An Overview of Fault Models and Testing Approaches for Reversible Logic

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

***Abstract*—Reversible logic has been proposed as one solution to the problem of ever increasing power consumption. Work in areas such as synthesis techniques in reversible logic is growing, as is work on testing approaches. Numerous fault models have been proposed, but many researchers are still utilising models proposed for traditional logic. We provide an overview of the various fault models and testing approaches for reversible logic, as well as highlighting important results and comparisons/connections between the various models.**

I. Introduction

In the field of electronics consumers are demanding smaller and smaller devices, with lower and lower power consumption; unfortunately today's technologies are having difficulties keeping up and power consumption and dissipation are becoming serious issues. However a solution is offered in the form of reversible logic, and in fact Frank has stated that "...[a] computer based mainly on reversible logic operations can reuse a fraction of the signal energy that theoretically can approach arbitrarily near to 100% as the quality of the hardware is improved..." [1]. This is not a new idea; in 1961 Landauer published the first paper linking reversible logic to lower power dissipation [2], and in 1973 Bennett showed that for power *not* to be dissipated it is necessary that a binary circuit be constructed from reversible gates [3]. Reversible logic also has connections to quantum computing, and reversible circuits can be viewed as a special case of quanutm circuits [4]. As Shende *et al.* note, "[w]hile the speed-ups which make quantum computing attractive are not available without purely quantum gates, logic synthesis for classical reversible circuits is a first step toward synthesis of quantum circuits" [5]. Although the area of reversible logic is not new, it is only now that it seems to be gaining interest, and only recently that techniques for testing reversible logic circuits have been proposed.

The paper progresses as follows: section II provides some background on reversible logic in general, followed by overviews of existing work on fault models and testing in sections III and IV. The paper finishes with a section offering comparisons and connections between the various testing approaches, with the goal of offering a new perspective on these fault models.

II. Reversible Logic Circuits

The term reversible function is used to refer to a function that is bijective. For example, Figure 1 gives two functions, one of which (A) is reversible while the other (B) is not. A logic gate can then be considered reversible if the function it computes is bijective [5], and a circuit is considered reversible if it consists entirely of reversible gates. We assume in this work that the functions in question are limited to the Boolean domain, although this is not a requirement for reversible functions, gates and circuits.

$x\ y$	$x'\ y'$
00	00
01	10
10	01
11	11

(A)

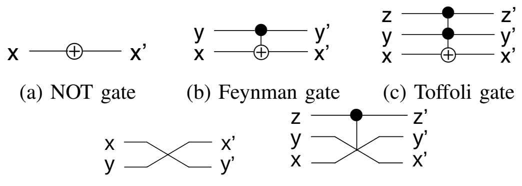
$x\ y$	$f(x,y)$
00	0
01	0
10	0
11	1

(B)

Fig. 1. (A) An example of a reversible function. (B) An example of an irreversible function.

TABLE I. The behaviour of a selection of more commonly used reversible logic gates.

gate	behaviour
Not	$(x) \rightarrow (x \oplus 1)$
Feynman	$(y, x) \rightarrow (y, x \oplus y)$
Toffoli	$(z, y, z) \rightarrow (z, y, x \oplus yz)$
swap	$(x, y) \rightarrow (y, x)$
Fredkin	$(z, y, x) \rightarrow (z, x, y)$ iff $z = 1$
	else $(z, y, x) \rightarrow (z, y, x)$



(a) NOT gate (b) Feynman gate (c) Toffoli gate

(d) swap gate (e) Fredkin gate

Fig. 2. Symbols for some of the more commonly-used reversible logic gates.

Table I lists the behaviour of each of the most commonly used reversible gates. Figure 2 illustrates the symbols usually used for each of the gates. Controls are shown as solid dots where the gate intersects a line. Toffoli gates with varying numbers of controls are sometimes referred to as k-CNOT gates, where k refers to the number of lines that control how the gate affects the target line. For instance, the Toffoli gate shown in Figure 2 could be referred to as a 2-CNOT gate, while the Feynman gate would be referred to as a 1-CNOT gate and the NOT gate would be a 0-CNOT gate [6]. Circuits that are designed using only NOT, Feynman (CNOT) and Toffoli gates are said to be designed using the *NCT*-library, which is a common approach in reversible logic synthesis since k-CNOT gates are universal.

