Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 11, No. 2, June 2023, pp. 416~430
ISSN: 2089-3272, DOI: 10.52549/ijeei.v11i2.4184

Testable Design for Positive Control Flipping Faults in Reversible Circuits

Mousum Handique[1], Hiren Kumar Deva Sarma[2]
[1]Department of Computer Science and Engineering, TSSOT, Assam University, Silchar, Assam, India
[2]Department of Information Technology, Gauhati University, Guwahati, Asaam, India

Article Info

Article history:

Received Sep 27, 2022
Revised Mar 23, 2023
Accepted Apr 26, 2023

Keywords:

Reversible logic circuit
Test vector
Testable design circuit
Parity-test pattern
Positive control flipping faults

ABSTRACT

Fast computational power is a major concern in every computing system. The advancement of the fabrication process in the present semiconductor technologies provides to accommodate millions of gates per chip and is also capable of reducing the size of the chips. Concurrently, the complex circuit design always leads to high power dissipation and increases the fault rates. Due to these difficulties, researchers explore the reversible logic circuit as an alternative way to implement the low-power circuit design. It is also widely applied in recent technology trends like quantum computing. Analyzing the correct functional behavior of these circuits is an essential requirement in the testing of the circuit. This paper presents a testable design for the *k*-CNOT based circuit capable of diagnosing the Positive Control Flipping Faults (PCFFs) in reversible circuits. The proposed work shows that generating a single test vector that applies to the constructed design circuit is sufficient for covering the PCFFs in the reversible circuit. Further, the parity-bit operations are augmented to the constructed testable circuit that produces the parity-test pattern to extract the faulty gate location of PCFFs. Various reversible benchmark circuits are used for evaluating the experimental results to establish the correctness of the proposed fault diagnosis technique. Also a comparative analysis is performed with the existing work.

Corresponding Author:

Hiren Kumar Deva Sarma,
Department of Information Technology,
Gauhati University,
Guwahati-789034, Assam, India.
Email: hirenkdsarma@gmail.com

1. INTRODUCTION

Due to the rapid advancement of the electronics industry in the present day, integrated circuits are constructed with a tiny semiconductor chip, where a billion transistors on a single chip. The ever-increasing demand for the technology leads to more components being fabricated into a single chip. The rapid growth of the fabrication technology reduces the chip size and increases the computational speed. However, complex interconnection features and higher operating frequencies increase the power dissipation and more probability of the error rate.A lot of effort has been utilized to design the low-power circuit; still, power dissipation is challenging for semiconductor electronic devices. Therefore, reversible computation is considered the alternative solution to compute the computational functions such that low power dissipation can be achieved, and it can be explored for the low-power circuit design [1, 2]. The reversible logic is considered to perform reversible computation and the applicability of reversible computation is widely

applied in recent technology, such as quantum computing [3, 4], optical computing [5], DNA computing [6] and Quantum Cellular Automata (QCA) [7].

In the traditional logic circuit, the computation is performed by irreversible function, i.e., the inputs and outputs are different in terms of numbers. In other words, the system is unable to retrieve the inputs from the outputs. As a result, the traditional logic circuit changes or losses of bit information. During the computation, every bit of information is lost or changed and it is required the power dissipation *kTln2* Joules, which is also called the Landauer principle [8]. In contrast, the reversible circuit maintains the operation in a reversible manner, where a number of inputs and outputs equally exist and run in the backward direction, i.e., information can be uniquely retrievable from both inputs and outputs. Therefore, the reversible computation leads to no information loss, which was postulated by Charles H. Bennett [9] in 1973. In this context, the reversible circuit gains much importance for low-power circuit design technologies as compared to the traditional circuit.

In conventional circuits, the cost metrics are measured by a gate count and a circuit delay. The gate count defines the total number of gates applied in a given circuit that represents a circuit depth, where each gate has a different transistor cost in CMOS technology [10]. The circuit delay depends on the propagation delay and the transmission delay of a circuit. The propagation delay indicates how long it takes to traverse one bit of information from one wire (signal) to another. At the same time, the transmission delay is considered the time required to extract all the bits in the first place of the wire. In contrast, the reversible circuit structure is not similar as compared to the conventional circuit. The formulation of the reversible circuit is based on the linear sequence of reversible gates and each gate's interconnection is free from the fan-out and feedback connections. Extra line(s) is included, called garbage line, to maintain logical reversibility. The garbage line produces the garbage bit and relates to the non-constant output line that is not considered the resultant primary output in the reversible circuit. However, it is utilized to execute logical reversibility [11]. Therefore, the garbage line(s) is another parameter for evaluating the reversible circuits' cost metric performance. The metric cost analysis depends on which technology is used to implement the reversible circuits physically. Based on the different applied technologies, the cost metric evaluation of the reversible circuit shows different cost performances. In this context, the quantum cost of a reversible circuit is a cost metric to evaluate the cost function in quantum computing technology, where elementary quantum gates are measured by the quantum cost and the reversible gate is formed by several elementary quantum gates [12]. In CMOS technology, gate cost or gate count is accurately measured by the transistor cost that defines the number of transistors that can fabricate a reversible circuit. For example, the NOT gate quantum cost is 1 in quantum computing technology [13], whereas no transistor cost is required for the NOT gate in CMOS technology [10]. Apart from these, the circuit delay is another parameter to evaluate the cost metric of a reversible circuit computed by considering the maximum number of reversible gates on any input-output propagation path in a given reversible circuit [11].

Several CMOS technologies (e.g., [10, 14, 15, 16]) have been proposed to build the reversible circuits physically. In CMOS technology, the circuit is controlled by the on-off switch that is realized by the transmission gates and these gates are used in the actual fabrication process of the reversible gates. A transmission gate formulates one n-MOS transistor and one p-MOS transistor that leads two MOS transistors in parallel [16]. Based on the formulation of transistors, various reversible gates have a different number of transistors. The two physical lines A and $\overline{A}$ are represented by the single variable A in dual-line CMOS realization [11]. As a result, the NOT gate is performed straight cross-over of the metal for implementation purposes. Due to this reason, transistor cost does not occur and the gate delay is null in NOT gate operation. In general, the number of transistors (transistor cost) is evaluated by 8(n-1) for the reversible gates, except the NOT gate in dual-line CMOS realization, where n indicates the total number of input lines (signals) [11]. Thus, the reversible gates CNOT, TOFFOLI, FREDKIN and PERES have required 8, 16, 16 and 24 transistors, respectively. For example, the CNOT gate includes the pair of one control and one target line; thus, the number of transistors is 8 (i.e., 8(2-1) =8). The authors in [11] have calculated the circuit delay in the reversible circuits by considering all inputs simultaneously; an n-bit CNOT gate and a Fredkin gate produced a unit delay and no delay occurred in the NOT gate.

In the above discussion, we have observed that the transistor cost depends on the number of gates (i.e., gate count) present in reversible circuits, causing a large area needed in CMOS technology. However, it is not always true for large reversible circuits because the transistor cost in CMOS technology affects the number of control lines appearing in any reversible gate. Though the gate count is less, more control connections appear in reversible gates, making transistor costs quite expensive. Moreover, a delay cost is calculated based on the logic depth in the circuit, excluding the counting of the NOT gate in CMOS technology. Therefore, synthesis approaches of a reversible circuit (e.g., [17, 18, 19]) play a vital role for the large reversible circuits before applying any physical implementation technologies.

In every computing device, the incorrect state may occur during the computation and it refers to a fault(s). It indicates that any imperfection causes affects the functional behavior in a circuit. The fault can be represented by the conceptual mathematical model, which is known as a fault model.A fault model helps to reduce the testing complexity to analyze the faults [20]. The author in [21] discussed several structural fault models in reversible circuits and established correlations among the fault models.

Testing of a circuit design domain is the experimental process that is applied to observe the behavior of the correct functionalities of a system. Two essential phases are used to perform the fault diagnosis efficiently in the field of circuit testing. In the first phase, the fault detection process is computed to detect any fault(s) of the circuit. The second phase is used to evaluate the exact location of faults. The computational complexities for both these phases are individual and separate evaluation approaches may be required. Therefore, it is mandatory to formulate an efficient diagnosis test set that can cover both phases in the testing process. The set of input test vectors is generated to distinguish the fault-free and faulty behaviour of the circuit. The test set is a complete test set if all the probable faults are entirely detected. In addition, the test set is converted to a minimal test set when it includes the test vectors in the minimum form for covering all possible faults. Due to the nature of the reversibility, the reversible circuit contains controllability and observability properties [22]. The controllability property allows the generation of the individual input test vector at the beginning level (initial) of the reversible circuit. It is generated using the backtracking process by the test vector at any level of the circuit. Due to the observability property, the circuit's final output is also affected if any test vector changes at any intermediate level of the circuit due to the occurrence of a fault(s). Due to the presence of high controllability and observability, the test generation process is quite observable in reversible circuits.

