Journal of Electronic Testing manuscript No.
(will be inserted by the editor)

Online Testable Approaches in Reversible Logic

N. M. Nayeem · J. E. Rice

Received: date / Accepted: date

Abstract We present an overview and analysis of existing work in the design of online testable reversible logic circuits, as well as propose new approaches for the design of such circuits. We explain how previously proposed approaches are unnecessarily high in overhead and in many cases do not provide adequate fault coverage. Proofs of the correctness of our approaches are provided, and discussions of the advantages and disadvantages of each design approach are given. Experimental results comparing our approaches to existing work are presented as well. Both approaches that we propose have better fault coverage and significantly lower overheads than previous approaches.

Keywords Reversible circuit · online testing · testable circuit · Toffoli gates

1 Introduction

Today's consumers are demanding more of their electronic devices while also wanting these devices to be smaller and smaller; in addition they also want their devices to use less and less power. While the industry has so far been able to meet these demands with a remarkable validation of Moore's law [24], some researchers are predicting that this will come to an abrupt end around the year 2015. Logic voltages can not be reduced much further than their current levels without comprompising efficiency and impacting power consumption; industry will be unable to reduce transistor sizes much further without running into

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

University of Lethbridge
4401 University Dr. W.
Lethbridge, AB, Canada
Tel.: +1 403 329 2783
Fax: +1 403 317 2882
E-mail: j.rice, noor.nayeem@uleth.ca

leakage problems, and the reduction of signal energies is only a temporary fix as below a certain threshold digital devices cannot reliably function due to thermal noise [11].

This is not only being identified by academics and researchers, but also by industry groups. The ITRS (International Technology Roadmap for Semiconductors), a group sponsored by semiconductor manufacturers around the world, produces a report every 2 years on the state of the industry and its future directions. In 2009 their report stated that "It is forecasted that by the end of the next decade it will be necessary to augment the capabilities of the CMOS process by introducing multiple new devices that will hopefully realize some properties beyond the ones of CMOS device." This was clarified later in the report as referring to the possibilities of using "particle spin to perform non-volatile memory and logic functions on a CMOS chip", or performing "an information processing function utilizing spin waves" [1]. Both of these, as the report later states, belong to the fields of quantum and reversible computing. The 2011 report provides additional evidence of an industry struggling to keep up with Moore's law through small improvements in existing technology. Statements such as this illustrate clearly the difficulties being encountered by our industry: "Cost-effective heat removal from packaged chips remains almost flat in the foreseeable future. Driven by the 2 times increase in transistor count per generation, *power management is now the primary issue across most application segments*" (emphasis added).

One solution may be the use of reversible logic in the design of circuits. Traditional logic loses information during computation, which is dissipated as heat [16]. In 1973 Bennett stated that "loss of information implies energy loss" [4] and proved that in order to *not* dissipate energy a circuit must be entirely composed of reversible gates; more recently Frank theorized that a reversible logic device could recover a fraction of energy that could reach up to 100%. Research on reversible computing has also shown its use in a variety of technologies such as quantum computing [28], low power CMOS design [7,8], optical computing [30], and nanotechnology [20].

Despite having been proposed in 1961, reversible logic has only recently gained popularity, no doubt driven by the looming power management issues. Prototypes of reversible devices and chips are beginning to appear, *e.g.* [14] and [40], and many options for synthesis of reversible circuits have been proposed including [10, 22, 37, 41, 43], to name just a few. Testing approaches are also beginning to develop, and existing work in this area includes [9,15,17,39]. We provide an overview of this work and analyse the issues related to these designs. We also present our own approaches for creating online testable reversible circuits based on work first presented in [27] and [25]. This paper provides further evidence of the fault coverage obtained by our approaches as well as experimental results and comparisons with the previous approaches. Both of our approaches are intended to build on an existing cascade of Toffoli gates, such as that generated by the technique in [10]. To generate the online testable circuit our approaches add a small number of gates and a single parity line such that any single bit fault is propagated both forward to the end of the

circuit as well as "down" to the parity line. Thus any single bit fault in the circuit can be detected by examining the value of a single line at the end of the circuit.

2 Background

2.1 Reversible Gates and Reversible Circuits

[22] provides the following definition:

Definition 1 An m-input, m-output, Boolean function $f(X)$, $X = \{x_1, x_2, ..., x_m\}$ is reversible if it maps each input assignment to a unique output assignment.

In other words, a reversible function has the same number of inputs as outputs and there is a one-to-one mapping between its input and output vectors. A reversible gate realizes a reversible function. Since traditional logic gates other than the NOT gate are not reversible, a different set of gates are used in reversible logic. In this work we focus only on the family of Toffoli gates.

An n-bit Toffoli gate is a reversible logic gate with n inputs and n outputs and that maps the input vector $[k_1, k_2, \ldots, k_{j-1}, k_j, k_{j+1}, \ldots, k_n]$ to the output vector $[k_1, k_2, \ldots, k_{j-1}, (k_1 k_2 \cdots k_{j-1} k_{j+1} \cdots k_n) \oplus k_j, k_{j+1}, \ldots, k_n]$ where the symbol $\oplus$ denotes the EXOR operation. That is, all *control* line values are passed through the gate unchanged, while the *target* line is inverted if all control line values are 1. The NOT gate is a special case of a Toffoli gate with $n = 1$ and no controls. The 2-bit (that is, $n = 2$) Toffoli gate is also known as the CNOT gate or Feynman gate. A NOT gate, a CNOT gate and an n-bit Toffoli gate are shown in Figures 1(a), 1(b) and 1(c).

A negative-control Toffoli gate maps the input vector $[k_1, k_2, \ldots, k_{j-1}, k_j, k_{j+1}, \ldots, k_n]$ to the output vector $[k_1, k_2, \ldots, k_{j-1}, (k_1 k_2 \cdots \overline{k}_i, \cdots k_{j-1} k_{j+1} \cdots k_n) \oplus k_j, k_{j+1}, \ldots, k_n]$ where k_i is a negative control and $i \in \{1, 2, \ldots, n\}, i \neq j$. This gate may have one or more negative controls; in that case, the value of the target line is inverted if all positive controls have the value 1 and all negative controls have the value 0. A 3-bit Toffoli gate with a single negative control in its first input is shown in Figure 1(d).

The extended Toffoli gate (ETG) is a multi-target Toffoli gate proposed in [6]. In our work we use an $(n+1)$-bit ETG with two target outputs o_n and o_{n+1} as shown in Figure 1(e). This gate maps the input vector $[k_1, k_2, \ldots, k_n, k_{n+1}]$ to the output vector $[o_1, o_2, \ldots, o_n, o_{n+1}]$, where $o_j = k_j$ for $j = 1, 2, \ldots, n-1$, $o_n = k_1 k_2 \cdots k_{n-1} \oplus k_n$, and $o_{n+1} = k_1 k_2 \cdots k_{n-1} \oplus k_{n+1}$. Thus there are $n-1$ control lines and two target lines. Like the regular Toffoli gate the targets of an ETG may be any lines; we use the n^{th} and $n+1^{st}$ lines in this definition strictly for the sake of simplicity. Like a negative-control Toffoli gate, an ETG may have negative controls. We note that in our diagrams we illustrate the two target lines with different symbols (the $\oplus$ and $\times$ symbols); this is strictly to differentiate between regular Toffoli gates and the ETGs added for testability

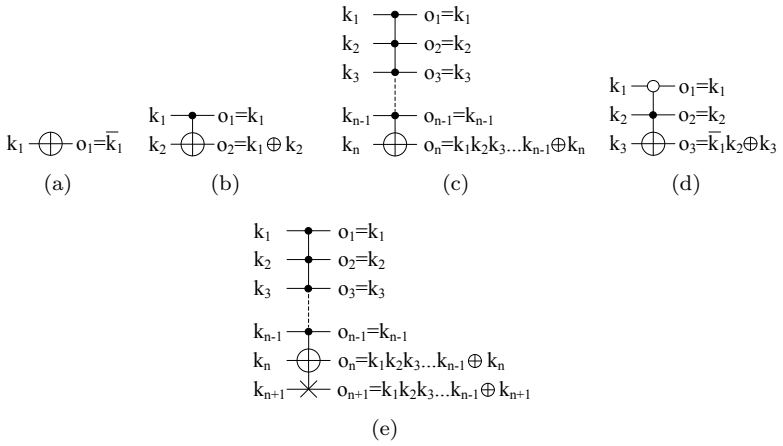

Fig. 1 (a) NOT gate. (b) CNOT gate. (c) n-bit Toffoli gate. (d) 3-bit negative-control Toffoli gate. (e) An $(n+1)$-bit ETG.

as we will describe later on in this paper. The two targets actually compute the same functionality, albeit on different lines.

A reversible circuit consists of only reversible gates which are interconnected without fan-out and feedback [28]. We refer to a reversible circuit as a Toffoli circuit if it is built using only Toffoli gates including NOT gates, CNOT gates and negative-control Toffoli gates.