III. Fault Models

In both traditional and reversible logic the complexity of generating tests for all possible faults in a circuit can be reduced through the use of models which encompass varying fault possibilities. The models vary according to the type of description that is being used, which in turn varies according to the level of abstraction. We list here a number of fault models common to traditional logic, followed by a description of various models used in reversible logic.

A. Traditional Fault Models

In general fault models can be grouped according to the level of abstraction used in describing the system [7]. Levels of abstraction might include behavioural, functional, structural and geometric.

In this work we focus mainly on structural fault models. Structural fault models describe faults that can occur at the gate level and in the interconnections between gates. The gates can be treated as a type of building block in the functional fault model, but then the interconnections are not considered. One of the most commonly used fault models is the stuck-at fault (SAF) model. An interconnection, or line, can be *stuck-at 0* (SA0) if it only can carry a low signal, or similarly it is said to be *stuck-at 1* (SA1) if it can only carry a high signal. The underlying reasons for this type of fault to occur are widely varying, but the model covers a large number of likely problems. It is assumed that a stuck-at fault will occur on only one line in the circuit then this is referred to as the single stuck-at fault model. If multiple faults can occur then the model that is used is the multiple stuck-at fault model.

A similar model can be used at the transistor (or switch) level, but with slightly more detail. For instance, a *stuck-open* fault (SOpF) refers to a transistor which becomes permanently non-conducting, while a *stuck-on* fault (SOnF) refers to a transistor that has become permanently conducting.

We also might have interest in models at the geometric level, where one finds information about line widths, interline widths, and intercomponent distances. For instance, as device sizes shrink there are lower and lower interline widths which is resulting in defects causing shorts where lines inadvertently touch; these are referred to as bridging faults.

B. Fault Models for Reversible Circuits

There are numerous fault models being proposed and used in reversible logic testing. We describe a selection of these in the following sections.

1) Stuck-at Faults: The single stuck-at fault (SSF) and multiple stuck-at fault (MSF) as described for traditional logic are also used in reversible logic testing. The reader will note that Jha and Gupta's notation refers to this model as the SAF model, however the concept is the same for reversible logic as in traditional logic.

2) Crosspoint Faults: Based on the crosspoint model for programmable logic arrays, [8] suggests 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.

3) Missing Gate Faults: The missing gate fault (MGF) model was proposed in [9] and extended in [6]. The single missing gate fault (SMGF) models the complete removal of a gate from the circuit; the multiple missing gate fault (MMGF) models the complete removal of two or more gates; the repeated gate fault (RGF) models repeating the behaviour of a gate within the circuit, and the partial missing gate fault (PMGF) model models the modification of a k-CNOT gate into a k'-CNOT gate where $k' < k$ (in other words, removing one or more controls from the gate). All of these models are based on having circuits composed entirely of k-CNOT gates.

As was the crosspoint model, these models were proposed to address the potential flaws in applying the SAF model to reversible circuits. However the authors also argued that the SAF and other classical fault models are difficult to justify based on proposed quantum gate implementations. Their proposed models were based on observations of how quantum operations would take place. The authors state that "[g]ate operations are pulse-like, localized and microscopic in scale", and "[e]rrors are caused by faults affecting the length, energy, or direction (spatial alignment) of the pulses." Based on these observations the authors go on to identify that "short, missing, misaligned or mistuned gate pulses" would appear as a missing gate, "long or duplicated gate pulses" would appear as a repeated gate, and "partially misaligned or mistuned gate pulses" would appear as a (partially) reduced gate, which led to the models defined above.

4) Bridging Faults: Like the SAF (SSF/MSF) models, bridging fault models are also used in both traditional and reversible logic.

5) Cell Faults: [10] proposed the use of a cell fault model, based on fault modeling for cellular logic arrays [11]. This model assumes that a single gate has failed in some way, which results in undesired modification of any of the lines interacting with the gate. This requires all possible 2^k values to be applied to a $k \times k$ gate in order to detect all possible cell faults [10].