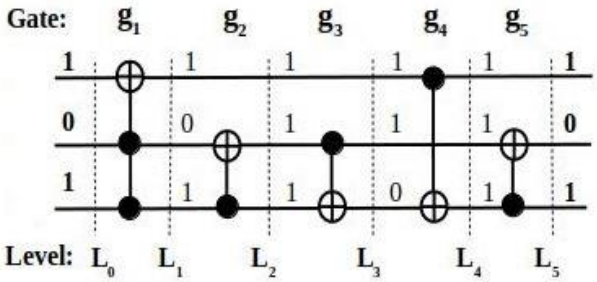

Figure 1. Illustration of controllability and observability property for the *ham3* circuit

To illustrate the controllability and observability property, we consider the *ham3* benchmark reversible circuit, as shown in Figure 1. Here, the circuit is composed with five gates g_1, g_2, g_3, g_4 and g_5 and their corresponding levels are L_0, L_1, L_2, L_3, L_4 and L_5, respectively. To observe the controllability property, let us consider the test vector ⟨110⟩ is appeared after the gate operation g_3 at level L_3. Now, if we apply the backtracking process to the gate g_3 using the same test vector ⟨110⟩, then the test vector ⟨111⟩ is generated at level L_2. Similarly, the backtracking process is executed to the gate g_2 and the test vector ⟨101⟩ is produced at level L_1. Finally, the initial input test vector ⟨101⟩ is produced after the gate operation g_1 at level L_0 by applying one more backtracking process. Therefore, the controllability property guarantees that the applying backtracking process is always generates the unique input at the initial level of the circuit. The observability property shows the effects of the circuit's primary output if the test vector changes at any intermediate level. Suppose the test vector ⟨111⟩ is generated at level L_2 after the gate operation g_2 and it propagates to the subsequent intermediate levels by applying the gate operations and the primary resultant output vector is produced ⟨101⟩ at level L_5, as shown in Figure 1. It means that if we consider another test vector instead of ⟨111⟩ at level L_2, then the final output of the circuit is also affected and produces a different output test vector.

The evaluation of circuit fault detection generally depends on generating an efficient complete test set. Moreover, the process of generating a minimal complete test set in a circuit is also considered an NP-hard problem [20]. In addition, determining the fault localization may be considered the different approaches, which are not similar to the test set generation approaches to detect the faults. Several works on ATPG (Automatic Test Pattern Generation) methods for reversible circuits have been applied (e.g., [22], [23], [24], [25], [26], [27], [28]) to detect faults. The fault detection process of reversible circuits is performed by various approaches, like exact algorithms (deterministic approaches), heuristic approaches and Design for test (DFT) techniques. Each of the approaches has its own computational complexities. The deterministic approach is used to compute the minimal solution for determining the complete test set, but computational run time is expansive for large circuits. In another way, the heuristic approaches are applied to generate an

ATPG. However, heuristic approaches are unable to provide the exact minimal solution. In contrast, the DFT technique is suitable for reducing the computational complexity to cover all the possible faults, but extra circuitry overhead cost is involved. In this work, we have generated a single test vector to detect the faults and as well as localize the PCFFs in reversible circuits. The computational complexity of the proposed work for generating the test vector is required constant time and no extra ATPG effort is involved.

In this proposed work, we have formulated the single test vector based on the detection property of PCFF in the reversible circuit. A detailed discussion of fault detection property is presented in Section 3. Next, we have constructed the testable design circuit that is composed of only k-CNOT based reversible gates. The constructed testable design circuit is used for detecting the PCFFs by applying the generated single test vector. During the fault detection process on the testable design circuit, it maintains the reversibility properties, controllability and observability, which ensure the correct functionalities of the testable design circuit. On the other hand, the same generated single test vector and testable design circuit are applied to evaluate the exact faulty gate location of PCFF. For this purpose, the testable design circuit is applied to generate the parity-test pattern by augmenting the parity-bit operations. The derived parity-test pattern is sufficient enough to localize the PCFF. It is noticed that many of the existing testing approaches have been focused only the generating the efficient complete test set or minimal complete test set for detecting the faults. Moreover, evaluating fault localization may use different approaches that are not precisely similar to fault detection approaches. Also, the computational cost is high for constructing the test set for the fault localization in conventional and as well as reversible circuits. The author in [29] identified that the test set is used for localizing the faults comparatively more significant than the fault detection test set. In this work, we apply the same test generation process on the constructed testable design circuit for the fault diagnosis process. Moreover, the generation process for the parity-test pattern is derived only from the simple execution of the parity-bit operation to locate the faulty gate in our proposed testable design circuit. The generation of the parity-test pattern is required linear time, which is based on the gate count.

The main contributions of this paper are as follows:

- The process for the test generation is formulated based on the fault detection property of PCFF that is derived by the single test vector;
- Construct a testable design circuit to detect all probable faults in PCFFs by applying the generated single test vector;
- The parity-test pattern is generated by augmenting the parity-bit operation on the constructed testable circuit to determine the faulty gate location of PCFFs.

The remainder of the paper is organized as follows. An overview of the reversible logic function, basic gates of the reversible circuits and the structure of the reversible circuits are presented in Section 2. A general discussion on fault models and the PCFF model in reversible circuits is included in this section. We also discuss the previous works relating to various approaches for detecting faults and identifying the faulty gate location in reversible circuits. The proposed testable design approach for detecting and locating the PCFFs is discussed with detailed illustrations in Section 3. Section 4 presents the experimental results, which are based on various benchmark circuits, and a comparative analysis with existing work is reported here. Section 5 presents the concluding remarks of the proposed work.

2. PRELIMINARIES

2.1. ReversibleLogic Function

A function behaves as a reversible if an equal number of inputs and outputs exists, such that permutation is performed by the one-to-one mapping operations over the set of inputs that produces the outputs. Due to the inherent properties of the reversible function, the input can be uniquely restored from the corresponding output. More precisely, the system runs backward direction if the operations are realized by the reversible function. In contrast, the conventional logic design does not maintain reversibility because binary inputs and outputs are unable to establish the one-to-one mapping in most of the operations. However, for any irreversible Boolean function is possible to convert the reversible function by adding some extra lines (also garbage lines) and the reversible operations [30].

Example 1. Table 1 shows 2-input and 2-output reversible function $f: (a, b) \rightarrow (a, a \oplus b)$. Suppose, we consider only 2-input and 1-output function $f': (a, b) \rightarrow (a \oplus b)$. A function f' is irreversible because inputs and outputs are not equal in numbers and no bijective operations are performed. Therefore, we add an extra garbage output line `` `a` `` to make *f* a reversible function.

Table 1. 2-input and 2-output reversible function *f*.

Input		Output	
A	b	a	a⊕b
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**0**	**1**	**1**
1	**1**	**1**	**0**

2.2. Reversible Gates

An $n \times n$ gate [22] consists of n input and output signals to implement the reversible function. The *k*-CNOT or MCT (Multiple-control Toffoli) gate [31] is most commonly used to realize any reversible function. Also, the logical structure of the *k*-CNOT gate represents several classical reversible gates, such as the NOT gate, CNOT, or FEYNMAN gate [32] and Toffoli gate [31]. Every input line of the *k*-CNOT gate is connected to the positive control/negative control (•/◦), unconnected connection(s) and target connection (⊕), where *k* denotes the number of control(s) connection that appears in the *k*-CNOT gate. If there is no control connection (i.e., $k = 0$), it behaves as the NOT gate. Similarly, 1-CNOT (i.e., $k = 1$) and 2-CNOT (i.e., $k = 2$) gates are represented as CNOT and Toffoli gates, respectively. The MCT gate is formulated with more than two control connections (i.e., $k > 2$). The gate operation and symbolic representation of each type of gate are depicted in Figure 2.

