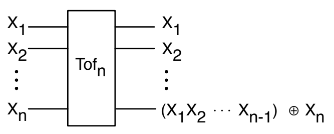


(b) The copy gate.

Figure 1. The two input ternary Feynman gate.



(a) The n input Feynman gate.



(b) The n input Toffoli gate.

Figure 2. The general formats of the n input ternary Toffoli and Feynman gates.

two input ternary Feynman gates. The reader will note that the copy gate, sometimes denoted by a C inside a circle, is simply the two input Feynman gate with the bottom input set to 0. The Feynman gate can be extended to n inputs, as shown in Figure 2. Another popular reversible gate is the Toffoli gate, which is also illustrated in Figure 2. For further details we refer the reader to [10] for an excellent overview of the notation used in this area as well as further background and explanation.

2.2 Fault Models

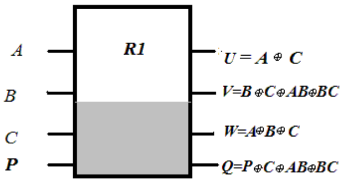
Recognizing and modeling the behaviour of possible defects in a circuit is known as fault modeling. [6] discusses how different levels of abstraction in circuit design can result in different fault models. Our technique builds on the method proposed in [21], which characterizes the type of fault that 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 propagates the result to the outputs. This type of fault falls best into Niraj and Gupta's functional fault model category: models that focus on the faults in functional blocks of the system, with the goal of ensuring fault-free behaviour of the blocks. The reader is directed to [6] for a discussion of fault models, and [22] for a beginning in the discussion of fault models as related to reversible logic.

2.3 Online Testing

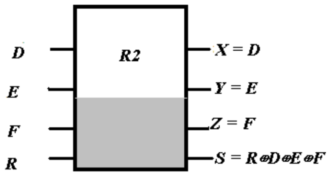
Online testability is the ability of a circuit to test a portion of the circuit while the circuit is operating [21]. Detecting a fault in operation, the point of occurrence and in some cases, attempt to recover from fault are the major focus of research into online testing [6]. Similar processes for binary reversible logic have been discussed in [6] and [7]. In traditional circuit design offline testability is often implemented in built-in self-test (BIST) structures; however we are not aware of any investigations towards this for reversible logic. Niraj and Gupta provide an overview of BIST in [6] should the reader be interested in further details.

2.4 Related Work

In [21] the authors propose online testable logic blocks for binary reversible logic. Two gates designated R1 and R2 are used in pairs to implement testable portions of reversible circuits, which are then cascaded together so that the entire circuit is checked. Figure 3 illustrates their design. The



(a) R1 gate.

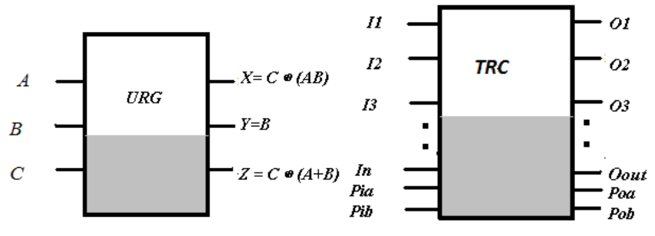


(b) R2 gate.

Figure 3. The two gates proposed in [21] for online testing of reversible circuits.

R1 gate is designed to implement arbitrary Boolean functions; for instance, the OR operation can be implemented by setting the inputs $A = P = 0$ which produces output $W = B \oplus C$ or output $U = A \oplus C$ can be directly used. The R2 gate 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. The basic idea is that the R1 gate is used to implement binary functions while the R2 gate is used to detect faults. [21] 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 [21].

In another related work, authors of [15] propose a universal reversible logic gate that can be used to construct online testable circuits. A principle similar to that used in [21] and [20] to detect the faults in logic blocks is also used in this approach. The gates used in this approach are shown in Figure 4. We propose using a similar principle to design online testable ternary reversible logic blocks.



(a) (b)

Figure 4. The gates proposed in [15] for online testing of reversible circuits.

3 Online Testable Ternary Reversible Logic Block

We next describe the design for an online testable ternary reversible logic block. The basic concept is similar to that described in [21], but we have extended this work to the ternary case and as well corrected some problems. The online testable ternary reversible logic block, or TR, is composed of two individual blocks as was first described in [19]. In this section we begin with the blocks originally proposed and then go on to describe the modifications that have recently been made.

