A Reversible Majority Voter Circuit and Applications

Md. Asif Nashiry and Jacqueline E. Rice
Department of Math and Computer Science
University of Lethbridge
Alberta, Canada
Email: [asif.nashiry, j.rice] @uleth.ca

***Abstract*—Reversible logic is becoming a more and more attractive option to traditional logic, due to its potential to reduce heat dissipation. However traditional circuit design techniques can not always be translated to reversible circuits, and so new techniques for *e.g.* fault tolerance must be developed. We present a design for a reversible majority voter circuit which has applications not only in the development of fault tolerant reversible circuits, but also in areas such as data mining and classification.**

I. Introduction

In 1961 R. Landauer showed that $KTln2$ joules are dissipated each time an information bit is lost during a logical operation, where K is Boltzmann's constant and T is the operating temperature in Kelvin [1]. For instance, when a two-input AND gate produces a single bit of output, this amount of energy is dissipated as heat. As Moore's law [2], predicting a doubling of components every few years, has held true over the last several decades, this heat dissipation is becoming a major concern in traditional irreversible systems. However reversible systems are, by definition, bijective, and thus there is no information loss. Other reasons for research into reversible systems includes connections to quantum computing [3], and applications in cryptography [4], nano-computing technologies [5] and digital processing [6].

Since a reversible circuit maintains a one-to-one relationship between inputs and outputs, achieving fault tolerance in such a system is not an easy task. Fault tolerance is defined as an attribute of a system that enables the system to correctly perform its specified operations even in the presence of faults. In order to make a system fault tolerant, it is necessary to build in redundancy of some type, generally hardware redundancy, software redundancy, information redundancy and/or timing redundancy [7]. This added redundancy comes at a cost. A common approach to fault tolerance is to incorporate hardware redundancy by replicating one or more physical components of a system. The cost of this is initially high, but is ammortized over the lifetime of the system. Hardware redundancy can offer an active approach, a passive approach or a combination of both [7]. An active approach to fault tolerance works by detecting a fault, locating the fault and recovering the system through some form of reconfiguration. However fault tolerance can also be achieved using a passive approach that does not require detecting or reconfiguring, rather masking the occurrence of faults. Masking contributes to a fault tolerant system by hiding faults from the final outcome of the system. Thus the fault may affect the system locally, but does not affect the global performance of a system. A majority voter is commonly used to achieve fault tolerance in traditional systems [7] by implementing triple modular redundancy (TMR) or N-modular redundancy. The basic idea of TMR is to triplicate the hardware and use a majority voter that determines the final output by observing the outputs from all the modules. If one of the modules become faulty, the majority voter can mask the faulty outcome by observing the outcomes of the remaining modules and correcting the faulty signal.

This paper presents designs for a majority voter to be implemented using reversible logic. The proposed designs can be used in order to build fault tolerant reversible circuits and also can be used in the field of machine learning, as will be described in Section IV.

II. Background

A. Reversible Functions, Gates, & Circuits

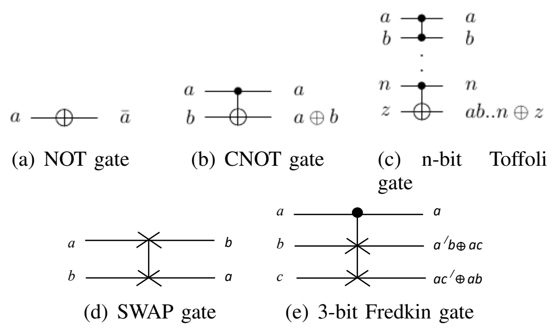
Definition 1. *A multiple output Boolean function $f(x) : B^m \rightarrow B^n$ is reversible if it is bijective.*

If B is a finite set and $f(x) : B^m \rightarrow B^n$ is a Boolean function which maps each input vector to a unique output vector (bijection) then $f(x)$ is reversible. Each reversible function is mapped on to a reversible circuit using reversible gates.