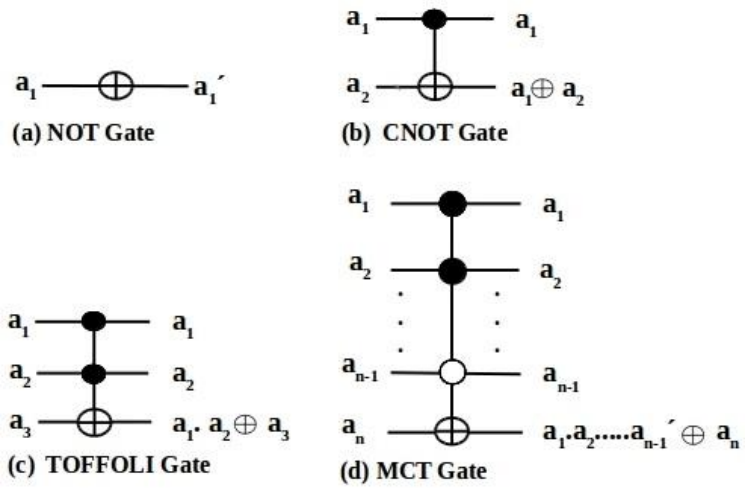

Figure 2. Symbol and operation of classical reversible gates

2.3. Reversible Circuits

The formulation of the reversible circuits is the sequence order of reversible gates. The arrangement of the gates maintains in the form of a linear cascade. The reversible circuit performs the bijective function over the *n*-input and *n*-output and is implemented by the reversible gates. Moreover, reversible circuits do not allow fan-out and feedback connections [3]. If the set of *k*-CNOT gates only consists of a reversible circuit, then it refers to a *k*-CNOT based reversible circuit [22]. We consider only the reversible circuits using *k*-CNOT gates in our proposed technique.

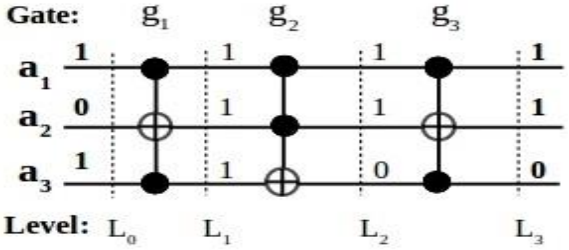

Figure 3. Illustration of *fredkin-6* circuit comprising of three Toffoli gates

Example 2. In Figure 3, the *fredkin-6*circuit contains 3-input and 3-output lines. The circuit comprises three TOFFOLI gates g_1, g_2 and g_3. Each gate follows the bijective operation, and the output generation in every gate level is unique to their corresponding initial input <101>. Here, the initial input vector <101> propagates to the output vector as <110> after the gate operation g_2, which lies between the level L_1 and L_2. If the intermediate gate g_2 produces a different output vector instead of <110>, then the initial input vector of the circuit must be different instead of <101> to maintain the reversibility. Finally, the primary output <110> is generated at level L_3, as illustrated in Figure 3.

2.4. Fault Models in Reversible Circuits

Apart from the traditional fault models like stuck-at fault [22], bridging fault [33], and cell fault [22], some specific fault models are also used only for reversible circuits. These fault models are Single Missing-Gate fault (SMGF) [23, 24], Multiple Missing-Gate fault [23, 24], Partial Missing-Gate fault [24], Repeated Gate fault (RGF) [24] and Crosspoint fault [34]. All of these are considered structural fault models, where faults appear in the gate position and one or more gate interconnections [21]. In the present work, we consider the PCFF [35, 36] in reversible circuits, also represented by the structural fault model.

The CFF represents the permanent fault in the circuit due to the wrong interconnection at the gate level. More precisely, the occurrence of CFFs in the circuit does not change the number of logic gates but changes the desired result. The *k*-CNOT circuit comprises the control connection(s) that appears as a positive control connection (denoted as •) or negative control connection (denoted as ◦) to implement a given reversible function. Due to the wrong interconnection (physical design error) in the gate level, there is a scope of flipping the positive control connection(s) to the negative control connection(s) and vice versa. As a result, the incorrect logic output produces at the affected gate and the desired output is affected. The proposed work focuses on PCFF under the CFF (Control Flipping Fault) fault model. The PCFF describes when the control connection flips from positive control (•) to the negative control (◦). Again, the PCFF is categorized into two types of flipping controls. If only one control is flipping at the gate level in the circuit, it is called a Single Positive Control Flipping fault (SPCFF). In contrast, the number of flipping positive control is more than one within the gate, called a Multiple Positive Control Flipping fault (MPCFF).

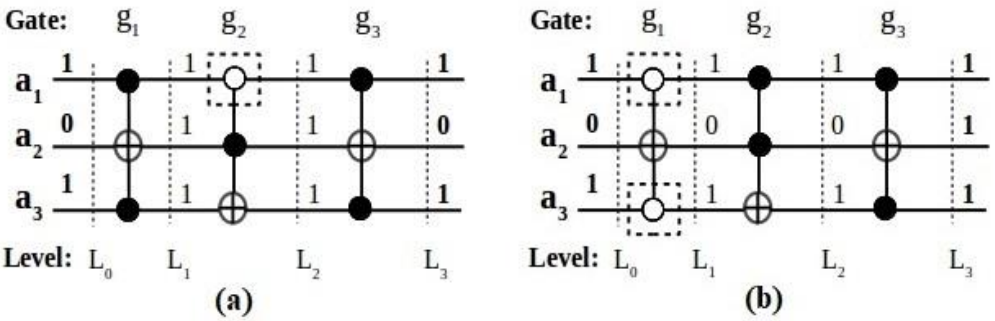

Figure 4. Illustration of PCFFs for the *fredkin-6* circuit: (a) SPCFF at line a_1 in gate g_2 (b) MPCFF at line a_1 and a_3 in gate g_1

Example 3.In this example, we consider the same *fredkin-6* benchmark circuit to illustrate the PCFF, as depicted in Figure 3. Here, the normal output vector <110 > is generated as a primary output of the circuit by applying the input vector <101> at the initial level if the circuit is fault-free. Figure 4 (a) shows the occurrence of SPCFF at line 'a1' (marked by a dotted box) in gate g2. The faulty primary output vector would be <101> instead of a fault-free primary output <110 >. In Figure 4 (b), the MPCFF occurs at lines 'a1' and 'a3' in gate g1. Due to this effect, faulty output vector <111> is generated at the primary level instead of fault-free output <110> by applying the same initial input vector <101>.

2.5. Related Work

This section discusses some of the relevant existing works that are related to the testing of reversible circuits. In 2011, Rahaman et al. in [30] proposed a testable design circuit using *k*-CNOT gates that comprises adding one extra line with one copy of each *k* number of CNOT gates, where *k* is the number of control connections. In this reconfigured design circuit, the universal test set (UTS) with the size (*n*+1) is used to detect different types of faults in the reversible circuit, like SMGFs, RGFs, and PMGFs. This technique showed that only one test vector is used to evaluate the location of SMGF. In 2011, Nayeem et al. [37] proposed an online designing technique that is capable of covering the single-bit fault. In this online design technique, the circuit formulation is based on the exclusive sum of product and each TOFFOLI gate consists of *n*-bit to construct the extended Toffoli gates (ETG) of (*n*+1)-bit. In 2014, Mondal et al. [38]

proposed a technique to identify the location of PMGFs in reversible circuits. For this purpose, the pseudorandom test patterns are applied in the sequence form to generate the unique fault signature for all possible PMGFs in reversible circuits. Further, the fault diagnosis tree is constructed with the help of the generated unique signature. Next, the fault diagnosis tree is used for the traversal process for diagnosing the faults. In 2016, Gaur et al. [39] presented a parity-preserving testable reversible circuit that is used to detect the full single-bit fault coverage. The proposed testability design requires one additional line and an extra *k*-CNOT gate. The extra *k*-CNOT gate is placed just after every original *k*-CNOT gate. In 2018, Nagamani et al. [27] proposed a heuristic method based on a genetic detection algorithm for the various faults in the reversible circuit using TOFFOLI, PERES and FREDKIN gates. This method is divided into two approaches, namely the random search and directed search, which are implemented to construct the test set. However, the heuristic random search approach does not cover all the faults for every test set. Whereas the second approach is a directed search approach that is genetic-algorithm based and has higher fault coverage than the previous approach. In this work, the genetic-based heuristic method is only used for fault detection. In 2020, Handique et al. [28] proposed a method for generating the test set that is used for determining faults in reversible circuits. Here, the faults are considered the SMGFs and MMGFs. The minimizing technique ILP (Integer Linear Programming) is introduced to generate the minimal test set derived from the complete test set to cover all possible faults. The generated complete test set is correlated with other fault models.In 2020, the authors in [40] proposed a fault diagnosis approach for SMGFs, PMGFs and MMGFs in reversible circuits. Here, the complete test set (*T*) is computed by comparing fault-free and faulty circuit specifications. The complete test set (*T*) applies the fault localization algorithm to extract the unique test set (*U*) for each faulty gate to detect and localize the faults. In 2021, Mondal et al. [41] introduced a design for testability (DFT) technique to detect SMGFs and PMGFs in reversible circuits. The DFT technique is formulated into different sets by clustering the gates in a given k-CNOT based circuits. The additional input line is connected to each gate within a cluster. The additional input line to the corresponding gate includes an extra control input so that the designing technique is covered 100% fault coverage for the SMGFs and PMGFs with less quantum cost. Also, the DFT technique is capable of detecting nearly 100% MMGFs.In 2021, Kheirandish et al. [42] proposed a fault diagnosis approach to detect and localize the SMGF, MMGF, PMGF, repeated gate fault (RGF), and single-bit fault (SBF) in reversible sequential circuits. Here, fault detection and correction are implemented by the complement specification table. Based on the complement specification table, faults are detected first, and a correction method is applied to identify the fault(s) location. This approach is applied to the reversible full adder to verify the correctness using the complement specification table concept and the full adder realized by the Clifford+T library. In this approach, different cost metrics are evaluated for the designed circuits. However, the performance of the cost metrics of the fault detection and correction based approach is marginally higher as compared to the existing approaches. In 2022, authors in [35] have proposed a method to construct the complete test set without using any minimization technique for detecting PCFFs in reversible circuits.