2.2 Metrics

A given function can be synthesized several ways, each resulting in different reversible circuits. Thus cost metrics are required to evaluate and compare different circuits realizing the same function.

Gate count is a simple cost metric sometimes used to compare and evaluate different reversible circuits. It refers to the number of gates required to implement the circuit. It simply counts gates but does not take into account the complexity of the gates. As a result, it fails to provide meaningful information if the underlying implementation requirements of the gates under comparison are significantly different [23].

Quantum cost is a very popular measure used to compare reversible circuits. The quantum cost of a gate is the number of basic quantum operations needed to realize the gate [21]. Any reversible circuit can be decomposed into basic quantum gates, and the number of basic quantum gates required to implement the circuit is the quantum cost. Unlike gate count, quantum cost can provide accurate information since it considers the complexity of the circuit by counting the number of basic gates. In this work we calculate the quantum cost of a gate/circuit using the table given in [18] and information from [19] and [2]. For more information on this topic we direct the reader to [3].

In reversible circuits some outputs do not behave as final results but they are required to maintain the reversibility property. These unwanted outputs are known as garbage outputs. For some circuits, however, it is impossible to remove all the garbage outputs. The reader is directed to [45] for further discussion on the issue of garbage in reversible circuits.

In this paper we compare circuits in terms of their quantum cost and number of garbage outputs. We have computed the cost of an $(n+1)$-bit ETG to be 2+ the cost of an n-bit Toffoli gate since it can be simulated by an n-bit Toffoli gate and two CNOTs. Similarly, an $(n+1)$-bit negative-control ETG has the cost of 2+ the cost of a negative-control n-bit Toffoli gate.

2.3 Fault Models and Testing Approaches

There are several fault models currently proposed for use in reversible logic. These include traditional models such as the stuck-at model, but also newly proposed models such as the missing gate fault model (MGF) family and crosspoint fault model. The MGF model was first proposed in [12] and modeled the disappearance (or non-operation) of an entire gate. This is refined and extended in [31] to include the single missing gate fault (SMGF) in which a single gate is missing; the multiple missing gate fault (MMGF) in which multiple gates are missing; the repeated gate fault (RGF) in which gates may be repeated, and the partial missing gate fault (PMGF) in which all or some control points of a gate may be missing/non-operational. The crosspoint fault model is based on the crosspoint model for programmable logic arrays and [46] proposed two categories of crosspoint faults: disappearance faults, which occur when one or more control points of a Toffoli gate are missing, and appearance faults, when an additional control point is erroneously added. Disappearance faults in fact model the same fault as the PMGF model. A more generic model referred to as the bit fault model has been considered in various articles such as [39] and [17]. In this model a fault in a gate changes the behavior of its outputs, whether they are control lines or targets. A single bit fault is reflected on exactly one output of a gate, changing the correct value of the output to a faulty value. This model is very similar to the single stuck-at fault model, where a single stuck-at fault erroneously fixes the output of a gate to either 0 or 1. However a bit fault on a particular line inverts the value of that line, changing it from logic 0 to logic 1 or vice versa. Thus unlike the stuck-at fault model the single bit fault model depends on the input values. In this paper we consider primarily the single bit fault model.

Testing is required to ensure quality, availability, and reliability of a circuit or device. According to [42], testing can be performed online (concurrent testing) or offline (non-concurrent testing), or a combination of both. Online testing takes place while the system is performing its normal operation, allowing faults to be detected in real time. Offline testing requires the system or a part of the system to be taken out of operation to perform testing, and generally involves the application of a set of test vectors that will detect all

possible faults under a given fault model (a complete test set). This assumes, of course, that faults are permanent. In some cases a design for testability (DFT) approach will be used, requiring the addition of extra circuitry or redesign of the circuit in order to facilitate the testing process. Work in offline testing of reversible logic circuits includes [29] and [5] as well as DFT methods such as in [13] and [32]. If, however, we assume that faults may be transient, then an offline approach may not be desirable. In this case there is a strong argument for an online approach that is continuously testing for faults while the circuit is in operation. We suggest that for reversible circuits, where the underlying technology is still being theorised, such an approach is particularly desirable. For instance, [12] suggests that many potential technologies would involve static states which are modified by electromagnetic pulses (EM); these pulses then are modeled as gates, and their sequential application would form the "cascade" of gates in the circuit. As argued by [33] it is not unreasonable to assume that there would be variability in the EM pulses or in the maintenance of the static states.

In this work we focus on the bit fault model in an online testing approach. While [31] has suggested the MGF family, with corresponding arguments for each type of fault related to potential problems with EM pulse "gates", this again assumes that the pulses will perform in exactly the same way in an offline testing situation as when the circuit is under normal operations. In addition, while there appear to be good arguments for the MGF family and/or crosspoint fault models in application to the unknown quantum technologies in which reversible circuits are likely to be implemented, there is also an equally good argument for a generic fault model such as the bit fault model. We provide an example of this: if we assume that one or more permanent bit faults are present in a circuit such as shown in Figure 2, then it is necessary to apply only one test vector to detect each fault – a bit fault (shown with an X in Figure 2) on any of the lines will invert the value on that line. However if we

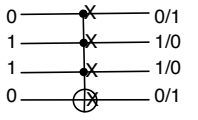

Fig. 2 A simple circuit with multiple permanent bit faults. The test vector of {0110} was chosen arbitrarily.

assume a transient fault that, like the cell fault model proposed in [29], can be triggered by certain input values, then all possible 2^k (for a gate with k lines) values must be applied in order to detect the fault, which would trivially also detect all other faults under the different fault models. It is this second possibility that we are targeting with the approach described in this work. We do not, however, propose the development of a test set; rather our proposed online approach is designed to detect faults *of any type* as they occur.

3 Review of Online Testable Approaches

We next provide an overview of a number of different approaches for generating online testable reversible circuits. We discuss the potential design issues and the benefits and limitations of each approach.

3.1 Testable Circuit Design Using R1, R2, and R Gates

Vasudevan *et al.* [39] proposed a design methodology for constructing online testable reversible circuits. Three new reversible gates R1, R2 and R were introduced as shown in Figure 3. The R1 gate is used to realize NAND, OR, EXOR, and EXNOR operations by setting different values on inputs. This gate has a parity output at q. The gate R2 passes the inputs through to the outputs, with a parity output again being computed at s, and in order to construct a testable block (TB), the gates R1 and R2 are cascaded by connecting the first three outputs of R1 to the first three inputs of R2, creating a testable block with two parity outputs that can be compared to determine whether a single bit fault has occurred or not. Figure 4 shows the construction of a TB and its block diagram. The TBs are then used to realize the reversible circuit.

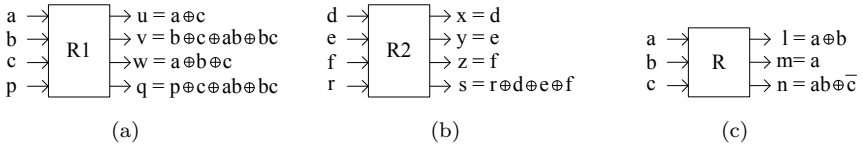

Fig. 3 (a) R1 gate, (b) R2 gate, and (c) R gate.

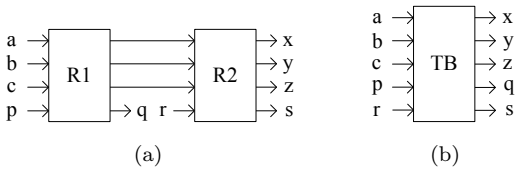

Fig. 4 (a) Construction of a testable block (TB), and (b) its block diagram.

Since each TB generates two parity outputs, a two-pair two-rail checker circuit is also required to test the parities of two TBs. The checker circuit takes (q_1, s_1) and (q_2, s_2), the parities of two TBs, as inputs and produces two outputs ($e_1 = q_1s_2 + s_1q_2$, $e_2 = q_1q_2 + s_1s_2$). This checker circuit is built using eight R gates. If a circuit contains more than two TBs a cascade of checker circuits is required. A fault in a circuit propagates through the checker circuits. By comparing the outputs of the last checker circuit, the circuit detects a fault.

Analysis

For analysis, we consider a function $f = ab + \overline{c}$. We rewrite this function as $f = abc \oplus \overline{c}$ and implement a non-testable circuit for this function using the basic ESOP-based method from [10]. The circuit has a quantum cost of 16 and 3 garbage outputs and is shown in Figure 5(a).

We then implement an online testable circuit for the same function, as shown in Figure 5(b). The first TB takes inputs a and b and generates $\overline{ab}$ which is then connected along with input c to the second TB, producing the final output $f = \overline{\overline{ab}c} = ab + \overline{c}$. The checker circuit checks the parity outputs of two TBs. The circuit should be able to detect a fault by examining the outputs (e_1 and e_2). The testable circuit has a quantum cost of 56 and 12 garbage outputs. Thus the overheads are 250% and 300% in terms of quantum cost and garbage outputs, respectively, compared to the non-testable circuit.