3.1 The TR1 Block

The TR1 block is the ternary reversible testable block that is used to implement the logic needed for the functionality of the circuit. 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 5. This block is used to implement the five basic func-

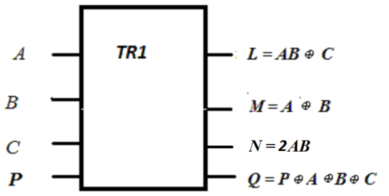


Figure 5. Ternary TR1 gate.

tions: AND, OR (EXOR), successor, negation/complement and mod-difference. The unary operation called successor ($\overrightarrow{x}$) is defined as $(x+1)mod_p$ [16] where x is the input and p is the cardinality of the logic value (in this case, 3). As an example, let $x = 1$ and assume we wish to find $\overrightarrow{x}$. Setting inputs $A = x = 1, B = 1, C = 0, P = 0$ in TR1 produces the desired result, $M = A \oplus B = x \oplus 1 = 2 = \overrightarrow{x}$.

3.2 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 $O = (U = D, V = E, W = F, S = R \oplus D \oplus E \oplus F)$, where I and O are input and output sets respectively. Outputs U, V and W are the copies of inputs D, E and F, which provide these values for further use in the circuit if necessary. Figure 6 shows the block diagram of the TR2 block (shown as a sub-component of the encompassing TR block). Output S is used to detect any single bit error when TR2 is cascaded with a TR1 block to form an online testable circuit.

3.3 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 so that $P = \overrightarrow{R}$. For normal operation we simply set $P = 0$ and $R = 1$. Figure 6 shows the composition of the TR block. TR1 takes ternary logic values at A,

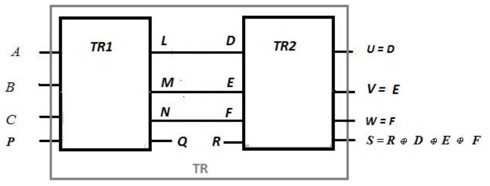


Figure 6. Configuration of the online testable ternary reversible block TR.

B, C as its inputs and we set $P = 0$. The inputs A, B and C are a set depending on the required operation as discussed in Section 3.1. 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 or in TR1.

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 blocks remain unchanged.

For 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 in 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.

3.4 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 that would be generated for the combination of inputs before the error occurs. The result or output (Q) for the single bit error must never be the same as the result for the error-free situation, as otherwise the fault cannot be detected. This condition is the core of the single bit error detection policy. [19] elaborates further on how the TR block operates in accordance with this policy. This principle is the one that is suggested in [21], however in their work it is possible to determine certain input combinations that differ by a single bit and yet do not result in a change in the output. This could result in the failure to detect faults for those input combinations. We have developed our TR block design such that there are no input combinations that differ by a single bit but give the same output.

4 Internal Designs

The TR1 block is built from basic Toffoli and Feynman gates, and three 2-input Feynman gates are used to implement the TR2 block. Figure 7 shows the TR1 block while Figure 8 illustrates the TR2 block.

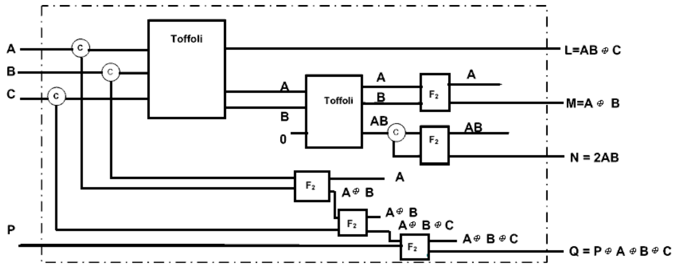


Figure 7. Internal diagram of the TR1 block.

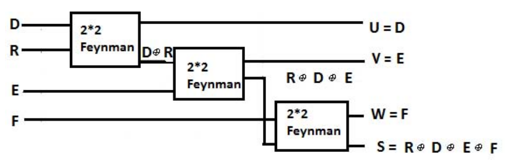
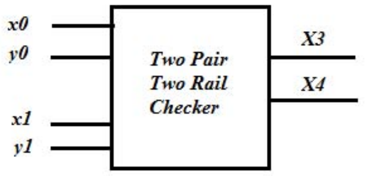


Figure 8. Internal diagram of the TR2 block.