As discussed in the above literature review, we have found that the existing methods that are related to the testable design circuits focus only on fault detection purposes. Also, an online testable design circuit required an extra circuitry designing cost to convert the original *k*-CNOT gates and add an extra input line with several control connections for the newly added *k*-CNOT gates. Moreover, we observed that work regarding applying only one test vector for fault diagnosis of the reversible circuits islimited. It is also noticed that some of the existing work used pseudorandom test patterns or considered all the possible inputs to evaluate the location of faults in reversible circuits. This approach is required high computation costs for the larger circuits. Moreover, the test set for fault detection requires further computation to localize the faults.

In our proposed work, the test generation process is formulated based on the fault detection property of a PCFF fault model in reversible circuits that is computed in constant time. The proposed testable design circuit requires only a copy of each original *k*-CNOT gate for fault detection; the parity-bit operations are included without adding any extra control connections. The parity-test pattern is derived from the testable design circuit and is sufficient for localizing the PCFFs in reversible circuits. A detailed discussion of our proposed testable design approach is presented in the following section.

3. PROPOSED FAULT DIAGNOSIS TECHNIQUE

This section elaborately discusses the proposed fault testable design technique that is used for detecting the SPCFF and MPCFF under the PCFF model and localizing the faulty gate in *k*-CNOT based reversible circuits.

3.1. Test Vector Generation Process for PCFF

The *k*-CNOT gate of the reversible circuit comprises a set of positive/negative (•/◦) control and unconnected connections and only one target (⊕) connection. By the definition of PCFF in *k*-CNOT based

reversible circuit, one or more control point is flipped to the positive to negative connection(s). The fault detection technique [35] of PCFF ensures that the target output connection line must be changed the logic value when the flipping controls(s) exist. The purpose of the fault detection technique of PCFF is to set the binary value 1 for all unflipping control connections and assign the binary value 0 or 1 (don't-care) to the remaining connections. Based on the fault detection technique, we set the binary value 1 in all connection lines (i.e., control(s), unconnected(s) and target) to generate the test vector for diagnosis of the PCFFs in the testable design *k*-CNOT circuits. More precisely:

Definition 1. A set of binary values <b_1 b_2…..b_n> is considered a test vector *TV*that is used to an*n*input lines and *N*gates-based reversible circuits for executing the testing process. The operation of each gate g_i is performed by applying the test vector *TV*=<b_1 b_2…..b_n>, where i^{th} gate denotes g_i, for $1 \leq i \leq N$and each bit $b_j \in \{1\}$, for $1 \leq j \leq n$.

Example 4. In Figure 3, the *fredkin-6* circuit comprises three input lines (i.e., $n = 3$) and three gates (i.e., $N = 3$). Now, we generate $TV = <111>$ as per the fault detection technique of PCFF, as mentioned in Definition 1. The $TV = <111>$ must be propagated to every subsequent level of the circuit to satisfy the fault detection property of PCFF.

3.2. Construction of Testable Design Circuit

In this section, we discuss the construction of a testable design circuit that is used for detecting PCFFs in *k*-CNOT reversible circuits. The main objective of constructing the testability circuit is that the generated test vector *TV* will remain the same for all subsequent circuit levels during each *k*-CNOT gate operation, such as the fault detection technique PCFFs applicable for the entire *k*-CNOT gates. The following definition is presented that is used for constructing the testable design circuit.

Definition 2.A block $B=\{B_1, B_2, \ldots, B_N\}$ is a set of blocks, where block B_i represents the i^{th} block in the circuit. The block B_i comprises the combination of the original *k*-CNOT gate g*i* and the duplicate copy of the *k*-CNOT gate, denoted as g´*i*, for $1 \leq i \leq N$. Here, a total number of blocks equals the *N* number of original *k*-CNOT gates.

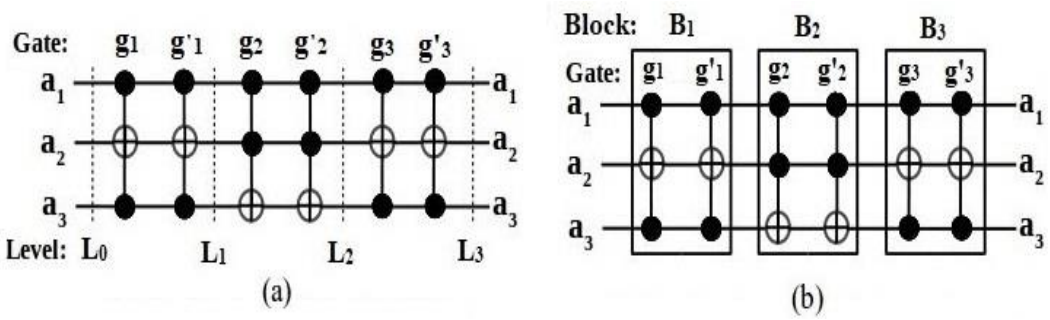

Figure 5. Illustration of testable design circuit for the *fredkin-6* circuit: (a) Design for testability after Step 1 (b) Design for testability after Step 2

The construction of the testable design circuit from the original circuit (Figure 3) comprises two steps. In **Step 1**, we include an additional extra *k*-CNOT gate g´$_i$ in the consecutive place after each of the actual *k*-CNOT gate g_i, where the size of the original gate g_i and the newly added gate g´$_i$ are identical, as illustrated in Figure 5 (a). In our proposed testable design technique, each of the extra *k*-CNOT gates g´$_i$ is assumed to be fault-free for the purpose of testing. In **Step 2**, the testable design circuit is categorized into blocks and each block B_i consists of the pair of gi and g´$_i$, as shown in Figure 5 (b), which is considered a testable design for detecting the faults in the *fredkin-6* circuit.

Lemma 1. The constructed testable design circuit is capable of propagating the initial test vector *TV* to every intermediate block *Bi* and also appears in the primary output.

Proof.Let us consider block B_i in a given testable design circuit that comprises the gates g_i and g´$_i$, as per Definition 2. Suppose we consider the initial inputs $\{a_1, a_2, \ldots\ldots, a_{n-1}, a_n\}$ is provided to testable design circuit. Let be the gate g_i structure at block B_i allows the set of inputs $\{a_1, a_2, \ldots\ldots, a_{n-1}\}$ to be associated with the control or unconnected lines and a*n* associates with the target line. After the gate g_i operation, the output would be $O_{gi} = \{a_1, a_2, \ldots.., a_{n-1}, (a_1.a_2\ldots\ldots a_{n-1} \oplus a_n)\}$, which is propagated an input of the duplicate copy *k*-

CNOT gate g'_i at block B_i in the testable design circuit. Similarly, after the gate g'_i operation, the output $O_{g'i}$ is obtained as follows:

$$=\{a_1,a_2,...,a_{n-1},(a_1.a_2....a_{n-1} \oplus a_1.a_2....a_{n-1} \oplus a_n)\}$$
$$=\{a_1,a_2,...,a_{n-1},(0 \oplus a_n)\}$$
$$=\{a_1,a_2,...,a_{n-1}, a_n\}$$

Hence, the output $O_{g'i}$ of every block B_i produces the same initial input $\{a_1,a_2, \ldots\ldots, a_{n-1}, a_n\}$and also appears in the primary output, which is applied to the testable design circuit.