Definition 2. *A reversible gate computes a reversible function.*

Let $X := \{x_1, ..., x_n\}$ be the set of Boolean variables. Then a reversible gate has the form $g(C, T)$, where $C = \{x_{i1}, ...x_{ik}\} \subset X$ is the set of control lines and $T = \{x_{j1}, ..., x_{jl}\} \subset X$ with $C \cap T = \phi$ is the set of target lines [8]. Two commonly used reversible gates are Toffoli gates and Fredkin gates. A Toffoli gate with no controls is a NOT gate i.e. $g(0, x_{j1})$. Similarly, a Toffoli gate $g(x_{i1}, x_{j1})$ can be thought of as a controlled NOT (or CNOT) gate, and $g(\{x_{i1}, ..x_{in}\}, x_{j1})$ is a n-bit Toffoli gate.

A Fredkin gate with no controls is a SWAP gate $g(x_{j1}, x_{j2})$, which interchanges the two target input bits at output. A n-bit positive control Fredkin gate $g(\{x_{i1}, ..x_{in}\}, x_{j1}, x_{j2})$ interchanges the two target bits at output when all the control inputs are equal to 1. A reversible gate may also have negative



(a) NOT gate (b) CNOT gate (c) n-bit Toffoli gate (d) SWAP gate (e) 3-bit Fredkin gate

Fig. 1. Several commonly used reversible gates.

control. In this case the gate becomes active when negative control has a value of 0.

Figure 1 shows the symbols and notation for several reversible gates. Two important metrics used to compare reversible circuit implementations are gate count and quantum cost. The gate count (GC) is the number of gates in a circuit and the quantum cost (QC) is the number of basic quantum gates required to implement macro-level reversible gates such as the Toffoli and Fredkin gates [9], [10].

Definition 3. *A network of reversible gates forms a reversible circuit and implements a reversible function.*

An example reversible circuit is shown in Figure 2. For each gate, the control lines are those that affect the functionality of the gate, while the target is the line whose value may be changed by the computation carried out by the gate.

Another important metric in the design of reversible circuits is that of garbage, or ancillary lines. These are lines that are necessary to maintain the reversibility of the function but do not produce any desired functionality. For instance, Figure 2 illustrates a reversible full adder [11], where the desired functionality can be observed on lines C and S. Lines g_1 and g_2 are necessary to maintain the reversibility, but do not produce any desired functionality. Further discussion of garbage lines can be found in *e.g.* [12], [13].

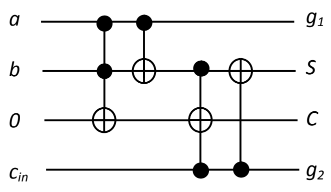


Fig. 2. A reversible full-adder circuit.

III. Three-input Reversible Majority Voter

A. Previous Work

There have been only a few attempts in the literature to design majority voter circuits in reversible logic. In [14], authors proposed two designs based on Triplicated Modular Redundancy. The circuit takes three inputs from the output of three copied circuits and generate threes data outputs. In order to maintain reversibility the voter is a 5-bit circuit with two garbage lines and two constant input lines. The design goal is to produce all 0 or all 1 on the three data outputs, thus masking any faulty output. However our analysis suggests that the design presented in [14] does not generate the intended corrected output.

A simplifed majority voter circuit as shown in Figure 3 is presented in [15]. The authors describe a reversible fault tolerant multiplexing scheme using a 3-bit repetition code. The voter consists of two CNOT gates and one Toffoli gate, and the majority output value is taken from the line labeled a_o.

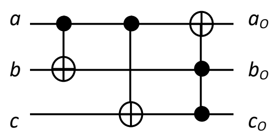


Fig. 3. A reversible three input majority voter as proposed in [15].

B. Proposed Design

In this paper we present a simplified and cost effective approach for designing a majority voter in reversible logic. To introduce our approach, a reversible three input majority voter is shown in Figure 4. The behaviour of this circuit is characterized by the truth table in Table I. The output of

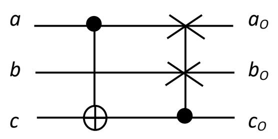


Fig. 4. A three input reversible majority voter.