6) Bit Faults: The bit fault model assumes that one or more wires or lines will have their value altered from the correct (fault-free) value to some incorrect value. Technological reasons for this change are not specified, as one assumes that any number of reasons could be the cause for such an alteration. In the single bit fault model it is assumed that a single bit fault is reflected on exactly one wire coming from a gate. This model can also be extended to allow multiple bit faults, as is done for the SSF fault model.

If we assume a permanent bit fault in a gate, then one might also assume that the value of the line in question will *always* be modified by the fault. This makes fault detection fairly easy. However it may be more accurate, depending on the technology, to assume that a fault can occur in a more transient manner, perhaps depending on the inputs applied to the gate. In this case the model behaves as does the cell fault model.

The reader will note below that the bit fault model is used exclusively in online detection approaches, where the test vectors are essentially the input values used in regular operations of the circuit, and thus any bit fault occuring during regular operations is detected without requring a large number of test vectors.

7) Quantum Fault Models: [12] and others have also suggested a number of fault models, approaching this from the quantum paradigm. However the focus of our paper is primarily on structural (gate-level) models, and so we direct the reader to [12] as a comprehensive overview in this area. Briefly, quantum models for faults include initialization faults, where a line (qubit) is incorrectly initialized to some erroneous starting value; bit-flips, where a qubit changes its value at an undesired location/time; and faded control faults, where the control of a gate does not behave as desired. While the SAF, MGF and other models described here could be considered to be at the structural level, these quantum models could be considered to be at the switch level of abstraction. We do, however, point out the overlap with the bit fault model described above.

IV. Testing Approaches

A. Offline approaches

Offline testing assumes that the circuit will be taken out of usual operations and can be tested by applying a number of test vectors for which the correct output values for the circuit are known. Thus a key factor in determining offline testing approaches for a given fault model is the development of test-sets that are *complete* for the model under consideration. A test-set is complete if it detects all faults in the fault set F, and such a set is *minimal* if it contains the fewest possible vectors [10]. Additional circuitry or modification of the circuit is sometimes required, in which case the approach may be referred to as a *design-for-test* (DFT) approach.

1) SSFs and MSFs: [10] addresses the SSF and MSF models, being the first to identify that any test-set that detects all SSFs in a reversible circuit also will detect all MSFs. The authors identify two important properties of reversibility:

- controllability: whether there exists a test vector that generates *any* given desired state on the wires at any given level, and
- observability: the condition that any single fault that changes an intermediate state in the circuit also changes the output.

Both of these properties hold for all reversible circuits, but if fan-out or loss of information is permitted (as in irreversible circuits) then these properties do not hold. This makes testing of reversible circuits much easier than it is for irreversible circuits. An important result from [10] identified the following:

Proposition. *A complete test set [under the SSF model] for an n-wire circuit using the CNT library must have at least $2^{n-1}+1$ vectors, and* any *$2^{n-1}+1$ test vectors will give such a set.*

This is because SSFs are detected by setting each of the wires to both 0 and 1, both before and after each gate (in some works this is described as levels of the circuit).

[13] built on this work, proving that all stuck-at faults in an n-wire k-CNOT circuit with $k \geq 2$ can be detected using at most n test vectors, *and* that the addition of an extra wire that is attached to every k-CNOT gate through a control results in a circuit with a universal test set of size 3 for MSFs. [14] offers further work on test sets for the SSF and MSF models.

2) MGFs: These models were proposed in [6], and this work also provided a number of important results in the area:

Theorem. *For a circuit consisting of N CNOT gates there is always a complete SMGF test set of $\lceil N/2 \rceil$ vectors or less,* and *by adding one extra wire and several 1-CNOT gates, every circuit can be transformed such that the resulting circuit retains its original functionality but has a complete SMGF test set consisting of one test vector.*

RGFs are easily dealt with by identifying that if a gate is replaced by k instances of the gate, then if k is odd the fault cancels itself out, and if k is even the effect is that of the gate being missing which then falls under the SMGF model. The following theorem from [6] deals with PMGFs:

Theorem. *A k-CNOT requires k test vectors to detect all first-order PMGFs and $k+1$ vectors if the SMGF must also be detected.*

The order refers to the number of controls that are removed from the gate, and the SMGF referred to is that created by removing all controls and the target from the gate (*i.e.* the entire gate). MMGFs are harder to deal with, despite the restriction in [6] of the affected gates occuring consecutively. [15], [16] and [17] offer three approaches to generating complete test sets for this type of fault.