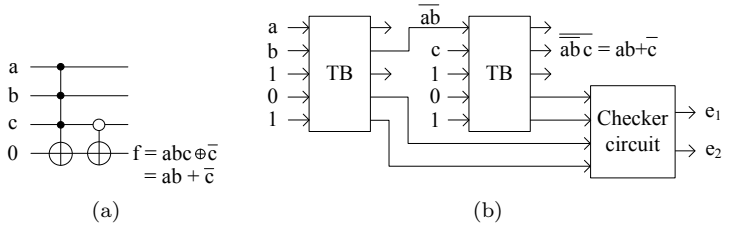

Fig. 5 (a) Non-testable circuit for $f = abc \oplus \overline{c} = ab + \overline{c}$ and (b) Online testable circuit for $f = ab + \overline{c}$ built using the design approach in [39].

A close investigation reveals that this approach cannot detect all single bit faults. If a fault occurs between two TBs, the circuit is unable to detect it. Note that TBs generate parities which are tested by checker circuits; thus a fault in any of the TBs is detected but occurrence of any fault outside the TBs is left undetected. For example, in the circuit given in Figure 5(b), if a fault occurs at the first input of the second TB, the circuit is unable to detect it. As a result this approach can detect some single bit faults, more specifically faults that occur in TBs, but cannot detect all faults.

3.2 Testable Circuit Design Using Testable Reversible Cells (TRCs)

Mahammad *et al.* [17] proposed an approach involving converting from an existing circuit. The first step transforms each $n \times n$ reversible gate G used in the circuit into an $(n + 1) \times (n + 1)$ deduced reversible gate DRG(G). Given the input vector $[k_1, k_2, \ldots, k_n]$ and output vector $[o_1, o_2, \ldots, o_n]$ of an $n \times n$ gate as shown in Figure 6(a), an extra input p_{iG} and the corresponding output p_{oG} are added to construct an $(n + 1) \times (n + 1)$ DRG(G) as shown

in Figure 6(b), which maps the input vector $[k_1, k_2, \ldots, k_n, p_{iG}]$ to the output vector $[o_1, o_2, \ldots, o_n, p_{oG}]$ where $p_{oG} = o_1 \oplus o_2 \oplus \ldots \oplus o_n \oplus p_{iG}$.

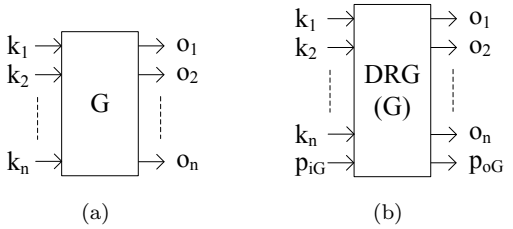

Fig. 6 (a) G gate and (b) DRG(G).

The second step constructs a testable reversible cell (TRC) of G, denoted by TRC(G). For example let us consider another $n \times n$ gate X which has the same input and output vectors; that is, the inputs of X pass through to the outputs without any change. DRG(X) is constructed by adding an input p_{iX} and the corresponding output p_{oX}. DRG(G) and DRG(X) are cascaded by connecting the first n outputs of DRG(G) to the first n inputs of DRG(X) in order to form an $(n+1) \times (n+1)$ TRC(G) with two parity outputs p_{oG} and p_{oX} as shown in Figure 7. Given $p_{iG} = p_{iX}$, p_{oG} and p_{oX} are complementary only if TRC(G) is faulty.

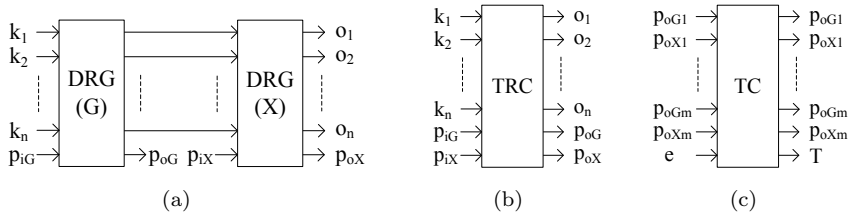

Fig. 7 (a) Construction of a testable reversible cell TRC(G), and (b) its block diagram. (c) Test cell (TC).

To create a testable circuit each gate in the circuit is replaced by its TRC. Let us assume that there are m TRCs in the circuit and that p_{oGj} and p_{oXj} are the parity outputs of the j^{th} TRC. To test all the parity outputs, a $(2m+1) \times (2m+1)$ test cell (TC) is formed. The first $2m$ inputs of the TC are the parity outputs which pass through to the outputs. The last input is e which is set to 0 or 1, and the corresponding output is $T = ((p_{oG1} \oplus p_{oX1}) + (p_{oG2} \oplus p_{oX2}) + \ldots + (p_{oGm} \oplus p_{oXm})) \oplus e$. The block diagram of the TC is given in Figure 7(c). According to the design, if a single bit fault occurs in the circuit then T becomes 1 provided $e = 0$.

Analysis

For analysis, we convert the non-testable circuit given in Figure 5(a) into an online testable circuit using this approach. The 4-bit Toffoli gate and 2-bit Toffoli gate in the non-testable circuit are replaced by a 6×6 TRC and 4×4 TRC. The parity outputs of these two TRCs are connected to a 5×5 TC to form a testable circuit as shown in Figure 8. This testable circuit has a quantum cost of 39 and 8 garbage outputs. Thus the overheads are 143.75% and 166.67% in terms of quantum cost and garbage outputs, respectively, compared to the non-testable circuit.

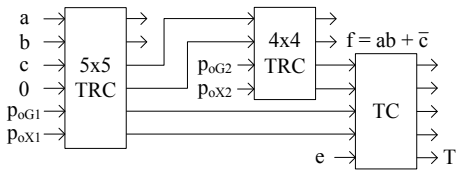

Fig. 8 Online testable circuit for $f = ab + \overline{c}$ built using the design approach in [17].

Unfortunately the design flaw that we described in the previous approach also exists in this design. For example, any fault between the connections of two TRCs in Figure 8 is undetectable. Thus this approach fails to detect a fault between two TRCs, contrary to the claims of the authors.

3.3 Testable Circuit Design Using Online Testable Gates (OTGs)

Thapliyal and Vinod [38] proposed an approach similar to the one described in Section 3.1. A new 4×4 reversible online testable gate (OTG) introduced in their work has a parity output at q as shown in Figure 9(a). The R2 gate (see Figure 3(b)) is combined with the OTG as shown in Figure 9 to design a block that is online testable. Two parity outputs of the testable block (s and q) are compared to check whether the block is faulty or not. In [39] (see Section 3.1) the two-pair two-rail checker circuit was designed using eight R gates, whereas Thapliyal and Vinod designed the checker circuit using four 3-bit Toffoli gates and two 3-bit Fredkin gates.

Analysis

In this analysis, we consider the same function $f = ab + \overline{c}$ as in previous sections. A testable circuit realizing this function is built using this approach as shown in Figure 10. In the circuit, the first block produces $\overline{ab}$ which is then fed along with input c to the second block. The second block produces $= ab+\overline{c}$. A checker circuit is required to test these two blocks. This testable circuit has a quantum cost of 52 and 12 garbage outputs. Thus the overheads are 225% and

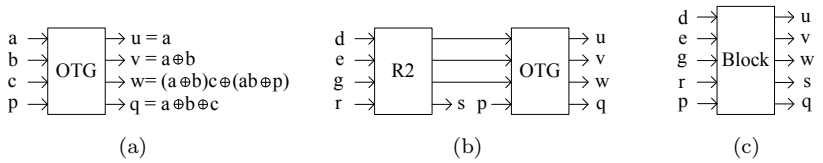

Fig. 9 (a) Online testable gate (OTG). (b) Construction of a testable block, and (c) its block diagram.

300% in terms of quantum cost and garbage outputs, respectively, compared to the non-testable circuit of Figure 5(a).

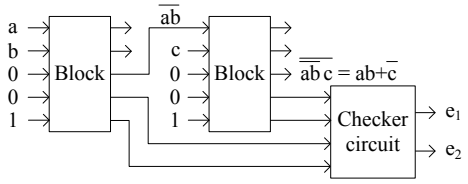

Fig. 10 Online testable circuit for $f = ab + \overline{c}$ built using the design approach in [38].

Like the previous two approaches, this approach also fails to detect a fault that occurs between two blocks.

