

An Adjacent Gray Code Pairing Approach for Fault Identification in Reversible Circuits

Dimpimoni Kalita¹, Mousum Handique²

^{1,2}Department of Computer Science and Engineering, TSSOT, Assam University, Silchar, 788011, Assam, India

Article Info

Article history:

Received Aug 10, 2025

Revised Sep 21, 2025

Accepted Nov 22, 2025

Keywords:

Reversible Circuit

Fault Models

Missing Gate Fault

Test Vector

Adjacent Gray Code Pairing

ABSTRACT

The classical computing works on the concept of irreversibility increasing the heat dissipation in computing machinery. The remarkable ability of reversible computing is to reduce this dissipation of heat and to generate lossless information. The reversible circuit is the way to implement the reversible function. Therefore, the perfection of the functional behavior of the circuit plays a vital part within the realm of testing. Occurrence of faults in the reversible circuits creates a dysfunctional behavior in the circuit. Here in this study, a fault detection approach has been devised for reversible circuits that effectively identifies all categories of Missing Gate Faults (MGFs), such as single, multiple, partial, and repeated gate faults, through the use of the Adjacent Gray Code Pairing (AGCP) technique. The approach includes a process of conversion of binary to gray codes and pairing of non duplicative consecutive adjacent codes by which test vectors can be achieved for finding the respective faults with a 50% reduction in test set size. An experimental study and comparative analysis with existing methods have been carried out using a variety of standard reversible circuits.

Copyright © 2025 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Mousum Handique

Department of Computer Science and Engineering, TSSOT, Assam University

Silchar-788011, Assam, India

Email: mousum.handique@aus.ac.in

1. INTRODUCTION

Gordon Moore's prediction [1] suggests that the complexity of integrated circuits, in terms of fabrication density, grows exponentially every couple of years. Due to the rapid development in semiconductor devices, reducing the transistor size on an atomic scale generates lower power voltages and compromises with the smaller interconnect features in a small silicon area. As a result, power dissipation and error rate are increased. In modern electronic circuit technology, operating signals with minimal power dissipation are closely related to logical and physical irreversibility, where logical irreversibility performs a one-way function [2] without using the inverse function and physical irreversibility indicates that the system is capable of running only the one direction. Logical irreversibility fails to provide a unique correspondence between outputs and their original inputs, leading to the loss of information per bit during circuit computation [3]. In this context, diminution of information content per bit is denoted as $kT \ln 2$ Joules and also known as Landauer limit [3], where k is denoted as the Boltzmann constant and value is approximately $1 \times 10^{-23} J/K$, and T is the temperature in Kelvins. Charles H. Bennett [4] effectively demonstrated in 1973 that reversible logic operations can minimize power dissipation and preserve information. The essential idea behind reversible logic is to perform computations without information loss, aiming to reduce per-bit data dissipation to the Landauer limit. Therefore, reversible logic has gained attention from the research community in lower power circuit design in

different technology-oriented applications.

As a means to realize information-preserving computation, reversible circuits are utilized as an alternative design methodology. Reversible circuits utilize distinct input-output pairing (i.e., bijective) function between inputs and outputs, where the input count matches the output count throughout the computational process. In other words, input-output symmetry is a fundamental requirement in reversible circuits and ensure an unambiguous link between inputs and outputs [5]. In contrast, existing gates of the traditional circuit like AND, OR, NAND, NOR, XOR, and XNOR are irreversible in nature. Because some bit of information is lost or changed while computing the primary output during these gate operations and causes more power dissipation [6]. The applicability of the reversible circuit is applied in DNA computing [7–9], optimal computing [10, 11], cryptography [12, 13], quantum-cellular automata [14, 15], etc. Reversible circuits are often treated as an intermediate conceptual layer for developing quantum circuits. Perkowski et al [16] presented symmetric functions with $O(n^2)$ depth using reversible logic to implement the quantum circuit. Javadi Abhari et al. [17] developed a technique for the quantum compiler, named ScaffCC, that is based on the simulation of a reversible circuit to formulate and transform to the quantum sub-circuits and quantum circuits, respectively. Oumarou et al. [18] presented a design framework for simulating the quantum circuits, named as QUANTIFY, which is enabled through the use of TOFFOLI reversible gates in circuit construction. From this perspective, both the synthesis and functional assessment of reversible circuits are important—synthesis for optimal and efficient design, and testing for assessing functional correctness. The efficiency of reversible circuit synthesis is assessed through multiple parameters, including circuit depth, quantum costs, the quantity of ancilla inputs, and the amount of garbage outputs [19]. Several synthesis strategies have been researched for reversible circuit development [19–23]. Alternatively, testing is crucial for evaluating the precise operation and proper functionality of reversible logic circuits. Analogous to the traditional circuit, testing mechanism is an indispensable approach to evaluate and justify the correct functionality of the reversible circuits.