3.3. Fault Detection Process for PCFFs

The fault detection process starts with applying the generated test vector *TV* to the testable design circuit. As per the above discussion, the applied test vector *TV* is stored in every block *Bi* of the circuit and it also appears as a primary output. As a result, the initial test vector and primary output test vector are identical in the testable circuit if the circuit is fault-free. In contrast, some block *Bi* exists where the applied test vector produces a different test vector after gate operation due to the presence of PCFF(s) in the circuit. If the circuit is faulty, this effect generates a different primary output test vector. The following examples are provided to illustrate the fault detection process of PCFFs.

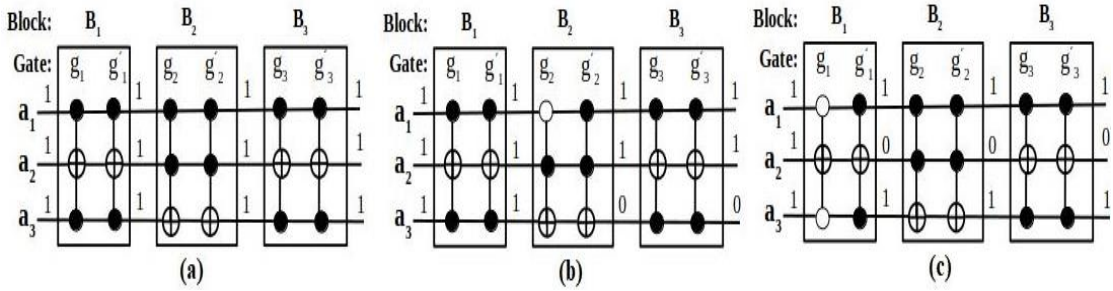

Figure 6. Illustration of fault dectection process in *fredkin-6* circuit: (a) fault-free circuit (b) SPCFF at line a_1 in gate $g_2$of block B_2 (c) MPCFF at line a_1 and a_3 in gate $g_1$of block B_1

Example 5.In this example, we consider the *fredkin-6* circuit to demonstrate the detection process of the single positive control flipping fault (SPCFF). As explained above, the *TV*=<111> is generated for the *fredkin-6* circuit. Now, we apply the *TV*= <111> to the constructed testable circuit and the same *TV*= <111> propagates to blocks B_1, B_2 and B_3 in the circuit. Thus, the same initial input *TV*= <111> occurs at the primary output of the circuit, which is shown in Figure 6 (a). It indicates that the circuit is fault-free. Let us consider that the SPCFF at line 'a_1' in gate g_2 lies in block B_2, where the positive control connection 'a_1' flips to the negative control connection, as shown in Figure 6 (b). Due to the presence of SPCFF, the affected block B_2 produces a different test vector *TV*= <110> that is not identical to the input test vector TV= <111> at the initial level of the circuit. Therefore, the primary output *TV*= < 110> produces in the testable circuit, which is different from the fault-free circuit.

Example 6.Figure 6 (c) shows the fault detection process for the MPCFF, where two positive control connections, 'a_1' and 'a_3' in gate g_1, flipped to the negative control connections. Now, we apply the *TV*=<111> to the testable circuit, and the primary output test vector produces <101> thatis different from the applied initial test vector <111>. Thus, the circuit is faulty due to the occurrence of MPCFF.

In the context of fault detection, we present the following two lemmas with proofs that establish the covering of all probable SPCFFs and MPCFFs in the PCFF model.

Lemma 2. The generated *TV*can detect all possible SPCFFs in the constructed testable design circuit.

Proof. In our test generation process, we consider the input test vector $TV = \langle b_1 b_2 \ldots b_n \rangle$ that is applied to the testable design circuit, where $\forall b_j \in 1$, for $1 \leq j \leq n$ (as per Definition 1). Let us consider that the $TV = \langle b_1 b_2 \ldots b_n \rangle$ is associated with the input lines $\{a_1, a_2, \ldots, a_n\}$, respectively. Suppose, we consider the input line a_n is connected to the target connection (denotes as $\oplus$) and all other remaining lines $\{a_1, a_2, \ldots, a_n\} \setminus \{a_n\}$ are connected to the positive control connections (denoted as •) of gate g_i in block B_i (block B_i as mentioned in Definition 2). We first consider the circuit as a fault-free condition to prove the statement mentioned above. It means that there is no occurrence of SPCFF in the circuit. Based on our constructed testable design circuit, the same initial input test vector $TV = \langle b_1 b_2 \ldots b_n \rangle$generates every block $B_i$in the fault-free circuit condition, as mentioned in Lemma 1. Now, the $TV = \langle b_1 b_2 \ldots b_n \rangle$ is used for the first gate operation g_i of block B_i and it

performs the output of gate g_i as $\langle b_1b_2 \ldots b_{n-1}(b_1.b_2 \ldots b_{n-1} \oplus b_n)\rangle$. Next, this generated output of gate g_i propagates as an input of the next gate g'_i (copy of gate g_i) in block B_i, as per our constructed testable design circuit. After the gate operation g'_i, it performs the output $\langle b_1b_2 \ldots b_{n-1}.(b_1.b_2 \ldots b_{n-1} \oplus b_1.b_2 \ldots b_{n-1} \oplus b_n)\rangle = \langle b_1b_2 \ldots b_{n-1}.(0 \oplus b_n)\rangle = \langle b_1b_2 \ldots b_n\rangle$. It implies that the $TV = \langle b_1b_2 \ldots b_n\rangle$ is identical for every subsequent block B_i in the testable design circuit, if the circuit is fault-free condition. Let us consider that the circuit ina faulty condition due to the occurrence of SPCFF. Due to the effect of SPCFF, any of the control line $a_k \in \{a_1, a_2, \ldots, a_n\} \backslash \{a_n\}$, for $1 \leq k \leq n-1$, is flipped positive control connection to the negative control connection (denoted as ◦). As per a similar process as mentioned above, we apply the $TV = \langle b_1b_2 \ldots b_n\rangle$ to the first gate $g_i$at block B_i and it produces the output of gate g_i as $\langle b_1b_2 \ldots b_k \ldots b_{n-1}.(b_1.b_2 \ldots \overline{b_k} \ldots b_{n-1} \oplus b_n)\rangle$. This generated output behaves as an input to the next gate operation g'_i within the same block B_i. After executing the gate operation g'_i , it generated the output as $\langle b_1b_2 \ldots b_k \ldots b_{n-1}.(b_1.b_2 \ldots b_k \ldots b_{n-1} \oplus b_1.b_2 \ldots \overline{b_k} \ldots b_{n-1} \oplus b_n)\rangle = \langle b_1b_2 \ldots b_k \ldots b_{n-1}.(1 \oplus b_n)\rangle = \langle b_1b_2 \ldots b_k \ldots b_{n-1}.\overline{b_n}\rangle$. It indicates that the $TV = \langle b_1b_2 \ldots .\overline{b_n}\rangle$ is not similar after the gate operation g'_i of block B_i, as compared to the fault-free condition of the circuit. This effect occurs due to the presence of SPCFF at the control connection a_k in the testable design circuit. The generated faulty output test vector $TV = \langle b_1b_2 \ldots \overline{b_n}\rangle$ propagates to every subsequent block B_i and as well as it reflects the final output in the circuit. Hence, it confirms that the *TV* is sufficient for detecting all the possible SPCFFs in the testable design circuit.

Lemma 3.In the constructed testable design circuit, the test vector *TV* detects all possible MPCFFs.