TABLE I
Truth table for the reversible voter circuit.

abc	after 1st gate $a'b'c'$	after 2nd gate $a_ob_oc_o$
000	000	000
001	001	001
010	010	010
011	011	101
100	101	011
101	100	100
110	111	111
111	110	110

interest is a_o, which always gives the value held by majority of the input lines. The other two lines are not useful, and thus are considered garbage. The quantum cost of this circuit is very low: 1 for the CNOT gate and 5 for the Fredkin gate, for a total of 6, which is an improvement in design as compared to the approach presented in [15] with a gate count of 3 and a quantum cost of 7. As we see from the figure, the control of the Fredkin gate is $(a \oplus c)$. When $a = c$ then $(a \oplus c) = 0$ and the Fredkin gate will be inactive and will not interchange the values of a and b at the output. In this case a is the value used as the majority output. However when $a \neq c$ then$(a \oplus c) = 1$

and the Fredkin gate will be active and swap the value of a and b.. In this case, the value of b will be the majority output.

IV. Applications & Related Work

A. Fault Tolerance in Reversible Circuits

Although the concepts of fault tolerance and testing are often confused, they are not the same. For instance, works on testing such as [16] and [17] present approaches that allow the circuit to be tested for faults, often using parity preservation. However this does not provide fault tolerance, as defined earlier. We provide here an overview of some works that address fault tolerance in reversible logic.

Parity Preservation vs Fault Tolerance: Many works in the literature that addresses fault tolerant circuit designs in reversible logic use the term *fault tolerant* when really they are referring to testing. Works that fall into this category include [18], [11], [19], [20] and [21]. For instance, in [18] the authors propose a 4-bit parity preserving reversible gate referred to as an IG gate. The authors present an implementation of a reversible full adder circuit with two IG gates and claim that their proposed design is fault tolerant, suggesting that fault tolerance can be achieved without any extra design effort if a reversible circuit is built using parity preserving gates. Similarly in [20], the authors present a synthesis of a parity preserving Toffoli gate. Since a Feynman gate is already a parity preserving gate, the authors used two of their proposed parity preserving Toffoli gates and two double-Feynman gates to design what they claim is a fault tolerant full adder circuit. Fault tolerance is a property which enables a system to continue operating properly and generating the intended (correct) output even in the event of a failure of some of system's components. In this event a fault tolerant circuit must have the capability to supply corrected (intended) bits at the output. A parity preserving circuit does not guarantee that the circuit is fault tolerant, since their use of parity preservation offers only error detection. Thus these designs cannot be categorized as offering fault tolerance. Other works that indeed fall into the category of fault tolerance were described previously in Section III. As we present below, our proposed majority voter can be used to build a reversible fault tolerant circuit with the capability of masking a fault in a single output bit.

Fault Tolerant Reversible Full Adder: A fault tolerant full adder design based on our proposed majority voter is shown in Figure 5. Since we have two outputs (sum and carry) of interest in a full adder, we need two majority voters to ensure that faults in either output can be masked. As we see from Figure 5 there are three carry lines that are connected to the voter on top, while the bottom voter is connected to the three the sum lines. This design can generate corrected output in the presence of faults from any of the proposed fault models ([22], [23]) as long as the fault affects at most one of the triplicated full adders. Similarly, this design can be applied to any type of circuit, albeit with a significant amount of hardware overhead.

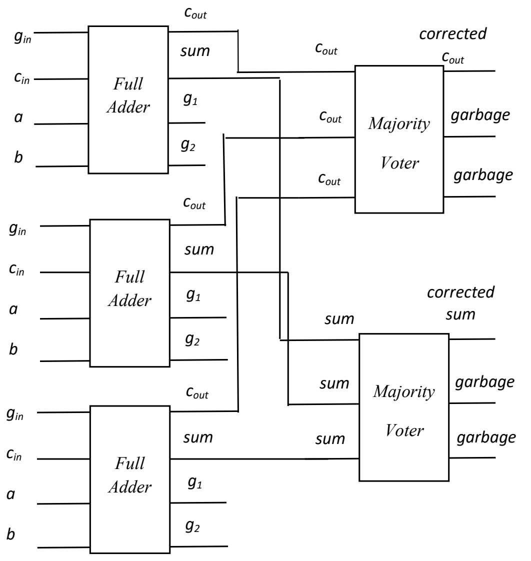


Fig. 5. A fault tolerant reversible full-adder circuit design.

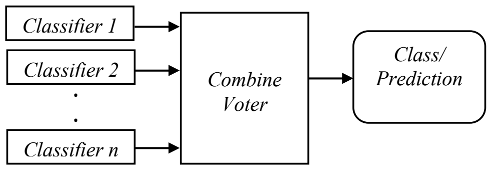


Fig. 6. Ensemble method in classification/prediction.