High operating frequency and small interconnection of features lead to the total defects in the circuit. Thus, testing is essential in making the circuit work reliably and perform steady functions. An arbitrary anomaly that is prone to a circuit which disturbs the circuit's functionality or behavior can be termed a fault or an error [24]. The testing process lies between the two procedures; the fault detection procedure observes the correct functionality of the circuit. If the circuit's behavior is unknown or does not provide the desired output, the detection strategy is carried out to uncover the fault(s). Following this, fault diagnosis is conducted to pinpoint the location of the intended fault(s), and suitable actions are implemented to ensure effective fault identification. Therefore, fault detection is the initial procedure in the testing process of any circuit. There is a need for abstract representations to accurately evaluate the physical defects or faults in the circuit. These conceptual depictions of the faults have been created using a mathematical framework known as a fault model [25]. In the present literature, numerous fault modeling techniques have been explored to extract the structural behaviour of the circuit to represent the faults in reversible circuits. Established fault models like the stuck-at [26–29], bridging [28, 30, 31], and cell fault models [26] are employed to depict structural faults in reversible circuits using a wire-focused methodology. Nonetheless, the behavior of reversible gates is fundamentally pulse-driven [32]. In this connection, a set of targeted fault models are incorporated into reversible circuit analysis. Also, the structural representation of these fault models is the technology independent [33], which are considered more appropriate in reversible circuits. Within the framework of the Missing Gate Fault (MGF) model, faults are categorized into Single Missing Gate Fault (SMGF), Multiple Missing Gate Fault (MMGF), Repeated Gate Fault (RGF) and Partial Missing Gate Fault (PMGF) [32, 34, 35]. The interconnection of these faults in the MGF model has been discussed in [36].

Through testing, faults can be identified and accurately located in any given circuit. It is performed by the two approaches: static and dynamic testing. In static testing, the test generation process is applied to the circuit after the normal operation is completed from the circuit. Conversely, dynamic testing is applied in parallel with the circuit's standard operational processes. The performance of these testing approaches depends on the various parameters that are required for evaluating the faults in a circuit. Generating an effective test set is the central objective of static testing in traditional and reversible circuits, measured by key factors such as test length, number of gates, fault coverage, execution time, quantum cost, and the robustness of the Design for Testability (DFT) methodology. The online testing approaches have been formulated with the designing of existing circuit using the modification or included new gates; one of the common dynamic testing scheme is Built-In Self-Testing (BIST) that is also applied in reversible circuits [37, 38].

This paper presents the implementation of a testing strategy utilizing gates from the Multiple-Controlled

Toffoli (MCT) gate library. The approach is based on MGF model, which includes SMGF, MMGF, RGF, and PMGF. The proposed method in this study generates a mixed test set aimed at detecting all fault types defined under the MGF model in MCT-based reversible circuits. The method incorporates reflected binary codes or gray codes employed in a tiered computational process across all circuit gates. The essential outcomes of this study are listed as follows: (i) Design a unified fault detection approach to identify multiple fault types within the MGF framework (ii) The constructed entire test set is labeled a mixed test set owing to its effectiveness in identifying all variants of MGF faults (iii) Experimental result shows to reduce the test set size to the half of the origin that ultimately reduces the computational mechanism.

2. PRELIMINARIES

2.1. Reversible Logic Function and Reversible Gates

Reversible logic functions are defined by their bijective nature, where each input maps uniquely to an output and every possible output is covered and the quantity of source data and produced units remains the same. Let us consider two set A and B . The set A with a logic function $f(a_1, a_2, a_3, \dots, a_n)$ is containing n number of elements and the set B with a logic function $f(b_1, b_2, b_3, \dots, b_m)$ contains m number of elements then the function $f: A \rightarrow B$ is said to be onto if $f(A)=B$, i.e., cardinality in both the set should be equal ($n=m$) and it is said to be one-one if the elements from both the sets maintain a unique relation among them. Therefore, the function $f: A \rightarrow B$ is bijective as it is both onto (surjective) and one-one (injective). To illustrate the reversible logic function let us consider an example which is described in Table 1. A reversible function with four variables is considered in the example which leads to ($2^4=16$) number of elements in the function. Here, the function $f(3, 4, 9, 6, 7, 1, 15, 12, 10, 0, 14, 2, 5, 11, 8, 13)$ is permuted over $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$ so that one to one mapping occurs between the elements i.e., $f(0)=3, f(1)=4, f(2)=9, f(3)=6, f(4)=7, f(5)=1, f(6)=15, f(7)=12, f(8)=10, f(9)=0, f(10)=14, f(11)=2, f(12)=5, f(13)=11, f(14)=8, f(15)=13$ and the number of elements is also same in both the functions. Therefore the function f is bijective as it satisfies both one-to-one and onto properties.