3) Cell Faults: [10] also proposes the generation of a test set to cover the cell fault model. Because this model suggests that any of the lines interacting with a gate could be modified, a test set for this model is only complete if the inputs of every $k \times k$ gate in the circuit can be set to all 2^k possible values (assuming the use of the NCT-library for the circuit).

4) Bridging Faults: [18], [19] and [20] focus on test sets for bridging faults in k-CNOT reversible circuits. [19] identifies that if two lines involved in a bridging fault are assigned opposite logic values then the error will be propogated to the output(s) of the circuit. This provides a basis for developing a test set. [20] extends the model to include single input bridging faults (SIBF), which are faults involving only two inputs lines of the circuit, as well as multiple input bridging faults (MIBF) which involve more than two lines. [20] also proves that a complete test set for the SIBF model is complete for the MIBF model, and that n test vectors are sufficient for detecting all single and multiple input bridging faults *as well as* all stuck-at faults for an n wire reversible circuit. [18] offers a DFT approach by adding an extra input line/wire, similar to the approach in [10].

5) Crosspoint Faults: [8] identified that all single crosspoint faults are detectable, and that the number of gates in a circuit is an upper bound on the size of a complete test set for all single appearance crosspoint faults in the circuit. The authors also suggest a randomized automatic test pattern generation (ATPG) technique for generating complete test sets for all single crosspoint faults.

B. Online approaches

Online testing involves testing approaches that can be assessed while the circuit is operating normally. This may involve the addition of circuitry to enable the detection of faults while the circuit is being used. While offline testing approaches assume the presence of a permanent fault of some type, online testing may be able to detect both permanent and transient faults.

1) Bit faults: Numerous online testing approaches have focused on the bit fault model, including [21], [22], [23] and [24]. We hypothesize that one reason for its popularity is the unknown nature of the technologies to be used in implementing reversible circuits. While elementary particle spins may be a commonly theorized implementation approach [6], it is by no means the only option; thus a fault model that allows for flexibility may be desired. [24] suggests an approach involving the addition of one line and several gates to a NCT-based circuit in order to detect all single bit faults. [21] proposes an approach using testable gates that are then cascaded together to implement the required functionality; and [22] suggests a similar approach, although with different building blocks. [23] also offers an approach based on cascading together testable gates, however their approach also covers multiple bit faults.

2) SAFs: [25] presents a method that detects both stuck-at faults and single bit faults (referred to as bit flips), as well as some multiple bit faults. The approach is to replace all gates, including Toffoli, Peres, and Fredkin, with testable versions, and then a parity-checking technique is used.

3) MGFs and RGFs: The work in [26] focuses on missing gate and repeated gate faults. Briefly, information on the *number* of gates that exist (or should exist) in the circuit is maintained, and this can be used to provide information on whether a gate is missing or being repeated, and a new gate is proposed in order to build a circuit with this detection line built into every gate.

[27] also proposed an online-testing technique for detecting single missing gate faults (SMGFs), this time in k-CNOT-based circuits. The testability is achieved by adding an extra line and replacing each gate in the cascade with 4 gates. We note that [24] built on this method to offer an alternative approach that addresses the single bit fault model.

C. Fault Tolerance

We briefly address the issue of fault tolerance, although this is not the focus of this paper. Fault tolerance in circuits is a different approach than testing and testability; in this case the goal is to assume that there is some resilience to error built in to the circuit. [28], for instance suggests a reversible approach to this. In practice, however, some works using the term “fault tolerant” are reporting on approaches similar to what we have described previously, providing primarily designs that require a human to check some value at the end of the circuit’s computation to ensure that the computation took place without error. Works that could be considered to fall into this category include [29], [30], [31] and [32], as each suggests the use of parity-checking or a similar approach to ensure “tolerance” of faults.