3.4 Dual Rail Coding Approach

The dual rail coding approach proposed in [9] uses a set of dual rail reversible gates to design an online testable circuit. Each dual rail gate is a 4×4 gate which has two pairs of inputs and two pairs of outputs. It is noted that two signals are in dual rail form if they are complement of each other. In other words, a dual rail form represents either 01 or 10. Similarly, a non-dual form is either 11 or 00. A dual rail gate is designed in a way so that two inputs of each pair are given in the dual rail form and two outputs of each pair also appear as the dual rail form. Figure 11(a) shows a dual rail NAND gate. According to the design of the gate, a single fault in a gate causes the outputs of a pair to be in non-dual rail form. To implement a testable circuit, a cascade of dual rail gates is used. A fault in any dual rail gate propagates through the circuit. Thus by examining the output-pairs of last gate, the circuit can detect the fault. Consequently, this approach does not need a checker circuit for fault propagation and testing of intermediate gates, although it does require a checker circuit to test the final outputs.

Analysis

We again implement a testable circuit for the function $f = ab + \overline{c}$ according to this approach as shown in Figure 11(b). The inputs a, b, and their complemented forms are connected to a dual rail NAND gate. The two outputs ($\overline{ab}$ and ab) of this gate, input c and its complemented form are passed to another dual rail NAND gate. This gate produces $f = \overline{\overline{ab}c} = ab + \overline{c}$. The quantum cost of this circuit is 126. It also produces 5 garbage outputs. Thus the overheads are 687.5% and 66.67% in terms of quantum cost and garbage outputs, respectively, compared to the non-testable circuit of Figure 5(a). Although quantum costs of the dual rail gates are very high, this approach can detect any single bit fault in the circuit.

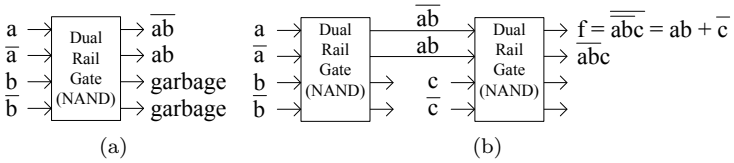

Fig. 11 (a) A dual rail NAND gate. (b) Online testable circuit for $f = ab + \overline{c}$ built using the design approach in [9].

3.5 Testable Circuit Design with Duplication of Gates

Kole, Rahaman and Das [15] proposed a technique that can detect online any single missing gate fault in a circuit consisting of only Toffoli gates. This technique requires one extra line T which is tested to detect the fault. For each n-bit Toffoli gate in the circuit, three extra Toffoli gates, two CNOT gates and another n-bit Toffoli gate are embedded to construct a testable circuit. Figure 12 shows how a Toffoli gate t_1 is surrounded by three Toffoli gates c_1, c_2, and t_2 to make it testable. If the initial value and the final value of the line T are the same then no gate is missing; otherwise the absence of a gate is assumed.

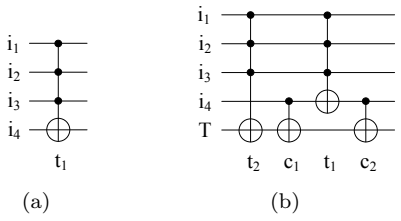

Fig. 12 (a) A Toffoli gate t_1 and (b) its corresponding testable circuit.

Analysis

The testable circuit implemented in this way requires 4 times as many gates as the non-testable design requires. The complexity of this technique in terms of quantum cost is $2g + 2q$, where g and q are the gate count and quantum cost of the non-testable design, respectively. Thus the quantum cost of this design is more than twice the cost of the non-testable design. As indicated above, this approach can detect any single missing gate fault.

4 Optimized Approaches for Online Fault Detection

The previous section has described a variety of approaches to generate online testable circuits. In this section we present a simple technique that takes a reversible circuit consisting only of NOT, CNOT and Toffoli gates and converts it into an online testable reversible circuit. The resulting circuit is able to detect any single bit faults. We present this approach in two stages; the first stage being designed for a specific type of Toffoli circuit, and the second stage being applicable to *any* type of Toffoli circuit.

4.1 Testing of ESOP-based Circuits

The basic ESOP-based reversible logic synthesis approach [10] and several of its variants [34, 35] all generate circuits which have separate input and output lines. A common structure of this type of circuit is that controls of the Toffoli gates are connected only to input lines and targets are connected only to output lines. This structure is maintained when the given ESOP-circuit is converted into a testable circuit, allowing easy detection of single bit faults in the resulting circuit. While techniques such as those listed above generate circuits with this structure, other synthesis techniques that ensured the requisite structure could also be used to generate a reversible circuit to which we could then apply our modifications to make it online testable.

4.1.1 ***Construction of a Testable Circuit from the ESOP-based Circuit***

Given a function with p inputs and q outputs, consider a reversible circuit which realizes this function and has the structure mentioned earlier. One method for generating a reversible circuit with the required structure is to create an empty circuit with p input lines and q output lines, and initialize each output line with zero. If the function is described as an exclusive-or sum of products (ESOP) then for each ESOP term (product) of each output a Toffoli gate is added at the end of the circuit. This method was proposed in [10] and creates a circuit with p input lines and q output lines.

To convert such circuit into an online testable circuit, we need to add some NOT gates and CNOT gates. It also requires a parity line L which is initialized with a zero. The procedure is as follows. We first replace every n-bit Toffoli gate in the given circuit by an $(n+1)$-bit ETG. The connections of the first n bits of the ETG remain the same as that of n-bit Toffoli gate. The last, *i.e.* $(n+1^{st})$ bit of the ETG is connected to L. The NOT gates in the given circuit are also kept. For each NOT gate in the input and output lines, one extra NOT gate is added on the line L. In total m extra NOT gates are required in this step, where m is number of NOT gates in the given circuit. In order to test the output lines, a CNOT gate is added from each output line to the L line at the end of circuit, requiring q more gates. To test the input lines, CNOT gates are added from each input line to L before and after the whole circuit. This step requires $2p$ CNOT gates. Now in the resulting circuit, if a single fault occurs in any of the input lines, output lines or even in L, the value of L at the end will become 1. If no fault occurs, L will remain 0. It is important to note that this technique can also be applied for the ESOP-based circuit consisting of inverted-control Toffoli gates. The following example describes the conversion procedure.

Example 1 *For a given 4-input (I_1, I_2, I_3, I_4), 2-output (I_5, I_6) ESOP-based circuit shown in Figure 13(a), the corresponding online testable circuit generated by the proposed technique is shown in Figure 13(b). We can see that Toffoli gates (t_1, t_2, t_3, and t_4) are replaced by ETGs (e_1, e_2, e_3, and e_4). CNOT gates c_1 through c_{10} are added to test the input and output lines.*

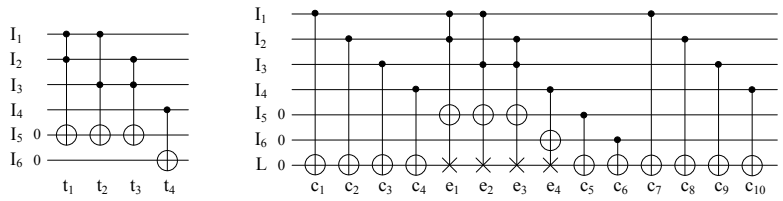

Fig. 13 (a) An ESOP-based circuit, and (b) equivalent online testable reversible circuit.

4.1.2 ***Analysis***

In our testable circuit, a single bit fault can generate multiple faults. Faults can occur in control bits and/or target bit of an ETG. A fault in a target bit affects the gate as it changes the target output. However, a fault in a control bit affects the gate only if it changes the target output to a faulty value. For instance, given that all controls of an ETG are positive, a fault in a control bit has an effect on the target output if all fault-free control bits have the value 1. Thus if a single fault in the control bit affects an ETG, then the fault propagates

to other lines by the target outputs. In our testable design, since controls of the ETGs and CNOTs are connected to input lines, faults on input lines can propagate to output lines and L by the target outputs. However, faults on output lines and L cannot cause other lines to be faulty since no controls are connected to output lines or L. Our proposed approach can detect any single bit fault even though it causes multiple faults. The following example shows the propagation of a single fault from an input line to multiple output lines.

Example 2 *Consider a circuit consisting of two ETGs as shown in Figure 14. This circuit has two input lines (I_1 and I_2), two output lines (I_3 and I_4), and a parity line L. The initial values of I_1, I_2, I_3, I_4, and L are 0, 1, 0, 1, and 0, respectively. Assume a fault occurs on I_1 just before the first ETG; thus the value of I_1 changes to 1. For the faulty lines, values are given in the form [fault-free value/ faulty value]. The fault on I_1 affects the first ETG and propagates to I_3 and L by the target outputs of this gate. As a result, two extra lines I_3 and L become faulty. The fault on I_1 also affects the second ETG. This causes I_4 to have the faulty value and fixes the value of L. Therefore, after the second ETG, lines I_1, I_3 and I_4 are faulty.*

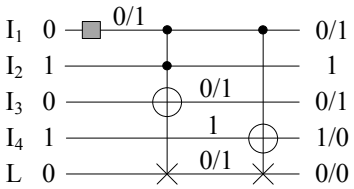

Fig. 14 Fault propagation in multiple lines.