Table 1. Reversible function f

Function	Input Vectors				Output Vectors				Permutation
f	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	f
0	0	0	0	0	0	0	1	1	3
1	0	0	0	1	0	1	0	0	4
2	0	0	1	0	1	0	0	1	9
3	0	0	1	1	0	1	1	0	6
4	0	1	0	0	0	1	1	1	7
5	0	1	0	1	0	0	0	1	1
6	0	1	1	0	1	1	1	1	15
7	0	1	1	1	1	1	0	0	12
8	1	0	0	0	1	0	1	0	10
9	1	0	0	1	0	0	0	0	0
10	1	0	1	0	1	1	1	0	14
11	1	0	1	1	0	0	1	0	2
12	1	1	0	0	0	1	0	1	5
13	1	1	0	1	1	0	1	1	11
14	1	1	1	0	1	0	0	0	8
15	1	1	1	1	1	1	0	1	13

The components utilized for designing reversible circuits are referenced to as reversible logic gates. The reversible gate must follow the reversible logic function, i.e., it must be bijective in nature containing identical count of sources and responses and maintaining non-redundant pairing between sources and responses. The Multiple Controlled Toffoli (MCT) or k -CNOT library contains three basic gates namely NOT, CNOT and Toffoli gate which is shown in Figure 1. Two core elements: control point ($\bullet$) and target point ($\oplus$) are used in the construction of all these gates; the source lines $I_1, I_2, \dots, I_n$ and response lines $O_1, O_2, \dots, O_n$ [39]. If $I_i=1$ or $I_i=0$ and there exists a control point in the line then it results $O_i=1$ or $O_i=0$, respectively, where $1 \leq i \leq n$ and n denotes input counts of the circuit. If $I_i=1$ and if there exists a target point in the line without having any connections with control point then it results $O_i=0$ and similarly, for $I_i=0$ the result for O becomes equal to 1, as shown in Figure 1 (i). Thus, the target point acts as an inverse point. This functionality of target point is named as a basic NOT gate. A gate consisting of two source lines containing a control point and a target

point and if there is a connection between the control and the target point is known as Feynman gate (or CNOT gate) [40]. The response for the source line containing a control point will be exactly similar (i.e., $O_1 = I_1$) and for the source line containing the target point and having connection with the control point of the source line will operate according to XOR function (i.e., $O_2 = I_1 \oplus I_2$), which is depicted in Figure 1 (ii). Thus there will be four numbers of combinations for two input lines ($2^2=4$). For source $I_1=0$, $I_2=0$ the response becomes $O_1=0$, $O_2=0$; for source $I_1=0$, $I_2=1$ the response becomes $O_1=0$, $O_2=1$; for source $I_1=1$, $I_2=0$ the response becomes $O_1=1$, $O_2=1$; for source $I_1=1$, $I_2=1$ the response becomes $O_1=1$, $O_2=0$. A gate containing three lines having control points in two lines connected to a target point in one line is known as the Toffoli gate [39]. The response for the source lines containing the control points will be exactly similar to the sources used for that line (i.e., $O_1 = I_1$ and $O_2 = I_2$) and for the line containing the target point and having connection with the lines having control points will operate according to XOR function (i.e., $O_3 = I_1 \cdot I_2 \oplus I_3$), as shown in Figure 1 (iii).

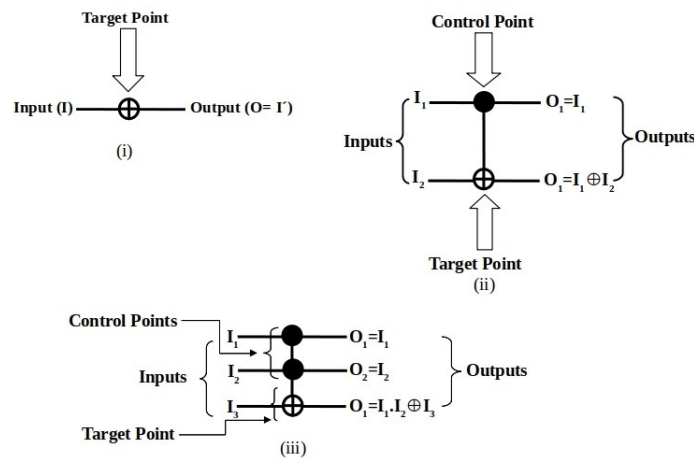


Figure 1. Illustration of three fundamental Reversible Gates: (i) NOT Gate, (ii) CNOT Gate (iii) Toffoli Gate.