V. Connections

There are a number of connections among the various types of fault models and testing approaches. These are useful to highlight, as it allows for the possibility that approaches targeted at one particular model can be applied to detecting faults under another model. In particular we note that many authors have begun in this area with a SAF or bit fault detection approach; thus we begin with these models in order to demonstrate their capability (or lack thereof).

A. SAFs and other models

a) SAF and Crosspoint faults models: [8] has identified that a complete test set for all single crosspoint faults also covers most SSFs. At least $(2n+k-1)$ of a possible $2(n+k)$ SSFs are covered (where n is the number of inputs and k is the number of gates in the circuit). However the authors show that a test set for the SAF model will not necessarily detect a crosspoint fault, as shown in Figure 3, and demonstrate experimentally that a complete test set for SSFs often covers less than half of the single crosspoint faults.

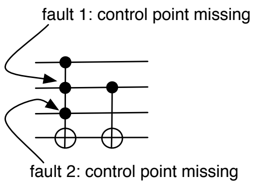

fault 1: control point missing
fault 2: control point missing

Fig. 3. Although the test set $\{0110, 1001, 1000\}$ is complete for all stuck-at faults in this circuit, it does not detect faults 1 and 2.

This can be explained through the following analysis: a test set is complete under the SSF model if each of the wires at every level (prior to each gate) can be set to both 0 and 1 by the test set [10], while appearance crosspoint faults are detected by setting the control inputs to logic 1 and other inputs (the target) to logic 0. Thus it is possible in many cases to contruct a test set which covers all SSFs and which will also satisfy the requirement to test all single appearance crosspoint faults. However disappearance crosspoint faults are tested by setting the controls to logic 1 with the exception of the control which is being tested for disappearance, which must be set to logic 0 (values for the other lines are not important). For instance, to test for disappearance of all possible controls in just the first gate in Figure 3 we require the test vectors $\{011-, 101-, 110-\}$ ($-$ indicates the value can be either logic 1 or logic 0) while the SSA faults could be tested by just two vectors (assuming only a single gate).

b) SAF and MGF models: As described above, SAFs are detected by setting the wires to both logic 0 and logic 1 at each level of the circuit. However MGFs are detected by setting all control bits for each gate to logic 1. Thus we can see that a test set for SAFs could potentially be designed to cover MGFs; future work is planned in this area.

c) SAF and Bit fault models: Bit faults and SAFs are similar in nature, although bit faults are dependent on the previous value of the line. The main difference is that a bit fault, if present, is assumed to always change a line’s value. This is not true of a SAF. Thus if we assume that a permanent bit fault in a gate will always modify the faulty line in question, then if a gate and/or wire exhibits permanent bit faults these faults (at each level) can be detected by the application of a single test vector, much like MGFs, while SAFs require a minimum of two test vectors. Again, we can see that a test set for SAFs could (would be likely to) cover bit faults.

d) SAF and Bridging faults models: According to [33], bridging faults between two lines at the same level can be detected by test vectors that set the two lines to opposite logic values, *i.e.* $\{01\}$ or $\{10\}$. Again, we see that a SAF model, which requires two test vectors to detect faults at any level, could potentially cover bridging faults as well, if designed carefully, and in fact [20] demonstrates this.

e) Cell faults and SAF models: As stated above, in order to detect all possible cell faults all 2^k values must be applied to the $k-1$ controls and the target of a $k \times k$ gate at every level. It is trivial to see that this will cover all SAFs as well.

B. Bit faults and other models

Under the bit fault model we assume that a line or wire's value will, if a permanent fault is present, always change to an incorrect value. Thus setting any value on that line will identify the fault. This means that any test set that specifies values for every line will cover all faults under the bit fault model.