4.1 Two Pair Two Rail Checker Circuit

Error checking and correcting is often implemented using one of two main techniques: parity codes and two rail checkers [17]. Two pair rail checkers compare the outputs from more than one identical system. We use a ternary reversible two pair two rail circuit in this work to cascade TR blocks in order to build an entire testable circuit. This concept has also been used in the reversible context by other authors such as in [4] and [21]. Our rail checker circuit is designed using ternary 1-qudit permutative gates and ternary controlled-controlled gates. Figure 9a shows the rail checker block diagram, and full details are given in [19]. 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 $X4 \neq \overrightarrow{X3}$. For example, let us say $x0 = 1$,



(a) Block diagram of two pair two rail checker.

B0	B1	X3	X4
True	True	1	2
True	False	1	0
False	True	0	2
False	False	0	0

(b) Truth table for the possible input states and the corresponding outputs of block B0 and B1.

Figure 9. Two pair two rail checker.

$y0 = 2$, $x1 = 0$ 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 $X4 \neq \overrightarrow{X3}$.

The internal architecture of the rail checker consists of an upper block to compare $x0$ and $y0$ and a lower block to compare $x1$ to $y1$. We label the upper block B0 and the lower block B1. The table in Figure 9b 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. The table uses the value True if the inputs of a block are successors of each other, and False otherwise.

5 Sample Design

In [21] 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) expression 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.* $F = ab + cd$, to demonstrate that the proposed blocks in this paper can successfully implement a GFSOP expression. Figure 10 shows the implementation of F using the proposed online testable ternary reversible blocks. Use of

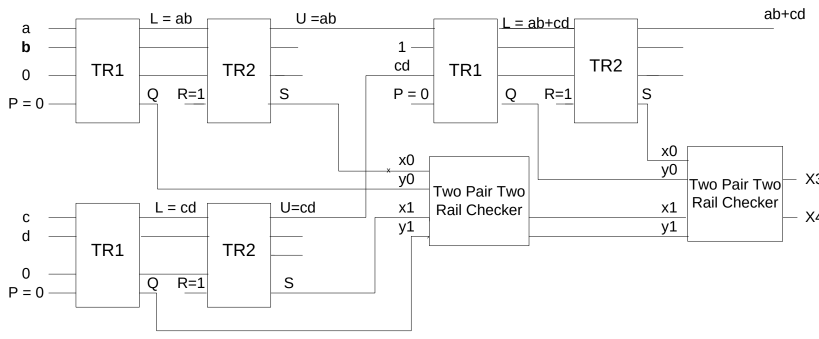


Figure 10. Online testable ternary reversible implementation of function $F = ab + cd$.

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 extended; *i.e.* if additional logic were to be added.

6 Modifications

This work was introduced as preliminary work in [19]. Since that time small modifications to the designs have been made. In particular the TR1 and TR2 blocks have modified outputs to allow for more efficient and flexible implementation of circuits using the testable blocks, and the internal designs of each of these gates have been improved. Here we provide some discussion of the modifications to the internal designs.

Both ternary Toffoli and Feynman gates can be implemented either using Generalized Ternary Gates (GTG) or M-S gates. GTGs are discussed in [12] and [9]. According to these works, GTGs provide an easy quantum realization using technologies such as ion-trap. However the design of a single Toffoli gate requires at least 8 GTGs [9]. Our design for the TR1 block requires two Toffoli gates and 9 Feynman gates, resulting in 25 GTGs. The TR2 block requires 3 Feynman gates, and therefore requires 3 GTGs. In total, 28 GTGs are needed. The count can be further reduced to 26 by replacing the two Feynman gates used to generate 2AB in Figure 7 by a 1*1 dual-shift-gate, as proposed in [11]. This is a very simple gate that implements $x \rightarrow 2x$.