B. Machine Learning

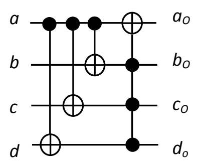
Classification and prediction are two widely used forms of data analysis in machine learning. Classification and prediction have many application areas such as medical diagnosis, performance prediction, forecasting, target marketing, and fraud detection [24]. The first step of a classification process is to build a classifier by learning from training data set. There are various methods which can be used to build a classifier such as decision trees, Bayesian classification, rule based classification, and genetic algorithms. The next step is to test the classifier for accuracy by using a test data set. The accuracy of a classifier refers to the ability of a classier to correctly predict the class label of a given data set. Ensemble methods are used in order to achieve a high degree of accuracy. The principle idea of an ensemble method as shown in Figure 6 is to combine n classifier models for creating an improved composite model. Bagging and boosting are two such ensemble techniques [24]. Bagging and boosting generate a set of n classifiers, and a voting strategy can be used to combine the n classifiers and generate a more accurate outcome. The voter in Figure 4 takes three inputs, so this voter can combine results from three classifiers. However, when the number of classifiers increases, we need to extend the design of the majority voter. The next section provides the design of a n-bit majority voter.

V. Extension of Reversible Majority Voter

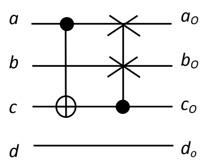
Although an odd value of n is desirable for a n-bit voter, a voter with an even number of inputs can sometimes be required in different applications, for example in a machine learning application where the goal is to predict an outcome from an even number of observations. Here we present designs for both odd and even bit majority voters.

First we consider the design of a n-bit majority voter where n is even. When n is even there are input combinations for which there is no majority value; that is, the number of lines/input bits carrying 0 and the number of lines carrying 1 are equal. In such cases, we can consider both values to be majority, or neither to be the majority, depending on the requirements of the particular application. One might refer to this as a "tie". For a instance, when we have a 4-bit input of $(0, 1, 1, 0)$, the majority voter circuit could send either a 1 or 0 to the final output. For our design we consider this to be a don't care condition.

Figure 7(a) shows a design of a 4-bit reversible majority voter consisting of three CNOT gates and one 4-bit Toffoli gate. The output that reflects the majority is labeled a_o. The other three outputs are non-functional outputs. The GC and QC of this 4-bit majority voter is 4 and 16 respectively. However, it is possible to reduce these costs by using the functionality of a 3-bit majority voter.



(a) A 4-bit majority voter.



(b) A 4-bit majority voter designed from a 3-bit voter.

Fig. 7. Two 4-bit reversible majority voters.

Lemma: For an even n, a $(n-1)$-bit majority voter is sufficient to determine the majority of n bits, assuming that "ties" are treated as don't cares.

Proof: Let x be a Boolean variable and $\{y_0, \dots, y_{n-1}\}$ be the n Boolean inputs to a n-bit majority voter where n is even. Also, let l be the number of bits in $\{y_0, \dots, y_{n-1}\}$ that take on the value x. A sufficient condition for x to have the value of the majority of bits in $\{y_0, \dots, y_{n-1}\}$ is: $\lceil n/2 \rceil \leq l$ which is also a condition to determine the majority bit in a combination of bits $\{y_0, \dots, y_{n-2}\}$ *i.e.* for the next lower (odd) value of n.

Figure 7(b) shows a design for a 4-bit majority voter. A 3-bit majority voter works on the input bits a, b and c, and supplies the majority bit of the four input bits to the final output a_o. The input bit d is passed to d_o unchanged. Table II shows the behaviour of the 4-bit majority voter from Figure 7(b). The design cost is the same as that of a 3-bit voter with a GC of 2 and QC of 6. We see also that the majority among the first three bits serves as the final majority bit of all four bits, and so in fact, the fourth bit has no effect in determining the final outcome.

TABLE II
Truth table of the 4 bits majority voter shown in Figure 7(b).