As the reader will note, the bit fault model has been used exclusively in online testing approaches, implying that the development of a test set to be used in an off-line setting is either a) not useful or b) trivial. In fact, both are the case. The triviality of developing a test set for bit faults is explained above. To discuss the lack of usefulness of such an approach we first discuss the cell fault model. As mentioned above, a $k \times k$ gate must have all 2^k values applied to its inputs to detect all possible cell faults. This implies that there are some input combinations which will result in (the appearance of) a fault on an output line while others (potentially) do not. If we assume that, rather than exhibiting permanent bit faults, a circuit might exhibit transient bit faults, then we have a situation where a given test vector may not identify a bit fault. That is, when a test vector is applied to the circuit, a bit fault may not be present, or may not "triggered" by that value. This theory is supported by, for instance, the description in [9] of potential quantum circuit technologies:

> These essentially static states are modified by dynamic electromagnetic (EM) pulses that implement gate functions like CNOT. For example, in trapped-ion technology, qubits are individual atoms whose electric charge states are altered by directing laser pulses of precise frequency and duration at them under control of a (classical) computer. ... Thus in the quantum case, the "gates" ... often represent EM pulses, while the "wires" indicate the order in which the gate operations are applied.

It is not at all unlikely for the EM pulses to be erratic or inconsistent in behaviour, resulting in bit faults appearing on a transient basis. This argues for an online testing approach, and as well for a generic fault model such as the bit fault model that would cover any type of potential error.

C. Connections between other models

a) Crosspoint and PMGF models: The PMGF model is a subset of the crosspoint fault model. The crosspoint model allows for either disappearance or appearance of any of the control points on a reversible gate, where disappearance faults are in fact the same as PMGFs [8]. Thus the approach from [8] trivially covers all PGMFs.

b) Crosspoint and other MGF models: In order to detect other MGFs it is necessary to apply a logic 1 to all control inputs of the faulty gate [6]. To test for appearance crosspoint faults we apply logic 1 to all control lines of the gate and logic 0 to the remaining lines. Thus we can see that a test set for appearance faults would also cover MGFs (other than PMGFs), and that it would be possible to design a test set for MGFs that would also cover appearance faults. Disappearance faults are detected by applying logic 1 to all controls except the control to be tested, which must have a logic 0 applied. This would be more difficult to incorporate into a test set for MGFs.

c) Bridging fault and other models: Bridging faults are tested by assigning the two lines assumed to be involved in a bridging fault opposite logic values. As described above, testing for disappearance faults requires a similar approach. As a reminder, we saw that to test for disappearance of all possible controls for the first gate in Figure 3 we required the test vectors $\{011-, 101-, 110-\}$. Thus this test set would also test for bridging faults between several of the lines in the circuit, and additional faults could be detected by careful selection of the value for the don't cares in this vector. However connections between the bridging fault and the MGF models are not apparent.

VI. Conclusion

While the idea of reversible logic has existed for some time, researchers are only now becoming interested in the idea. Thus significant work is now being put into determining efficient ways to synthesize and test reversible circuits. Fault models for reversible logic first tended to borrow from traditional logic, with models such as the stuck-at model, but as the area has matured we are seeing models specific to potential quantum technologies appear. In this paper we have provided an overview of existing work in this area and offered a survey of the testing approaches utilising the various fault models. We have drawn parallels between the available models, showing how there is some overlap among models such as the crosspoint (disappearance) fault model and the partial missing gate fault (PMGF) model, and as well provided preliminary suggestions as to how these and other similarities can be leveraged. In particular, we have clearly differentiated between the offline and online testing research in this area, and we point out that certain fault models, such as the bit fault model, are much better suited to online testing approaches.

References

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

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

[3] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 6, pp. 525–532, 1973.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

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

[6] I. Polian, J. P. Hayes, T. Fiehn, and B. Becker, "A family of logical fault models for reversible circuits," in *Proceedings of the 14th Asian Test Symposium (ATS)*, 8–21 Dec., Calcutta, India, 2005, pp. 422–427.

[7] N. K. Jha and S. Gupta, *Testing of Digital Systems*. Cambridge University Press, 2003.

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

[9] J. P. Hayes, I. Polian, and B. Becker, "Testing for missing-gate faults in reversible circuits," in *Proceedings of the 13th Asian Test Symposium*, ser. ATS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 100–105. [Online]. Available: http://dx.doi.org/10.1109/ATS.2004.84

[10] K. N. Patel, J. P. Hayes, and I. L. Markov, "Fault testing for reversible circuits," in *Proceedings of the IEEE VLSI Test Symposium (VTS)*, Napa, CA, April, 2003, pp. 410–416.