Lemma 1 proves that both targets of an ETG calculate the same function regardless of the occurrence of the fault. In this lemma, we consider all controls of the ETG to be positive. A similar lemma can be proved if some or all controls are negative. Lemma 2 proves the correctness of our technique.

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

1. *If faults occur in controls of the ETG and affect the gate, then both o_n and o_{n+1} compute $\overline{f}$; otherwise both outputs compute f.*
2. *If faults occur in targets of the ETG, then both o_n and o_{n+1} compute f.*
3. *If faults occur in targets and controls which affect the ETG, then both o_n and o_{n+1} compute $\overline{f}$; otherwise both outputs compute f.*

Proof 1. Consider faults in $k_i, k_j, \ldots, k_l$ such that $\{i, j, \ldots, l\} \subseteq \{1, 2, \ldots, n-1\}$. These faults affect the calculation of function f if the following two conditions hold. [i)]

(a) $k_m = 1 \forall m \in \{1, 2, \ldots, n-1\} - \{i, j, \ldots, l\}$, and
(b) $k_i = k_j = \ldots = k_l = 0$ or $k_i = k_j = \ldots = k_l = 1$.

If both conditions are true, then the faults have impact on o_n and o_{n+1} since both outputs compute f. Thus $o_n = \overline{f} \oplus k_n$ and $o_{n+1} = \overline{f} \oplus k_{n+1}$. If at least one of the conditions is false, then the faults do not affect the calculation of f. Thus according to the definition, $o_n = f \oplus k_n$ and $o_{n+1} = f \oplus k_{n+1}$. Therefore if the faults have an effect, then o_n and o_{n+1} compute $\overline{f}$; otherwise these outputs compute f.

2. We consider three cases, depending on whether one of the targets is faulty or both targets are faulty.
   Case 1. Assume a fault occurs in k_n. This fault does not propagate to o_{n+1} since o_{n+1} is independent of k_n; thus $o_{n+1} = f \oplus k_{n+1}$. However due to the fault, $o_n = f \oplus \overline{k}_n$.
   Case 2. Assume a fault occurs in k_{n+1}. The proof is similar to Case 1. We get $o_n = f \oplus k_n$ and $o_{n+1} = f \oplus \overline{k}_{n+1}$.
   Case 3. Consider that faults occur in both k_n and k_{n+1}. Due to these faults, $o_n = f \oplus \overline{k}_n$ and $o_{n+1} = f \oplus \overline{k}_{n+1}$.
   Hence, for any of the cases, both o_n and o_{n+1} compute f.
3. We consider two cases, depending on whether faults in controls affect the gate or not.
   Case 1. First consider that faults in controls affect the gate. From Lemma 1(1), we can write $o_n = \overline{f} \oplus k_n$ and $o_{n+1} = \overline{f} \oplus k_{n+1}$.
   Assume that faults also occur in both targets. According to Lemma 1(2), we can rewrite the outputs as follows: $o_n = \overline{f} \oplus \overline{k}_n$ and $o_{n+1} = \overline{f} \oplus \overline{k}_{n+1}$. Thus both outputs compute $\overline{f}$. Similarly, we can reach the same conclusion if a fault occurs in any of the targets.
   Case 2. For the case that faults in controls have no effect on the gate, the proof is similar to Lemma 1(2).

Lemma 2 *If any single fault occurs on any line, the value of L changes to* 1 *and the fault is detected.*

Proof Consider an online testable circuit generated by the proposed technique which has N ETGs, p input lines, q output lines, and a parity line L. Let $G = \{g_1, g_2, \ldots, g_N\}$ be the set of ETGs used in the circuit and let $I_1, I_2, \ldots, I_p$ be the input lines and $I_{p+1}, I_{p+2}, \ldots, I_{p+q}$ be the output lines. Let the initial values of lines $I_1, I_2, \ldots, I_p$ be $i_1, i_2, \ldots, i_p$. All output lines and L are initialized by 0. Given an $(n+1)$-bit ETG $g_i \in G$ and the input vector $[k_1, k_2, k_3, \ldots, k_{n-1}, k_n, k_{n+1}]$, the output vector is $[k_1, k_2, k_3, \ldots, k_{n-1}, fg_i \oplus k_n, fg_i \oplus k_{n+1}]$ where $fg_i = k_1 k_2 \cdots k_{n-1}$, and n can be at most p+1. In order to prove this lemma, consider the following three cases:

Case 1. Assume that a single fault occurs on any input line, say I_z (for $z = 1, 2, \ldots, p$), which affects a set of gates, $X = \{x_1, x_2, \ldots, x_u\} \subseteq G$. Consider another set of gates, $Y = \{y_1, y_2, \ldots, y_v\}$ which is not affected by the fault. We have $G = X \cup Y$, and X or Y can be empty.

As described before, the last two outputs of a gate $g_i \in G$ computes the same function fg_i. A gate y_r in Y computes fy_r, for $r = 1, 2, \ldots, v$. However,

from Lemma 1(1), due to the fault, each gate x in X computes $\overline{fx}_s$, for $s = 1, 2, \ldots, u$.

After the last gate in G, the line I_z has the faulty value $\bar{i}_z$, and L becomes $i_1 \oplus i_2 \oplus \ldots \oplus i_z \oplus \ldots \oplus i_p \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_v \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_u$. When all output lines are EXORed to L at the end, L becomes

$i_1 \oplus i_2 \oplus \ldots \oplus i_z \oplus \ldots \oplus i_p \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_v \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_u \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_v \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_u$

$= i_1 \oplus i_2 \oplus \ldots \oplus i_z \oplus \ldots \oplus i_p$

Finally, when all input lines are EXORed to L, the fault on I_z is propagated to L, and L becomes

$i_1 \oplus i_2 \oplus \ldots \oplus i_z \oplus \ldots \oplus i_p \oplus i_1 \oplus i_2 \oplus \ldots \oplus \bar{i}_z \oplus \ldots \oplus i_p$

$= i_z \oplus \bar{i}_z = 1.$

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

Case 2. Now consider that a single fault occurs on an output line I_{p+z} at any point, where $z = 1, 2, \ldots, q$.

Consider a set of gates, $X = \{x_1, x_2, \ldots, x_u\} \subseteq G$ which have targets connected on line I_{p+z} after the occurrence of the fault. Consider another set of gates, $Y = \{y_1, y_2, \ldots, y_v\}$ such that $Y = G - X$. From Lemma 1(2), each gate x_s in X computes fx_s, for $s = 1, 2, \ldots, u$. According to the definition, each gate y_r in Y computes fy_r, for $r = 1, 2, \ldots, v$.

The initial value of I_{p+z} is 0 and it becomes 1 because of the fault. At the end, the value of I_{p+z} is $fx_1 \oplus fx_2 \oplus \ldots \oplus fx_u \oplus 1$. As a result, when output and input lines are EXORed to L at the end, the faulty value of I_{p+z} appears at L. Thus L becomes

$i_1 \oplus i_2 \oplus \ldots \oplus i_p \oplus fx_1 \oplus fx_2 \oplus \ldots \oplus fx_u \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_v \oplus i_1 \oplus i_2 \oplus \ldots \oplus i_p \oplus fx_1 \oplus fx_2 \oplus \ldots \oplus fx_u \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_v \oplus 1 = 1$

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

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

4.1.3 ***Advantages of the Proposed Design***

Our proposed approach has several advantages over the previous approaches, which are listed below.

- The approaches in [39] and [9] make use of new gates such as the R1 gate, R2 gate, R gate, and dual rail gates. These approaches require new synthesis techniques to implement the testable circuits since no existing synthesis techniques work with these new gates. In contrast, our proposed approach is quite simple since it works on top of the non-testable circuit designed using the ESOP-based synthesis technique.
- The approaches in [39] and [17] need checker circuits to detect the fault. Although the approach in [9] does not need intermediate checker circuits, it does require a checker circuit for testing the final outputs. Our proposed approach does not require any checker circuit.
- Our approach does not produce any new garbage outputs.

4.2 Testing of General Toffoli Circuits

A Toffoli circuit consists of only Toffoli gates including NOTs, CNOTs and negative-control Toffoli gates. In general in a reversible circuit the input lines can become the output lines; this is the ideal case, in which no garbage is generated. The target and controls of a Toffoli gate can be connected to any line in the circuit. In Section 4.1 we proposed an approach for constructing online testable reversible circuits from a *certain type* of ESOP-based Toffoli circuit. In this section we extend our approach to work for any type of Toffoli circuit.

4.2.1 ***Construction of a Testable Circuit from the Toffoli Circuit***

Consider a Toffoli circuit consisting of p lines; in order to convert such circuit into an online testable circuit, a parity line L is added, which is initialized with a 0. The detailed procedure is given below.