2.2. Reversible Circuits

A collection of reversible gates is known as reversible circuits, which are based on reversible logic functions. The reversible circuit also works by following the rule of linear cascading, i.e., it takes the primary input and processes it through each gate successively, with the response of each gate becoming the source for the adjacent gate until it reaches the final gate of the circuit [41]. Also, no feedback and fan-out connections are allowed in the reversible circuits [42]. The gate count provides an estimate of the circuit's scale. Thus, the circuit's span is N when $G_1, G_2, \dots, G_N$ are the gates contained in the circuit. For a circuit having N gate count, it is going to have $N+1$ number of levels as depicted in Figure 2 from L_0 to L_N .

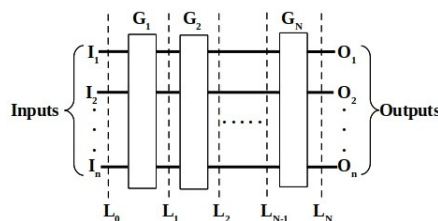


Figure 2. Demonstration of structure of Reversible circuit.

2.3. Fault Models in Reversible Circuits

When a circuit fails to behave normally as because of some unwanted features then a circuit is assumed to be affected from certain faults. Different fault models have been devised to pinpoint the faults so that it became easy for target of testing purpose. The current work is concerned with the Missing Gate Fault (MGF) model. They are structural faults that emerge in the circuit owing to insufficient or mistuned input pulses [34]. An MGF model includes SMGF [32, 34], MMGF [34], PMGF [34] and RGF [34]. When a gate gets disappeared from a circuit it is identified as SMGF. SMGF equals the total gate count within the circuit.

The sequential removal of two or more gates in succession is MMGF and the possible instances of MMGF can be evaluated by $[N(N-1)]/2$ where N represents gate count. If a circuit loses one or more control points, it results in a PMGF fault. The total presence of control points in the circuit defines the quantity of PMGFs. The first order PMGF means the missing of one control point at a time and it is regarded in our work. RGF is the repeated use of any gate indefinitely. Here, in our case, we have considered the singular occurrence of gate repetitions. Thus, the total count of RGF in a circuit will be same as the total gate usage in the circuit. The demonstration of various type of faults under MGF model is portrayed in Figure 3.

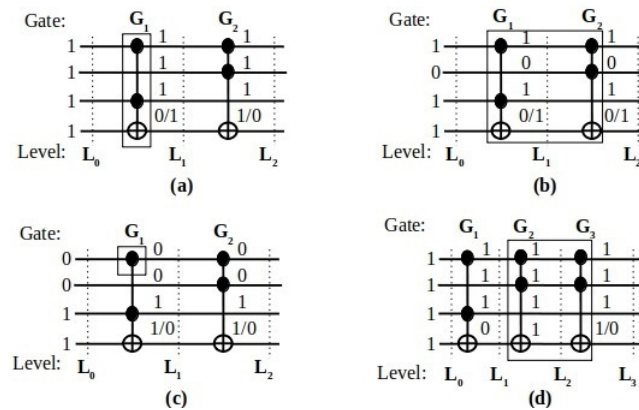


Figure 3. Illustration of various faults under MGF model using benchmark circuit *Toffoli_double4*: (a) SMGF at G_1 gate (b) MMGF at G_1 and G_2 gates (c) PMGF at first control connection of G_1 gate (d) RGF at G_2 and G_3 gates