$abcd$	$a_ob_oc_od_o$	
0000	0000	
0001	0001	
0010	0010	
0011	0011	don't care
0100	0100	
0101	0101	don't care
0110	1010	don't care
0111	1011	
1000	0110	
1001	0111	don't care
1010	1000	don't care
1011	1001	
1100	1110	don't care
1101	1111	
1110	1100	
1111	1101	

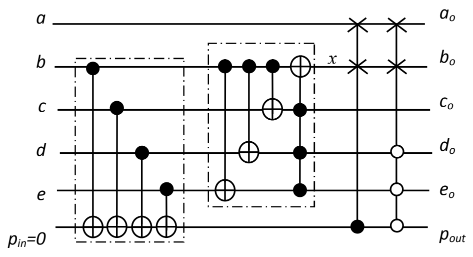
It is similarly possible to design a 6-bit voter by simply including a sixth line to a 5-bit majority voter. Figure 8(b) shows such a design. A 4-bit majority voter can be used in designing a 5-bit voter. The dashed box on right side of a 5-bit voter in Figure 8(a) shows a 4-bit majority voter. A constant input p_{in} is initialized to 0. A CNOT gate is connected from each of the inputs of the 4-bit voter to the parity line p_{in}. The 4-bit voter manipulates the four bits $(b, c, d$ and $e)$ and sends the majority of these four bits to the point labled x. The value at x indicates three cases based on the number of appearance of x in the four input bits: two, three or four times of appearance.

Case 1: When the value of x appears twice in $(b, c, d$ and $e)$, it indicates one of the don't care conditions and hence the fifth bit at the line labled a will be the final majority bit. A group of CNOT gates as shown in the dashed box in Figure 8(a) determines the parity of the four bits $(b, c, d$ and $e)$. In a don't care condition the parity bit is $(x \oplus \bar{x} \oplus x \oplus \bar{x}) = 0$. In this case, the 3-bit Fredkin gate in Figure 8(a) will be inactive, so the bit at a will provide the majority bit at output a_0. For example when $(a, b, c, d, e) = (0, 1, 0, 0, 1)$, the 4-bit voter sends 1 to the point labled x in Figure 8(a). The CNOT gates block calculates the parity $(1 \oplus 0 \oplus 0 \oplus 1) = 0$. A 0 on parity line will make two Fredkin gates inactive, hence the Fredkin gates will not interchange the value of a and x at the output. The bit at $a(=0)$ will go the output line a_o as the majority bit.

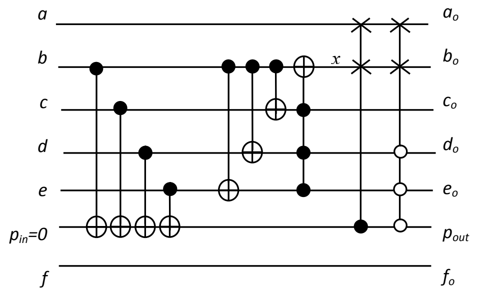
Case 2: When the value of x appears three times in the 4-bit inputs, the value of x serves as the final majority bit of the 5-bit voter. In this case, the parity line $p_{in} = (x \oplus x \oplus x \oplus \bar{x}) = 1$. Thus the 3-bit Fredkin gate becomes active and interchanges the value of a and x. The value of x goes to the final output a_o. For example, when $(a, b, c, d, e) = (1, 1, 0, 0, 0)$, the 4-bit voter supplies 0 to x . In this case the output of the CNOT gates block is $(1 \oplus 0 \oplus 0 \oplus 0) = 1$, which activates the 3-bit positive control Fredkin gate. Thus the 3-bit Fredkin gate swaps the values of $a(=1)$ and $x(=0)$, and 0 goes to the final output line a_o as the majority bit.

Case 3: The last 5-bit negative control Fredkin gate is included in Figure 8(a) to supply the majority bit at the output when all four inputs of the 4-bit voter are the same and the fifth input line of a 5-bit voter has an opposite bit. For example, when $(a, b, c, d, e) = (0, 1, 1, 1, 1)$, the output of the 4-bit voter is $((x = 1), 0, 0, 0)$ and the parity line is $(1 \oplus 1 \oplus 1 \oplus 1) = 0$. In this case, the 3-bit Fredkin gate remains inactive. However the three 0s on three lines labled p_{in}, d and e activate the 5-bit negative control Fredkin gate. The 5-bit Fredkin gate swaps $a(=0)$ and $x(=1)$, thus the circuit sends 1 to the output line a_o.