A CNOT gate is inserted from each line to L at the beginning of the given circuit. Every n-bit Toffoli gate is replaced by an $(n+1)$-bit ETG. The connections of the first n bits of the ETG are kept the same as that of n-bit Toffoli gate. The last bit of the ETG is connected to L. All NOT gates found on the lines are retained. If the number of NOT gates in the circuit is an odd number, an extra NOT gate is added at the end of line L; otherwise, no extra NOT gate is added. Finally, a CNOT gate is added at the end from each line to L.

This approach requires a total of $2p$ extra CNOT gates and at most one extra NOT gate to construct an online testable circuit. In the testable circuit, if a single fault occurs in any line (including L), the value of L changes from 0 to 1. If no fault occurs, the value of L remains 0. Thus the circuit detects the fault by checking the line L. Note that this procedure also works for any circuit consisting of inverted-control Toffoli gates.

Although this approach seems similar to our first approach, there are differences between the two. The first difference lies in optimizing the number of NOT gates. This approach is optimized in terms of NOT gates since at most one extra not gate is added on line L depending on the total number of NOT gates in the given circuit. In contrast, the first approach adds as many extra NOT gates as is found in the given circuit. Another difference is that our first approach does not require CNOT gates from the output lines to L at the beginning of the circuit since each output line is initialized with a constant zero. However, the second approach does require a CNOT gate from each line to L at the beginning of the circuit.

Example 3 *Consider a Toffoli circuit given in Figure 15(a) which has five lines (I_1, I_2, ..., I_5) and four Toffoli gates (t_1, t_2, t_3, and t_4). For constructing an online testable circuit, we add a parity line L as well as replace t_1, t_2, t_3, and t_4 by ETGs e_1, e_2, e_3, and e_4. We also insert five CNOT gates c_1 through c_5 at the beginning and five CNOT gates c_6 through c_{10} at the end of circuit. The resultant online testable circuit is shown in Figure 15(b).*

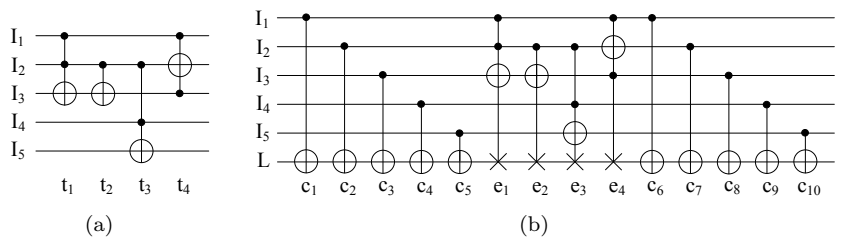

Fig. 15 (a) A Toffoli circuit and (b) an online testable circuit.

4.2.2 ***Analysis***

As described in Section 4.1.2, a single fault can generate multiple faults. In a testable circuit, a fault on a line (except parity line L) can cause any other lines including L to become faulty. It is important to note that a fault on L cannot propagate to other lines since controls of the CNOTs and ETGs are not connected to L. Lemma 3 proves that this approach can detect any single bit fault even if it propagates to other lines, generating multiple faults.

Lemma 3 *If any single fault occurs on any line, the circuit is able to detect it.*

Proof Consider an online testable circuit generated by the proposed technique which has N ETGs, p lines (I_1, I_2, ..., I_p), and a parity line L. Let $G = \{g_1, g_2, \ldots, g_N\}$ be the set of ETGs used in the circuit. Let the initial values of lines $I_1, I_2, \ldots, I_p$ be $i_1, i_2, \ldots, i_p$. The line L is initialized with a 0. Given an $(n+1)$-bit ETG $g \in G$ and the input vector $[k_1, k_2, k_3, \ldots, k_{n-1}, k_n, k_{n+1}]$, the output vector is $[k_1, k_2, k_3, \ldots, k_{n-1}, fg \oplus k_n, fg \oplus k_{n+1}]$, where $fg = k_1 k_2 \cdots k_{n-1}$, and n can be at most p. In order to prove this lemma, consider the following two cases.

Case 1. Assume that a single fault occurs on a line I_d (for $d = 1, 2, \ldots, p$) and propagates to multiple lines.

Let $W = \{w_1, w_2, \ldots, w_q\} \subseteq$ G be the set of gates that are not affected by the faults.

Let $X = \{x_1, x_2, \ldots, x_r\} \subseteq$ G be the set of gates such that faults occur only on controls and affect the gates.

Let $Y = \{y_1, y_2, \ldots, y_s\} \subseteq$ G be the set of gates with faults only on targets.

Let $Z = \{z_1, z_2, \ldots, z_t\} \subseteq$ G be the set of gates such that faults occur on targets and controls which affect the gates.

We have $G = W \cup X \cup Y \cup Z$, and W, X, Y or Z can be empty. According to the definition of the ETG, two targets of a gate $g \in G$ compute the same function fg. Similarly, a gate $w_j \in W$ computes fw_j for $j = 1, 2, \ldots, q$. A gate $x_l \in X$ computes $\overline{fx_l}$ for $l = 1, 2, \ldots, r$ according to Lemma 1(1). A gate $y_u \in Y$ computes fy_u for $u = 1, 2, \ldots, s$ according to Lemma 1(2). A gate $z_v \in Z$ computes $\overline{fz_v}$ for $v = 1, 2, \ldots, t$ according to Lemma 1(3).

At the beginning of the circuit, all I_m lines, for $m = 1, 2, \ldots, p$ are EXORed to parity line L. Thus the value of L, just before the first gate in G, is $i_1 \oplus i_2 \oplus \ldots \oplus i_d \oplus \ldots \oplus i_p$. After the last gate in G, the value of L becomes $i_1 \oplus i_2 \oplus \ldots \oplus i_d \oplus \ldots \oplus i_p \oplus fw_1 \oplus fw_2 \oplus \ldots \oplus fw_q \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_r \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_s \oplus \overline{fz}_1 \oplus \overline{fz}_2 \oplus \ldots \oplus \overline{fz}_t$.

At the end, when all I_m lines, for $m = 1, 2, \ldots, p$ are EXORed to L again, the faulty value of I_d (which is $\bar{i}_d$) propagates to L and hence L becomes $i_1 \oplus i_2 \oplus \ldots \oplus i_d \oplus \ldots \oplus i_p \oplus fw_1 \oplus fw_2 \oplus \ldots \oplus fw_q \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_r \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_s \oplus \overline{fz}_1 \oplus \overline{fz}_2 \oplus \ldots \oplus \overline{fz}_t \oplus i_1 \oplus i_2 \oplus \ldots \oplus \bar{i}_d \oplus \ldots \oplus i_p \oplus fw_1 \oplus fw_2 \oplus \ldots \oplus fw_q \oplus \overline{fx}_1 \oplus \overline{fx}_2 \oplus \ldots \oplus \overline{fx}_r \oplus fy_1 \oplus fy_2 \oplus \ldots \oplus fy_s \oplus \overline{fz}_1 \oplus \overline{fz}_2 \oplus \ldots \oplus \overline{fz}_t$

$= i_d \oplus \bar{i}_d = 1.$

Since at the end, the line L contains 1, the fault is detected.

Case 2. A fault can also occur on L. This causes L to have the value 1 since it was initialized with a 0. Hence, the circuit detects the fault. Note that a fault on I_d can cause other lines faulty (Case 1). However, a fault on L does not propagate to any I_d since controls of ETGs and CNOTs are not connected to L.

5 Experimental Results

A number of benchmark circuits have been collected from [44]. For approaches in [39] and [9], no synthesis technique is given, and none of the currently available reversible synthesis techniques provide output of the type required for these approaches (*e.g.* cascades of the specific types of gates proposed in the two papers). Thus we used the SIS synthesis tool [36] to simplify the benchmarks and map into traditional logic gates (AND, OR, NAND and NOR gates), and then we have implemented an approach to replace these gates with the specific gates from each paper. For the approach in [39] this involved replacing the original gates with TBs and adding a checker circuit to implement the testable circuits. For the approach in [9] we replaced the original gates with dual rail gates to implement the testable circuits.

The approach in [17] and our two approaches require that the circuit be synthesized prior to modification for the testability. For the first experiment we synthesized the benchmark circuits using the basic ESOP-based method [10]. We then applied the approach in [17] and our first approach (the approach requiring the specific structure of the circuit) to implement the testable circuits. For the second experiment we used the improved shared cube synthesis technique from [26] to implement the Toffoli circuits. We then applied the approach in [17] and our second approach to implement the testable circuits. For each of the above approaches we calculated the quantum cost and garbage outputs of the testable circuits. The results are summarized in Table 1. In this table Approach A represents the approach in [39], Approach B represents the approach in [9], Approach C represents the approach in [17] using the ESOP-based synthesis technique, Approach D represents the approach in [17] using

the improved shared cube synthesis technique, and Approach E represents the approach in [38]. Also note that QC and GO columns represent the quantum cost and the number of garbage outputs, respectively.