Proof. According to Lemma 2, it is ensured that the input test vector $TV = \langle b_1b_2 \ldots b_n\rangle$appears for every subsequent block $B_i$of the testable design circuit if the fault-free circuit condition. Let us consider the circuit in faulty condition due to the MPCFF at gate g_i. Due to MPCFF, more than one positive control connection flips and forms negative control connections. Suppose we consider the MPCFF occurs at two positive control lines a_i and a_k, where $a_i, a_k \in \{a_1, a_2, \ldots, a_n\} \backslash \{a_n\}$ and a_n is a target connection, as mentioned in Lemma 2. Now, we apply the$TV = \langle b_1b_2 \ldots b_n\rangle$ to the first gate operation g_i of block B_i and it generates the output as $\langle b_1b_2 \ldots b_i \ldots b_k \ldots b_{n-1}(b_1.b_2 \ldots \overline{b_i} \ldots \overline{b_k} \ldots b_{n-1} \oplus b_n)\rangle$. Next, the generated output of gate g_i propagates to the next gate g'_i of the same block Bi. The gate operation g'_i generates the output as $\langle b_1b_2 \ldots b_i \ldots b_k \ldots b_{n-1}(b_1.b_2 \ldots b_i \ldots b_k \ldots b_{n-1} \oplus b_1.b_2 \ldots \overline{b_i} \ldots \overline{b_k} \ldots b_{n-1} \oplus b_n)\rangle = \langle b_1b_2 \ldots b_i \ldots b_k \ldots b_{n-1}(1 \oplus b_n)\rangle = \langle b_1b_2 \ldots b_i \ldots b_k \ldots b_{n-1}.\overline{b_n}\rangle$. It implies that the generated $TV = \langle b_1b_2 \ldots \overline{b_n}\rangle$ is not similar for the block B_i, as compared to the fault-free condition of the circuit. Also, this faulty output vector $TV = \langle b_1b_2 \ldots \overline{b_n}\rangle$ propagates to every subsequent block B_i and appears resultant primary output in the testable circuit. The exact process applies to all other multiple-flipped control connections. The exact process applies to all other multiple-flipped control connections. Hence, it confirms the detection of all possible MPCFFs that occurred in the testable design circuit by applying the single test vector.

3.4. Fault Localization Process for PCFFs

The evaluation of the fault localization process for the PCFFs is presented in this section.For this purpose, parity-bit operations are augmented in the testable design circuit, which evaluates the location(s) of PCFFs in *k*-CNOT circuits. The following definitions are considered to complete the fault localization process.

Definition 3.The set of parity-bit operations is defined as $PB=\{PB_1, PB_2, \ldots, PB_N\}$, for $1 \leq i \leq N$, where each parity-bit operation PB*i* is attached to the corresponding block B_i of the testable design circuit. Each of the parity-bit operations PB_i is connected by the input and output of the target lines for the gates g_i and g'_i, respectively and performed the Exor operation between the logic values of the target lines.

Definition 4.The parity-test pattern is defined as $P_{parity} = \langle P_1P_2 \ldots P_n\rangle$ that consists of parity bits. Each parity bit P_i refers to the i^{th} parity operation PB_i and $P_i \in \{0, 1\}$, for $1 \leq i \leq N$, where *N*denotes the number of gates in the original *k*-CNOT circuit.

The fault localization process starts with checking the parity-bit $Pi \in Pparity$ of their corresponding parity operation PB*i*. Suppose the i^{th} parity-bit $P_i$generates the binary value 1 that is derived from the parity operation PB_i, indicating that the PCFF occurs at the gate g_i in block B_i. Therefore, the occurrence of the parity-bit P_i position ensures the location of the faulty gate affected by the PCFF. The following examples explain the fault localization process for the *fredkin-6* circuit.

Example 7.The set of parity-bit operations $PB=\{PB_1, PB_2, PB_3\}$ is augmented in the *fredkin-6* testable circuit, as shown in Figure 7 (a). The generated *TV*= <111> is applied to the testable circuit. Each parity-bit

operation PB_1, PB_2 and PB_3 has performed the Exoring with their corresponding logic values of the target lines for the input and output, as mentioned in Definition 3. As a result, the parity-test pattern P_{parity}= <000> produces, where every parity-bit generates the logic value 0, i.e., P_1=0, P_2=0 and P_3=0. It indicates the fault localization is null, so the circuit is fault-free. Let us consider that the SPCFF occurs at the control connection line 'a_1' in gate g_2, which lies in block B_2, as shown in Figure 7 (b). Due to the presence of SPCFF at block B_2, the corresponding operation of the parity-bit PB_2 generates the binary value 1 at the 2nd parity-bit position (i.e., P_2=1). Thus, the parity-test pattern would be P_{parity}= <010>. Hence, the location of the SPCFF exists in block B_2, which represents the faulty gate g_2.

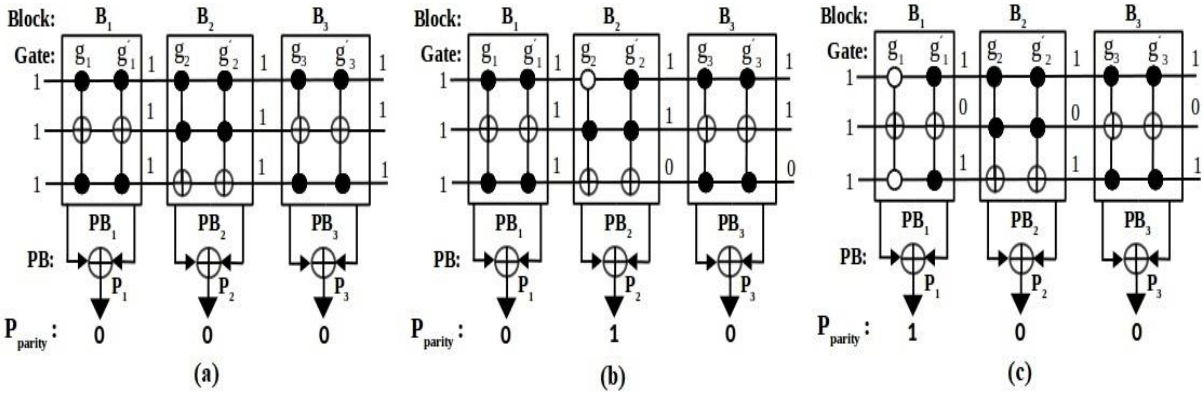

Figure 7. Illustration of *fredkin-6* circuit in the fault localization process: (a) fault-free circuit (b) SPCFF at line a_1 in gate g_2 (c) MPCFF at line a_1 and a_3 in gate g_1

Example 8.In Figure 7 (c), the MPCFF occurs at the control connections 'a_1' and 'a_3' in gate g_1 and the gate g_i associated with block B_1. The generated *TV*= <111> is applied for evaluating the location of the faulty gate. Due to the presence of MPCFF, the parity-test pattern would be P_{parity}= <100>, where the 1st-parity bit position is 1 (i.e., P_1=1) and indicates block B_1. Hence, the location of the MPCFF exists in block B_1, which represents the faulty gate g_1.

Lemma 4.The parity-test pattern P_{parity} is capable of evaluating the faulty gate localization for SPCFFs and MPCFFs under the PCFF model in the testable design circuit.

Proof. In our fault localization process, each parity-bit operation PB_i is augmented in the testable design circuit, where the PB_i corresponds to block B_i. Each PB_i generates the parity-test pattern $P_{\text{parity}}=\langle P_1P_2 \ldots P_N\rangle$, where the parity-bit $P_i \in \{0, 1\}$, for $1 \le i \le N$ (as mentioned in Definition 4). The parity-bit P_i refers to the i^{th} parity-bit operation PB_i. If any faulty gate occurs due to the presence of SPCFFs and MPCFFs, then P_i generates the binary value 1 and the position of the P_i that occurred in P_{parity} ensures the faulty gate location. Let us consider the SPCFF occurs at input positive control connection a_k in gate g_i, where $a_k \in \{a_1, a_2, \ldots, a_n\} \setminus \{a_n\}$, where a_n is considered as the target connection. Now, we apply the same $TV = \langle b_1b_2 \ldots b_n\rangle$ to the testable design circuit that isalso previously used for thefault detection process. Here, the binary value b_n to corresponds to an input of a_n in gate g_i, where $b_n \in TV$ and $b_n \in \{1\}$, as per our test generation process. Due to the effect of SPCFF at positive control connection a_k that refers to the binary value b_k, the output of the target connection a_n produces the same binary value b_n, after completion of the gate operation g_i in block B_i, i.e., $a_n = \{b_1.b_2 \ldots \overline{b_k} \ldots b_{n-1} \oplus b_n\} = \{0 \oplus b_n\} = b_n$. It is because the output of the target connection in the k-CNOT gate operation inverts the binary value if and only if all the input control connections are set as binary value 1. However, the binary value b_k of the positive input control connection a_k is flipped to the negative control connection, causing the binary value $\overline{b_k}$ (i.e., $\overline{b_k} \in \{0\}$) to be generated. As per our testable design circuit, the same generated output b_n of the first gate operation g_i is propagated to the target connection a_n for the next gate g'_i within the same block B_i. Then, the output of the target connection would be a binary value$\overline{b_n}$, after the gate operation g'_i, i.e., $a_n = \{b_1.b_2 \ldots b_k \ldots b_{n-1} \oplus b_n\} = \{1 \oplus b_n\} = \overline{b_n}$. The parity-bit operationPB_i executes the EXORing operation between the binary values b_n and $\overline{b_n}$ for the input and output target connection, respectively, in block B_i(as mentioned in Definition 3). Thus, $PB_i = \{b_n \oplus \overline{b_n}\} = 1$ and the generated binary value 1 is stored to the parity bit P_i that appears to the corresponding position in the parity-test pattern P_{parity}. The generated $P_i = 1$ refers to the i^{th} parity-bit operation PB_i and indicates the block B_i in the testable design circuit. Therefore, the location of the SPCFF occurs in block B_i and represents the faulty gate g_i. The same process applies if the number of flipping positive control is more than one due to MPCFFs.