[11] W. H. Kautz, "Testing for faults in combinational cellular logic arrays," in *Proceedings of the IEEE Conference Record of the Eighth Annual Symposium on Switching and Automata Theory (SWAT)*, 1967, pp. 161–174.

[12] J. D. Biamonte, J. S. Allen, and M. A. Perkowski, "Fault models for quantum mechanical switching networks," *Journal of Electronic Testing, Theory and Applications (JETTA)*, vol. 26, no. 5, pp. 499–511, 2010.

[13] A. Chakraborty, "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*, 2005, pp. 249–254.

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

[15] X. Fang-ying, C. Han-wu, L. Wen-jie, and L. Zhi-giang, "Fault detection for single and multiple missing-gate faults in reversible circuits," in *Proceedings of the IEEE World Congress on Computational Intelligence, IEEE Congress on Evolutionary Computation*, Jun. 2008, pp. 131–135.

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

[17] D. Kole, H. Rahaman, D. Das, and B. Bhattacharya, "Derivation of optimal test set for detection of multiple missing-gate faults in reversible circuits," in *Proceedings of the 19th IEEE Asian Test Symposium (ATS)*, Dec. 2010, pp. 33–38.

[18] M. Bubna, N. Goyal, and I. Sengupta, "A DFT methodology for detecting bridging faults in reversible logic circuits," in *Proceedings of TENCON – IEEE Region 10 Conference*, 2007, pp. 1–4.

[19] H. Rahaman, D. Kole, D. Das, and B. Bhattacharya, "Optimum test set for bridging fault detection in reversible circuits," in *Proceedings of the 16th Asian Test Symposium (ATS)*, Oct. 2007, pp. 125–128.

[20] P. Sarkar and S. Chakrabarti, "Universal test set for bridging fault detection in reversible circuit," in *Proceedings of the 3rd International Design and Test Workshop (IDT)*, Dec. 2008, pp. 51–56.

[21] D. P. Vasudevan, P. K. Lala, J. Di, and J. P. Parkerson, "Reversible-logic design with online testability," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 2, pp. 406–413, Apr. 2006.

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

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

[24] N. M. Nayeem and J. E. Rice, "Online fault detection in reversible logic," in *Proceedings of the 26th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, Vancouver, BC, Canada, Oct. 2011, pp. 426–434.

[25] B. Sen, J. Das, and B. Sikdar, "A DFT methodology targeting online testing of reversible circuit," in *Proceedings of the International Conference on Devices, Circuits and Systems (ICDCS)*, Mar. 2012, pp. 689–693.

[26] M. Zamani, N. Farazmand, and M. Tahoori, "Fault masking and diagnosis in reversible circuits," in *Proceedings of the 16th IEEE European Test Symposium (ETS)*, May 2011, pp. 69–74.

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

[28] M. Zamani and M. Tahoori, "Online missing/repeated gate faults detection in reversible circuits," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2011, pp. 435–442.

[29] B. Parhami, "Fault-tolerant reversible circuits," in *Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Oct. 29–Nov. 1 2006, pp. 1726–1729.

[30] M. S. Islam, M. M. Rahman, Z. Begumand, and M. Z. Hafiz, "Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders," in *Proceedings of the International Conference on Advances in Computational Tools for Engineering Applications*, 2009, pp. 296–401.

[31] M. Haghparast and K. Navi, "Design of a novel fault tolerant reversible full adder for nanotechnology based systems," *World Applied Sciences Journal*, vol. 3, no. 1, pp. 114–118, 2008.

[32] S. K. Mitra, L. Jamal, M. Kaneko, and H. M. Hasan Babu, "An efficient approach for designing and minimizing reversible programmable logic arrays," in *Proceedings of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '12. New York, NY, USA: ACM, 2012, pp. 215–220. [Online]. Available: http://0-doi.acm.org.darius.uleth.ca/10.1145/2206781.2206834

[33] M. Abramovici, M. A. Breur, and A. D. Friedman, *Digital Systems Testing and Testable Design*. IEEE Press, 1995.