From Table 1, we can see that both of our approaches reduce the quantum cost and garbage outputs significantly for every circuit. It is noted that we do not compare Approach D and our first approach since the underlying synthesis techniques are different. On average, each of our approaches produces 15 garbage outputs whereas the best existing approach produces 484 garbage outputs.

Table 2 summarizes the improvements achieved by our approach. Our first approach can decrease the quantum cost by 54.02%, 61.91%, 22.65%, 54.02% on average, as compared to [39], [9], [17], and [38], respectively. Average improvement of the quantum cost achieved by our second approach is up to 79.53%. Minimization of garbage outputs ranges from 96.90% to 99.85% on average, compared to the previous approaches.

The overhead costs of the existing work and our second approach over the non-testable designs are given in Table 3. The non-testable circuits were implemented using the improved shared cube synthesis technique [26]. The quantum cost overhead for our second approach is only 4.03% on average, compared to 321.06% for the approach in [39], 408.21% for the approach in [9], 40.53% for the approach in [17], and 321.02% for the approach in [38]. Our approach has absolutely no overhead in terms of garbage outputs since the testability feature does not produce any extra garbage. However, existing approaches produce extremely large numbers of garbage outputs and the average overheads are 65642.33%, 5684.82%, and 3162.39% for the approaches in [39], [9], and [17], respectively.

5.1 Coverage of Fault Models

Like the approaches in [39] and [17], our proposed work considers the single bit fault model. It is noted that the single bit fault model and single stuck-at fault model are very similar with the exception that the stuck-at fault model is independent on the initial values of the variables. Moreover, the behavior of a stuck-at fault can be translated into that of a bit fault, and vice versa. Consequently, a testable design which can detect bit faults can also detect stuck-at faults. The lemmas which are provided to prove the correctness of our approaches hold for both fault models.

As described in Sections 3.1, 3.2 and 3.3, previous approaches in [17,38,39] fail to detect all single faults; thus these approaches partially cover the bit fault and stuck-at fault models. In contrast, both of our approaches as well as the approach in [9] fully cover two fault models. Table 4 summarizes this discussion.

Table 1 Comparison of different online testable approaches.

Circuit	Approach A		Approach B		Approach C		Approach D		Approach E		1st approach		2nd approach	
	QC	GO	QC	GO	QC	GO	QC	GO	QC	GO	QC	GO	QC	GO
9symml	25598	5701	32220	532	12262	139	12262	139	25594	5701	11065	9	11066	9
alu2	40992	9123	52500	858	6549	215	6104	205	40988	9123	4882	10	4551	10
alu4	132367	29470	169332	2738	63165	1342	54160	1149	132363	29470	48951	14	42071	14
apex5	195061	43692	213843	3518	69596	1557	47847	1620	195057	43692	53947	117	35354	117
apla	17815	3979	20574	334	5457	185	2934	190	17811	3979	3978	10	1859	10
bw	19019	4216	25140	386	8757	620	4464	723	19015	4216	4994	5	1249	5
c17	486	109	564	12	190	23	190	23	482	109	103	5	103	5
cm82a	2163	482	2823	50	370	45	385	48	2159	482	176	5	173	5
cm163a	5259	1181	5940	106	1507	106	1099	104	5255	1181	977	16	661	16
co14	13429	3018	14163	250	3983	49	3983	49	13425	3018	3529	14	3529	14
con1	1647	370	1929	38	307	30	307	30	1643	370	184	7	184	7
cu	5560	1248	6234	110	1834	82	1288	84	5556	1248	1327	14	876	14
dc1	4141	919	5481	88	905	92	562	82	4137	919	435	4	204	4
dc2	11967	2666	15141	250	3066	143	1977	136	11963	2666	2088	8	1209	8
dist	38756	8634	48384	788	10594	363	5551	243	38752	8634	7786	8	3909	8
dk17	10376	2320	11478	186	2463	95	1649	95	10372	2320	1791	10	1113	10
ex2	1733	387	2241	42	267	23	267	23	1729	387	171	5	171	5
ex5p	84852	18914	98898	1540	38740	1518	12484	1623	84848	18914	26970	8	4918	8
f2	1948	434	2322	38	533	44	317	39	1944	434	306	4	152	4
f51m	128927	28694	164889	2670	42590	937	37323	832	128923	28694	33165	14	29072	14
frg2	218109	48779	247332	4048	264020	5073	150509	3616	218105	48779	205969	143	115211	143
ham7	5861	1301	8367	142	653	92	560	100	5857	1301	259	7	162	7
inc	13945	3102	17754	290	3679	197	2051	195	13941	3102	2389	7	1065	7
majority	787	177	876	18	224	18	224	18	783	177	154	5	154	5
max46	15321	3417	19314	326	5585	114	5585	114	15317	3417	4627	9	4627	9
misex1	7151	1593	8844	146	1694	111	941	113	7147	1593	1040	8	439	8
misex3	141354	31498	174732	2816	151363	2987	69153	2149	141350	31498	117215	14	50826	14
mlp4	36477	8116	46950	762	5917	248	4043	208	36473	8116	4094	8	2680	8
pdc	124627	27784	148926	2386	127947	2044	44931	1639	124623	27784	101170	16	32332	16
radd	4829	1075	6570	114	1508	126	1330	111	4825	1075	836	8	748	8
rd32	873	194	1257	24	110	18	104	18	869	194	49	3	45	3
rd73	16482	3674	20583	342	1862	140	1617	115	16478	3674	1064	7	959	7
root	14246	3174	18063	300	5073	176	2734	128	14242	3174	3773	8	1928	8
sqn	11236	2503	14256	240	3025	122	2053	100	11232	2503	2209	7	1437	7
sqrt8	5689	1269	7086	122	1056	76	835	63	5685	1269	657	8	530	8
sym9	24652	5485	31890	528	12262	139	12262	139	24648	5485	11065	9	11066	9
table3	223054	49769	265566	4294	112361	2019	31011	1767	223050	49769	87880	14	20050	14
tial	134689	29992	169002	2738	65578	1327	56703	1162	134685	29992	51314	14	44366	14
wim	2378	528	2934	46	639	64	462	62	2374	528	325	4	200	4
z4ml	3883	865	5070	88	1114	97	1048	92	3879	865	607	7	575	7
Average	43693	9746	52737	858	25970	570	14583	484	43689	9746	20088	15	10796	15

Approach A is [39], Approach B is [9], Approach C is [17] using ESOP-based synthesis technique, Approach D is [17] using improved shared cube synthesis technique, Approach E is [38].

Table 2 Improvements achieved by our approach.

Existing approaches	Achieved improvements			
	Our 1st approach		Our 2nd approach	
	QC	GO	QC	GO
Approach in [39]	54.02%	99.85%	75.29%	99.85%
Approach in [9]	61.91%	98.25%	79.53%	98.25%
Approach in [17]	22.65%	97.37%	25.97%	96.90%
Approach in [38]	54.02%	99.85%	75.29%	99.85%

Table 3 Overhead calculation of the testable design over the non-testable design.

Approach	Average overhead	
	QC	GO
Approach in [39]	321.06%	65642.33%
Approach in [9]	408.21%	5684.82%
Approach in [17]	40.53%	3162.39%
Approach in [38]	321.02%	65642.33%
Our 2nd approach	4.03%	0%

Table 4 Coverage of fault models.

Approach	Bit / stuck-at fault model
Approach in [39]	Partial
Approach in [17]	Partial
Approach in [38]	Partial
Approach in [9]	Complete
Our Approaches	Complete

6 Conclusion

In this paper we have reviewed previous approaches for online testing of reversible circuits and discussed their design issues and limitations. We have also presented two approaches that overcome the limitations of previous work. Our first proposal is a technique which converts a circuit synthesized from an ESOP [10] to an online testable circuit which can detect any single bit fault, first described in [27]. Our second proposed approach [25] allows conversion in a similar manner, but of any reversible circuit consisting solely of Toffoli gates. Our techniques compare favorably with the existing approaches, providing better fault coverage, as evidenced by the analysis we have presented in this work, and as well having significantly lower overhead in terms of garbage lines and quantum cost. The extensive analysis and comparisons based on benchmark tests, as reported in this paper, demonstrate conclusively the significant improvements achieved by our two approaches. We highlight that although we have applied our modifications for the testability to circuits generated by ESOP-based synthesis techniques, the underlying synthesis technique need not be a factor in the success of our techniques. That is, our techniques can be applied to any circuit consisting of only Toffoli gates, regardless of the synthesis technique that generates the circuit. In addition any new synthesis technique that generates a minimal (or close to) reversible circuit consisting of only Toffoli gates could be used to generate the initial circuit, and even lower overhead could be achieved if fewer gates and lines were generated by such a synthesis technique (fewer gates means less ETG-modifications, and fewer lines means fewer CNOT gates to be added). On-going work includes the extension of this technique to other fault models as well as to multiple-valued reversible logic circuits and to reversible circuits that incorporate other types of gates.