Hence, the generated P_{parity} is capable of evaluating the faulty gate localization for SPCFFs and MPCFFs in the testable design circuit.

4. EXPERIMENTAL RESULTS

In this section, the proposed testable design technique is implemented on several benchmark circuits [13, 43] to obtain the experimental results. The experiments are performed on a Core-i5 machine with processor i5-8250U CPU @ 1.60GHz × 8, OS Ubuntu v16.04 (64-bit) with 8 GB RAM. The various parameters are used in our experimental results, such as the number of input lines (n), number of gates (N), number of all possible PCFFs (SPCFF and MPCFF), number of fault diagnosis test vectors and CPU time, which are presented in Table 2. In our experimental results, we have observed that the CPU simulation time for evaluating the fault diagnosis process increases when the number of faults gradually increases. For example, consider the circuits *ham3* and *rd32-v0-66*, where the number of gates (N=5) is equal.

Table 2. Experimental Results for the various Benchmark Circuits For Fault Diagnosis of PCFFs with CPU Time (sec)

Benchmark Circuit	*N*	*N*	No. of PCFFs (SPCFF+MPCFF)	No. of Fault Diagnosis Test Vectors	CPU Time (sec)
Peres 9	3	2	4	1	0.0133
ham3	3	5	7	1	0.0182
ex-1-166	3	4	5	1	0.0177
rd32-v0-66	4	5	8	1	0.0190
mod10-171	4	10	33	1	0.7087
4-49-16	4	16	37	1	0.7918
alu-v0-26	5	6	13	1	0.0412
mod5d1-63	5	7	9	1	0.0218
mod8-1-177	5	14	53	1	1.1537
2of5d1	6	18	77	1	1.6632
ham7tc	7	23	53	1	1.0911
rd53rcmg	7	30	123	1	2.1145
rd53d2	8	12	28	1	0.6222
rd73-140	10	20	48	1	1.0581
9symd2	12	28	68	1	1.6127
adr4-197	13	55	487	1	5.6811
041018-169	14	46	60	1	1.1751
hwb5-131	28	88	393	1	4.0721
ham15-28	45	153	194	1	2.3055
hwb8-637	637	8	20575	1	2.875
hwb9-1544	1544	9	66756	1	5.192
ham15-109-214	109	15	427	1	8.618
Ham15tc1	132	15	2816	1	10.132

The circuit *rd32-v0-66* requires more CPU time (0.0190 sec) compared with the circuit *ham3* (0.0182 sec) to compute the fault diagnosis process, which is due to the occurrence of more number of faults (8) in the *rd32-v0-66*. It is also observed that the circuits *mod8-1-177* and *ham7tc* are produced an equal number of faults (53), whereas the circuit *mod8-1-177* requires a higher CPU time (1.1537 sec) as compared to the circuit *ham7tc* (1.0911 sec). However, the gate count (N=14) of the circuit *mod8-1-177* is less than the circuit *ham7tc* (N=23). The reason is that the higher-order k-CNOT gates (two numbers of 4-CNOT gates) have occurred in the circuit *mod8-1-177*, where the circuit *ham7tc* contains only four numbers of 3-CNOT gates. Therefore, the occurrence of the higher-order k-CNOT gates in the circuit is more significant in our fault diagnosis process and causes an increase in the number of faults. Column 5 in Table 2 shows that only one test vector is sufficient to diagnose the PCFFs in the testable design circuit with 100% fault coverage.

Table 3 demonstrates the comparative analysis of our testable design approach and compares it with the previous work [35]. Here, a comparative analysis is performed based on the number of test vectors, which is shown in Column 5 and Column 6 of Table 3. The work in [35] proposed a complete test set generation method that can only detect the SPCFFs and MPCFFs. In this existing work, the complete test set's size exponentially grows when the circuit includes more input lines (n) and the number of gates (N) in the maximum cases, as shown in Column 5 of Table 3. Moreover, this existing method can be used only for fault detection purposes. In contrast, our testable design approach can detect the faults and identify the faulty gate location with the same single test vector. Though our proposed approach is compromised with hardware cost for adding an extra k-CNOT gate for each of the original k-CNOT gates, we achieved only one test vector capable of the entire fault diagnosis process successfully for the SPCFFs and MPCFFs in reversible circuits. In our testable design approach, the required number of test vectors is relatively lower or less than 19× on average compared to the work in [35].

Table 3. Comparison analysis for various benchmark circuits in terms of test vectors withwork in [35]

Benchmark Circuit	*N*	*N*	Total No. of Faults (SPCFF+MPCFF)	No. of Test Vectors [35] (SPCFF+MPCFF)	No. of Test Vectors [Proposed Work] (SPCFF+MPCFF)
Peres 9	3	2	4	2	1
3_17_14	3	6	9	4	1
3_17_13	3	6	9	4	1
ex-1-166	3	4	5	3	1
fredkin_6	3	3	9	3	1
ham3_tc	3	5	7	4	1
4b15g_1	4	14	28	9	1
4b15g_2	4	15	35	7	1
hwb4-11-21	4	11	17	8	1
hwb4d1	4	17	41	7	1
mpsk_4b15g_2	4	15	24	9	1
mpsk_hwb4_13	4	30	19	9	1
ham15-70	15	70	278	53	1
ham15-109-214	15	109	427	94	1
ham15tc1	15	132	2816	73	1
Average			249	19	1

5. CONCLUSION

In this paper, we have introduced a testable design circuit for detecting and identifying the faulty gate location of PCFFs in the *k*-CNOT based reversible circuits. Constructing a testable design circuit requires only a copy of each original *k*-CNOT gate and extra parity-bit operations without adding any extra line and control connections. In this fault diagnosis technique, a single test vector is applied to a testable design circuit for detecting the faults. It produces the parity-test pattern for evaluating the fault localization. Though hardware overhead cost increases ($2\times N$) for additional *k*-CNOT gates involved in the testable design circuit, our fault diagnosis technique does not require any extra test generation process. Moreover, the same single test vector performs the fault detection and localization process, covering 100% fault coverage. In future work, this fault diagnosis technique may be extended to other control flipping faults like negative control flipping faults (NCFFs) and mixed control flipping faults (MCFFs) in reversible circuits.

REFERENCES

[1] A. B´erut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider and E. Lutz, “Experimental verification of landauerâs principle linking information and thermodynamics,” *Nature.*, vol. 483, no. 7388, pp. 187–189, 2012.

[2] R. Wille, R. Drechsler, C. Osewold and A. Ortiz, "Automatic design of low-poer encoders using reversible circuit synthesis," *in 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2012, pp. 1036–1041.