A smaller solution can be achieved by using 3-qutrit generalized Toffoli gates, as proposed in [10], and M-S gates to implement the Feynman gates. 3-qutrit generalized Toffoli gates can be used to realize GFSOP minterms of an arbitrary ternary function. The total number of M-S gates required to construct the TR1 block can be easily calculated. There are two 3*3 Toffoli gates, each requiring 28 M-S gates and nine 2*2 Feynman gates, each requiring 4 M-S gates [10]. This results in a total of 92 M-S gates. The output A is not considered as garbage since it is a primary input of the block. Hence there are only four garbage outputs.[1]

Further improvements can be achieved by reusing the A, B, and AB outputs of the Toffoli gates to generate $A \oplus B$, $2AB$ and $AB \oplus C$. Output AB must be copied by a Feynman gate because of the fan out limitation. One of the copies is EXORed with C to generate $AB \oplus C$ and the other copy is passed through a dual shift gate to generate 2AB. We can further reduce the cost by excluding the copy gates and reorganizing the Feynman gates. Since we are using only one Toffoli gate and use mostly Feynman gates and one dual shift gate, the cost will be dramatically reduced. Figure 11 shows the new design of the TR1 block. Figure 12 shows the realization of the TR1 block using a

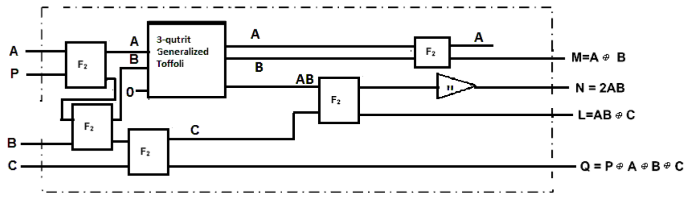


Figure 11. Upgraded design of TR1.

generalized Tofolli gate, five Feynman gates and one dual shift gate. The cost to realize this design is 28 M-S gates to build the Toffoli gate, 4 M-S gates for each Feynman gate, and one M-S gate for the dual shift gate, for a total of 49 M-S gates. Therefore the cost is reduced by 55.4% in the new design. The most significant achievement, however, is the fact that we have reduced the number of garbage lines to 0. This is a remarkable improvement, as nearly all reversible implementations require garbage lines.

7 Conclusion

The testable blocks described in this paper use basic ternary building blocks to incorporate online testing features for ternary reversible circuits. We describe two gates

[1]According to [12] the quantum cost of a M-S gate is one, so we use the count of M-S gates for comparison of the original and improved designs.

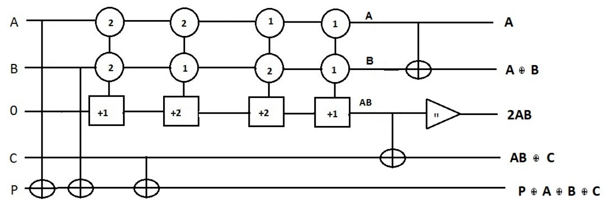


Figure 12. Realization of upgraded TR1 block

to be used in conjunction; the first gate is used to implement the desired functionality, while the second incorporates online testing to check for correct functionality at all times. These two gates can be used to build small parts of a ternary reversible circuit, and then these parts can be cascaded together using a dual-rail circuit (described in [19]) so that the entire circuit will be checked. The underlying concepts are based on that introduced in [21], although their work is strictly binary while we have extended it to the ternary case. As far as we are aware this is the only online testable design currently in the literature for ternary reversible logic. These 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 gates, analysis of overhead requirements and the development of synthesis approaches utilizing our gates.

Acknowledgment

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

References

[1] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 6:525–532, 1973.

[2] E. P. DeBenedictis. Reversible logic for supercomputing. In *Proceedings of the 2nd ACM Conference on Computing Frontiers*, pages 391–402, 2005. May 4–6, Ischia, Italy, ACM Press.

[3] G. Epstein. A summary of investigation into three and four valued logic. In *Proceedings of the 8th International Symposium on Multiple-Valued Logic*, page 257, 1978. Rosemont, Illinois, United States.

[4] 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)*, pages 204–205, 2010. Corfu, 5-7 July.

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

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

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

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

[9] 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.

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

[11] M. H. A. Khan, M. A. 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 (ISMVL)*, pages 146–153, 2003. M 16–19, Tokyo.

[12] 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*, pages 58–67, 2004. Toronto, Canada, 19-22 May.

[13] 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)*, pages 277 – 280, 2010. Vienna, 14–16 April.

[14] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–191, 1961.

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

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

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

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

[19] M. R. Rahman and J. E. Rice. Online testable ternary reversible circuit. In *Proceedings of the Reed-Muller Workshop*, 2011. accepted for presentation May 25–26, Tuusula, Finland.

[20] 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)*, pages 325–330, 2004.

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

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