References

1. International technology roadmap for semiconductors (itrs). http://public.itrs.net, 2009 executive summary (2009)
2. Arabzadeh, M., Saeedi, M., Zamani, M.: Rule-based optimization of reversible circuits. In: Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC), pp. 849–854 (2010)
3. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Phys. Rev. A **52**(5), 3457–3467 (1995). DOI 10.1103/PhysRevA.52.3457
4. Bennett, C.H.: Logical reversibility of computation. IBM Journal of Research and Development **17**(6), 525–532 (1973)

5. Chakraborty, A.: Synthesis of reversible circuits for testing with universal test set and C-testability of reversible iterative logic arrays. In: Proceedings of the 18th International Conference on VLSI Design, pp. 249–254 (2005)
6. Chen, J., Zhang, X., Wang, L., Wei, X., Zhao, W.: Extended Toffoli gate implementation with photons. In: Proceedings of 9th International Conference on Solid-State and Integrated-Circuit Technology (ICSICT), pp. 575–578. China (2008)
7. De Vos, A., Burignat, S.p., Thomsen, M.K.: Reversible implementation of a discrete integer linear transformation. JOURNAL OF MULTIPLE-VALUED LOGIC AND SOFT COMPUTING **18**(1), 25–35 (2012)
8. Desoete, B., De Vos, A.: A reversible carry-look-ahead adder using control gates. Integration, the VLSI Journal **33**(1), 89–104 (2002). DOI 10.1016/S0167-9260(02)00051-2. URL `http://dx.doi.org/10.1016/S0167-9260(02)00051-2`
9. Farazmand, N., Zamani, M., Tahoori, M.B.: Online fault testing of reversible logic using dual rail coding. In: Proceedings of 16th IEEE International On-Line Testing Symposium (IOLTS), pp. 204–205 (2010)
10. Fazel, K., Thornton, M., Rice, J.E.: ESOP-based Toffoli gate cascade generation. In: Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 206–209. Victoria, BC, Canada (2007)
11. Frank, M.P.: Introduction to reversible computing: motivation, progress, and challenges. In: Proceedings of the 2nd Conference on Computing Frontiers, pp. 385–390. ACM Press, Ischia, Italy (2005)
12. Hayes, J.P., Polian, I., Becker, B.: Testing for missing-gate faults in reversible circuits. In: Proceedings of the 13th Asian Test Symposium, pp. 100–105 (2004). DOI http://dx.doi.org/10.1109/ATS.2004.84. URL `http://dx.doi.org/10.1109/ATS.2004.84`
13. Ibrahim, M., Chowdhury, A.R., Babu, H.M.H.: Minimization of CTS of k-CNOT circuits for SSF and MSF model. In: Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp. 290–298. Boston, MA (2008)
14. Kim, S., Chae, S.I.: Implementation of a simple 8-bit microprocessor with reversible energy recovery logic. In: Proceedings of the 2nd Conference on Computing Frontiers, pp. 421–426. ACM Press, Ischia, Italy (2005)
15. Kole, D.K., Rahaman, H., Das, D.K.: Synthesis of online testable reversible circuit. In: Proceedings of 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp. 277–280. Vienna (2010)
16. Landauer, R.: Irreversibility and heat generation in the computing process. IBM Journal of Research and Development **5**, 183–191 (1961)
17. Mahammad, S.N., Veezhinathan, K.: Constructing online testable circuits using reversible logic. IEEE Transactions on Instrumentation and Measurement **59**(1), 101–109 (2010)
18. Maslov, D.: Reversible logic synthesis benchmarks page. `http://www.cs.uvic.ca/~dmaslov/` (2012)
19. Maslov, D., Dueck, G.W., Miller, D.M., Negrevergne, C.: Quantum circuit simplification and level compaction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(3), 436–444 (2008)
20. Merkle, R.C.: Reversible electronic logic using switches. Nanotechnology **4**(1), 21–40 (1993)
21. Miller, D., Wille, R., Sasanian, Z.: Elementary quantum gate realizations for multiple-control toffoli gates. In: Proceedings of the 41st IEEE International Symposium on Multiple-Valued Logic (ISMVL), pp. 288–293 (2011). DOI 10.1109/ISMVL.2011.54
22. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation based algorithm for reversible logic synthesis. In: Proceedings of the 40th annual Design Automation Conference (DAC), pp. 318–323 (2003)
23. Mohammadi, M., Eshghi, M.: On figures of merit in reversible and quantum logic designs. Quantum Information Processing **8**, 297–318 (2009). DOI 10.1007/s11128-009-0106-0. URL `http://portal.acm.org/citation.cfm?id=1555567.1555601`
24. Moore, G.E.: Progress in digital integrated electronics. In: Technical Digest 1975 IEEE International Electron Devices Meeting, pp. 11–13 (1975)
25. Nayeem, N.M., Rice, J.E.: Online fault detection in reversible logic. In: Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), pp. 426–434. Vancouver, Canada (2011)

26. Nayeem, N.M., Rice, J.E.: A shared-cube approach to ESOP-based synthesis of reversible logic. Facta Universitatis Series: Electronics and Energetics **24**(3), 385–402 (2011)
27. Nayeem, N.M., Rice, J.E.: 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. 85–90. Victoria, Canada (2011)
28. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press (2000)
29. Patel, K.N., Hayes, J.P., Markov, I.L.: Fault testing for reversible circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **23**(8), 1220–1230 (2004)
30. Picton, P.: Optoelectronic, multivalued, conservative logic. International Journal of Optical Computing **2**, 19–29 (1991)
31. Polian, I., Hayes, J.P., Fiehn, T., Becker, B.: A family of logical fault models for reversible circuits. In: Proceedings of the 14th Asian Test Symposium (ATS), pp. 422–427. 8–21 Dec., Calcutta, India (2005)
32. Rahaman, H., Kole, D.K., Das, D.K., Bhattacharya, B.B.: On the detection of missing-gate faults in reversible circuits by a universal test set. In: Proceedings of the 21st International Conference on VLSI Design, pp. 163–168 (2008). DOI http://dx.doi.org/10.1109/VLSI.2008.106. URL `http://dx.doi.org/10.1109/VLSI.2008.106`
33. Rice, J.E.: An overview of fault models and testing approaches for reversible logic (2013). Submitted to the 2013 Pacific Rim Conference on Computers, Communications, and Signal Processing
34. Rice, J.E., Suen, V.: Using autocorrelation coefficient-based cost functions in ESOP-based Toffoli gate cascade generation. In: Proceedings of 23rd Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1–6. Calgary, Canada (2010)
35. Sanaee, Y., Dueck, G.W.: ESOP-based Toffoli network generation with transformations. In: Proceedings of 40th IEEE International Symposium on Multiple-Valued Logic, pp. 276–281 (2010)
36. Sentovich, E., Singh, K., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: Sis: A system for sequential circuit synthesis. Tech. Rep. UCB/ERL M92/41, EECS Department, University of California, Berkeley (1992). URL `\url{http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/2010.html}`. Software downloaded Mar. 2012 from `http://web.cecs.pdx.edu/~alanmi/research/soft/softPorts.htm`; documentation accessed at `http://embedded.eecs.berkeley.edu/pubs/downloads/sis/index.htm`
37. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**(6), 710–722 (2003)
38. Thapliyal, H., Vinod, A.P.: Designing efficient online testable reversible adders with new reversible gate. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1085–1088. New Orleans, LA (2007)
39. Vasudevan, D.P., Lala, P.K., Jia, D., Parkerson, J.P.: Reversible logic design with online testability. IEEE Transactions on Instrumentation and Measurement **55**(2), 406–414 (2006)
40. Vos, A.D.: Reversible Computing: Fundamentals, Quantum Computing, and Applications. Wiley-VCH, Weinheim (2010). Chapter 4.5: An Application: Prototype Chips
41. Vos, A.D., Rentergem, Y.V.: Synthesis of reversible logic for nanoelectronic circuits. International Journal of Circuit Theory and Applications **35**(3), 325–341 (2007). DOI 10.1002/cta.413. Published online 17 April 2007 in Wiley InterScience (www.interscience.wiley.com)
42. Wang, L., Wu, C., Wen, X. (eds.): VLSI Test Principles and Architectures: Design for Testability. Morgan Kaufmann (2006)
43. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic. International Journal of Applied Metaheuristic Computing (IJAMC) **1**(4), 25–41 (2010). DOI 10.4018/jamc.2010100102
44. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: An online resource for reversible functions and reversible circuits. In: Proceedings of 38th International

Symposium on Multiple Valued Logic, pp. 220–225 (2008). RevLib is available at `http://www.revlib.org`

45. Wille, R., Kesz ocze, O., Drechsler, R.: Determining the minimal number of lines for large reversible circuits. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2011, pp. 1–4 (2011). DOI 10.1109/DATE.2011.5763314
46. Zhong, J., Muzio, J.C.: Analyzing fault models for reversible logic circuits. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 2422–2427. Vancouver, BC (2006)