[3] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition,10th ed. New York, NY, USA: Cambridge University Press, 2011.
[4] O. Oumarou, A. Paler and R. Basmadjian, "Quantify: A framework for resource analysis and design verification of quantum circuits,"*in 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE,* 2020pp. 126-131.
[5] C. Taraphdar, T. Chattopadhyay and J. N. Roy, "Mach-zehnder interferometer-based all-optical reversible logic gate,"*Optics & Laser Technology.*, vol. 42, no. 2, pp. 249–259, 2010.
[6] M, Rofail and A. Younes, "A. Synthesis strategy of reversible circuits on dna computers,"*Symmetry.*, vol.13, no. 7, p. 1242, 2021.
[7] X. Ma, J. Huang, C. Metra and F. Lombardi, "Reversible gates and testability of one dimensional arrays of molecular qca,"*Journal of Electronic Testing*, vol. 24, no. 1, pp. 297–311, 2008.
[8] R. Landauer, "Irreversibility and heat generation in the computation process,"*IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.
[9] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, Nov. 1973.
[10] Y. V. Rentergem and A. De Vos, "Optimal design of a reversible full adder," 2005.
[11] M. K. Thomsen and R.Glück, "Optimized reversible binary coded decimal adders,"*Journal of Systems Architecture*,vol. 54, no. 7, pp. 697–706, 2008.
[12] M. Szyprowski and P. Kerntopf, "Reducing quantum cost in reversible toffoli circuits," arXiv preprint arXiv: 1105.5831, 2011.
[13] D. Maslov, "Reversible logic synthesis benchmarks page (2015)," online: http://webhome.cs.uvic.ca/dmaslov/.
[14] B. Desoete, A. De Vos, M. Sibinski, and T. Widerski, "Feynman's reversible logic gates, implemented in silicon," 1999.
[15] B. Desoete and A. De Vos, "A reversible carry-look-ahead adder using control gates," *Integration.*, vol. 33, no. 1-2, pp. 89–104, 2002.
[16] A. De Vos and Y. V. Rentergem., "Reversible computing: from mathematical group theory to electronical circuit experiment," *in Proceedings of the 2nd Conference on Computing Frontiers*, pp. 35–44, 2005.
[17] V. V. Shende, A. K Prasad, I. L Markov, and J. P Hayes, "Synthesis of reversible logic circuits,"*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.
[18] Y. V. Rentergem and A. De Vos, "Synthesis and optimization of reversible circuits," 2007.
[19] N. M Nayeem and J. E Rice, "Improved esop-based synthesis of reversible logic," i*n Reed-Muller Workshop*, 2011.
[20] N. K. Jha and S. Gupta, *Testing of digital systems (pp. I-Vi)*, Cambridge: Cambridge University Press, 2003.
[21] J. Rice, "An overview of fault models and testing approaches for reversible logic," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*.IEEE, 2013, pp. 125–130.
[22] K. N. Patel, J. P. Hayes and I. L. Markov, "Fault testing for reversible circuits,"*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 8, pp. 1220-1230, 2004.
[23] J. P. Hayes, I. Polian and B. Becker, "Testing for missing-gate faults in reversible circuits," in *13th Asian Test Symposium*, IEEE, 2004, pp. 100–105.
[24] I. Polian, T. Fiehn, B. Becker and J. P. Hayes, "A family of logical fault models for reversible circuits," in *14th Asian Test Symposium (ATS'05)*. IEEE, 2005, pp. 422-427.
[25] R. Wille, H. Zhang, R. Drechsler, "Atpg for reversible circuits using simulation, boolean satisfiability, and pseudo boolean optimization,"*in 2011 IEEE Computer Society Annual Symposium on VLSI*, IEEE, 2011, pp. 120–125.
[26] A.N. Nagamani, S. Ashwin, B. Abhishek, and V. K. Agrawal, "An exact approach for complete test set generation of toffoli-fredkin-peres based reversible circuits," *Journal of Electronic Testing*, vol. 32, no. 2, pp. 175–196, 2016.
[27] A.N Nagamani, S.N Anuktha, N. Nanditha, and V. K. Agrawal, "A genetic algorithm-based heuristic method for test set generation in reversible circuits." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 324–336, 2017.
[28] M. Handique, J. K. Deka, and S. Biswas, "An efficient test set construction scheme for multiple missing-gate faults in reversible circuits." *Journal of Electronic Testing*, vol. 36, pp. 105–122, 2020.
[29] M. Handique, J. K. Deka, and S. Biswas, "Fault localization scheme for missing gate faults in reversible circuits." *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 4, pp. 1–29, 2022.
[30] H. Rahaman, D. K. Kole, D. K. Das and B. B. Bhattacharya, "Fault diagnosis in reversible circuits under missing-gate fault model,"*Computers & Electrical Engineering*, vol. 37, no. 4, pp. 475–485, 2011.
[31] T. Toffoli, "Reversible computing," in *International Colloquium on Automata, Languages, and Programming*. Springer, Berlin, Heidelberg, 1980, pp. 632–644.
[32] R. P. Feynman, "Quantum mechanical computers," *Foundations of physics*, vol. 16, no. 6, pp. 507–531, 1986.
[33] H. Rahaman, D. K. Kole, D. K. Das and B. B. Bhattacharya, "Optimum test set for bridging fault detection in reversible circuits," in *16th Asian Test Symposium (ATS 2007)*. IEEE, 2007, pp. 125–128.
[34] J. Zhong and J. C. Muzio, "Analyzing fault models for reversible logic circuits,"*in 2006 IEEE international conference on evolutionary computation. IEEE*, 2006, pp. 2422–2427.
[35] M. Handique, A. Prasad and H. K. D. Sarma, "Complete test set generation for control flipping faults in reversible circuits," in *Contemporary Issues in Communication, Cloud and Big Data Analytics*,Springer, Singapore, 2022, pp. 345–355.
[36] M. Lukac, M. Kameyama, M. Perkowski, P. Kerntopf and C. Moraga, "Fault models in reversible and quantum circuits," in *Advances in Unconventional Computing*, Springer Cham, pp. 475–493, 2017.

[37] N. Nayeem and J. Rice, "A simple aaproach for designing online testable reversible circuits," in Communication Computers and Signal Processing (PacRim), *inIEEE Pacific Rim Conference on communications, computers and signal processing*,IEEE, 2011, pp. 85–90.
[38] B. Mondal, P. Das, P. Sarkar and S. Chakraborty, "A comprehensive fault diagnosis technique for reversible logic circuits,"*Computers & Electrical Engineering*, vol. 40, no. 7, pp. 2259–2272, 2014.
[39] H. M. Gaur, A. K. Singh and U. A. Ghanekar, "A new dft methodology for *k*-CNOT reversible circuits and its implementation using quantum-dot cellular automata," *Optik*,vol. 127, no. 22,pp. 10593–10601, 2016.
[40] B. Mondal, C. Bandyopadhyay, A. Bhattacharjee, D. Roy, S. Parekh and H. Rahaman, "An aaproach for fault detection and localization of missing gate faults in reversible circuits," IETE Journal of Research, pp. 1–21, 2020.
[41] J. Mondal, A. Deb and D. K. Das, "An efficient design for testability approach of reversible logic circuits," *Journal of Circuits, Systems and Computers*, vol. 30, no. 6:2150094, 2021
[42] D. Kheirandish, M. Haghparast, M. Reshadi and M. Hosseinzadeh., "Efficient techniques for fault detection and location of multiple controlled toffoli-based reversible circuit," *Quantum Information Processing*, vol. 20, no. 11:370, 2021.
[43] R. Wille, D. Große, L. Teuber, G. W. Dueck and R. Drechsler, "Revlib: An online resource for reversible functions and reversible circuits," in *38th International Symposium on Multiple Valued Logic (ismvl 2008).* IEEE, 2008, pp. 220–225.

BIOGRAPHY OF AUTHORS

Dr. Mousum Handique received his Bachelor of Engineering (B.E) in Computer Technology from Nagpur University, in 2002 and Master of Technology (M.Tech) degree from Tezpur University, Tezpur, Assam in 2005. He has received his Ph.D. degree in the year 2020 in the field of VLSI Testing and Reversible Computing from India Institute of Technology (IITG), Guwahati-781039, Assam, India. He was working as a Lecturer in Sikkim Manipal Institute of Technology, Majitar, Sikkim, India (from September, 2005 to November, 2007). He is currently working as an Assistant Professor, CSE, TSSOT, Assam University, Silchar, Assam, India. His current research interests include Testing and Synthesis of Reversible and Quantum Circuits, Formal System Verification, Machine Intelligence, Workflow Automation, and Queueing Theory. Presently, QTEAM of the CSE Department, TSSOT, involves developing the methods to determine the erroneous behaviour and localize the faults for reversible quantum circuits. Moreover, QTEAM is developed DNA-based reversible circuits to detect the various faults in reversible circuits.

Dr. Hiren Kumar Deva Sarma is a Professor in the Department of Informaton Technology at Gauhati University (GU in NIRF Ranking 2022: 36 in University Category). He received B.E. degree in Mechanical Engineering from Assam Engineering College in 1998, completed M.Tech in Information Technology from Tezpur University in 2000 and received Ph.D. degree from Jadavpur University (Department of Computer Science and Engineering) in 2013. He is a recipient of Young Scientist Award from International Union of Radio Science (URSI) (2005). Dr. Sarma received IEEE Early Adopter Award in the year 2014.
He has published more than ninety research papers in different International Journals, referred International and National Conferences. Moreover, Dr. Sarma has co-authored 3 books, edited one book and five conference proceedings in book format; all are published by prestigious international publishing houses like Springer Nature Group and Apple Academic Press.
Prior to his appointment in Gauhati University, Assam he was with Sikkim Manipal Institute of Technology at Sikkim and SRM University Andhrapradesh at Amravati.
His current research interests are wireless sensor networks, IoT, mobility management in IPv6 based networks, cognitive radio networks, information centric networks, reversible and quantum circuits, network security, and big data analytics.