The GC and QC of a 5-bit voter is 10 and 54 respectively. We can use the 5-bit voter to design a 6-bit voter by simply including the sixth line with the same design cost as shown in Figure 8(b). In this way it is possible to extend a majority voter circuit by building on designs for smaller voters.



(a) A 5-bit reversible majority voter



(b) A 6-bit reversible majority voter

Fig. 8. Two reversible majority voters.

VI. Conclusion & Future Work

This paper presents a 3-bit reversible majority voter circuit. The purpose of this circuit is to identify the bit value with appears more than any other bit value on the three input bits, assuming the use of Boolean (binary) values. Our proposed design is simpler and of lower cost in terms of gate count and quantum cost than existing designs in the literature. We also provide the designs for extending the voter from 3-bits to higher order bits. Moreover, we demonstrate two applications for our voter, in the areas of machine learning and fault tolerant reversible circuit design. We provide an overview and analysis of existing works that term themselves to be fault tolerant, but which do not meet the required characteristics to be categorized as such. Lastly, we present a design for a fault tolerant reversible full adder, the approach to which can be extended to any reversible circuit. The proposed majority voter can be used to generate a corrected output in the presence of any type of fault as long as the fault affects a minority of the n input lines to the voter. Future work includes improving the robustness of the majority voter and developing techniques for fault location as well as fault correction.

Acknowledgment

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

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.
[2] R. R. Schaller, "Moore's law: Past, present and future," *Spectrum, IEEE*, vol. 34, no. 6, pp. 52–59, 1997.
[3] T. Hey, "Quantum computing: An introduction," *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 105–112, 1999.
[4] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Reversible logic circuit synthesis," in *Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*. ACM, 2002, pp. 353–360.
[5] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, no. 1, p. 21, 1993.
[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.
[7] B. W. Johnson, *Design & Analysis of Fault Tolerant Digital Systems*. Addison-Wesley Longman Publishing Co., Inc., 1988.
[8] T. Toffoli, "Reversible computing," MIT, Tech. Rep. MIT/LCS/TM No. 151, 1980.
[9] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computations," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, Nov 1995.
[10] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum fredkin gate," *Physical Review A*, vol. 53, no. 4, pp. 2855–2856, 1996.
[11] 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.
[12] R. Wille, M. Soeken, and R. Drechsler, "Reducing the number of lines in reversible circuits," in *Proceedings of the Design Automation Conference (DAC)*, 2010, pp. 647–652, 13–18 June, Anaheim, California.
[13] D. M. Miller, R. Wille, and R. Drechsler, "Reducing reversible circuit cost by adding lines," in *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, 2010, pp. 217–222, 26–28 May, Barcelona, Spain.
[14] M. Zamani, N. Farazmand, and M. B. Tahoori, "Fault masking and diagnosis in reversible circuits," in *European Test Symposium (ETS), 2011 16th IEEE*. IEEE, 2011, pp. 69–74.
[15] P. O. Boykin and V. P. Roychowdhury, "Reversible fault-tolerant logic," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005, pp. 444–453.
[16] M. A. Nashiry, G. G. Bhaskar, and J. E. Rice, "Online testing for three fault models in reversible circuits," in *Multiple-Valued Logic (ISMVL), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 8–13.
[17] 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. 55, no. 2, pp. 406–414, 2006.
[18] M. S. Islam, M. M. Rahman, Z. Begum, M. Z. Hafiz, and A. Al Mahmud, "Synthesis of fault tolerant reversible logic circuits," in *Testing and Diagnosis, 2009. ICTD 2009. IEEE Circuits and Systems International Conference on*. IEEE, 2009, pp. 1–4.

[19] 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.

[20] 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.

[21] 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

[22] I. Polian, T. Fiehn, B. Becker, and J. P. Hayes, “A family of logical fault models for reversible circuits,” in *Test Symposium, 2005. Proceedings. 14th Asian*. IEEE, 2005, pp. 422–427.

[23] J. Rice, “An overview of fault models and testing approaches for reversible logic,” in *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*. IEEE, 2013, pp. 125–130.

[24] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Elsevier, 2011.