Example 1. The concept of MGFs is illustrated by considering a reversible standard circuit *Toffoli_double4*, as illustrated in Figure 3. The setup comprises of 4-input, 4-output, and 2 k -CNOT gates. As outlined in Figure 3 (a), the first gate G_1 of *Toffoli_double4* circuit is missing because of the existence of SMGF. Thus, the faulty primary response $\langle 1110 \rangle$ is generated alternative to a defect-free primary outcome $\langle 1111 \rangle$ through activation of the initial test vector $\langle 1111 \rangle$. Figure 3 (b) shows MMGF model, where two consecutive gates G_1 and G_2 are missing. As a result, the faulty response $\langle 1011 \rangle$ is occurred instead of fault-free response $\langle 1010 \rangle$ for test vector $\langle 1011 \rangle$. To illustrate the PMGF, we consider the missing control point at the first source line for the gate G_1 . Thus, the last source line of the gate G_2 is affected by generating the faulty primary response $\langle 0010 \rangle$ instead of fault-free response $\langle 0011 \rangle$ of the circuit employing the original test sequence $\langle 0011 \rangle$, which is represented in Figure 3 (c). RGF is depicted in Figure 3 (d) in which one-time repetition of gate G_2 is regarded as leading to the occurrence of faulty response $\langle 1110 \rangle$ instead of fault-free response $\langle 1111 \rangle$ for the base input test vector $\langle 1111 \rangle$.

2.4. Related Work

Subsequent discussion highlights earlier research on fault anomaly scheme identification in reversible circuits. In 2004, the work in [32] introduced a method for identifying SMGFs through the generation of efficient test vectors. The authors developed a design-for-testability (DFT) approach utilizing a single wire and multiple 1-CNOT gates, triggered by one test vector, which necessitated modifications to the original circuit configuration. In 2008, authors [43] proposed an incorporation of one surplus line, and each k -Controlled NOT gate was cloned in the circuit to support the identification of SMGF, RGF, and PMGF. In 2011, a novel SAT-based algorithm [44] was designed to determine least entire test set for SMGF and PMGF. The algorithm's experimental evaluation demonstrates its effectiveness, showing its applicability to reversible circuits comprising over 2,000 gates. In 2011, the authors in [45] presented and assessed alternative method for giving rise to test patterns impulsively for reversible circuits. A long side a simulation driven approach, the methods leverage Boolean satisfiability and pseudo-Boolean optimization. The study in [46], published in 2012, presented a streamlined test vector generation strategy called the Ping Pong method for detecting SMGF and RGF faults in reversible circuits. The method comprises four main steps: first, identifying the fault coverage list for each SMGF and RGF; second, exploring a feasible order for fault simulation; third, performing cycle dropping; and finally, selecting the most effective sequence. In 2017, authors in [47] have generated Automatic test patterns

for MMGFs detection considering the missing of only gates. They used Binary Decision Diagram (BDD) to store the generated patterns and found dependency analysis between two gates. In 2017, authors in [33] specifically aims to argue that the testing of reversible circuits, considering handling contemporary fault models like omitted gate, skipped control and diminished control are fairly uncomplicated process. The assertion is that dedicating significant time and effort to generate more optimal test patterns may not be justified. To support this claim, two test generators representing extreme scenarios have been implemented and compared. The first is a naive test generator, characterized by its speed but lack of optimality guarantees. The second is a SAT-based test generator, slower but ensuring the generation of the smallest possible test sets. Empirical investigations have been conducted on benchmark reversible circuits, revealing that the test patterns size does not exhibit it significant variations across the spectrum. In 2019, the authors in [48] introduced a Boolean Difference method to extract the test set used for detecting each stuck-at fault and all missing gate faults, such as SMGFs, MMGFs, PMGFs, and RGFs, in reversible circuits. Additionally, an optimization algorithm is applied to construct an optimal test set for detecting all MGFs, along with stuck-at faults, in a given k -CNOT reversible circuit. In 2020, the authors in [49] explored a scheme for constructing the complete test set, which is capable of detecting SMGFs and any consecutive MMGFs in the k -CNOT circuits. The produced complete test set is not minimal. To achieve minimality, the integer linear programming (ILP) problem is applied to the complete test set, which covers 100% fault coverage of SMGFs and MMGFs in a given k -CNOT circuit. In 2021, the study in [50] presented a method for locating and detecting faults in MCT-based circuits taking into account all types of Missing Gate Faults (MGFs). The approach involves using both the original circuit and its complementary version to identify the faults. A comparator is employed to detect discrepancies by comparing the circuit's input mapped to the output of its complemented configuration. In 2021, the authors in [51] proposed a design-for-testability (DFT) technique for k -CNOT reversible circuits, aiming to evaluate the detection process of SMGFs, PMGFs, and MMGFs. In this DFT technique, the first step is to formulate the cluster of the circuit into different sets by partitioning the gates. The second step involves inserting an additional input line for each cluster in the original circuit, which serves as a control input to the gates of their corresponding cluster in the circuit. The third step is used to construct the final test from the DFT circuit for detecting SMGFs, PMGFs and MMGFs, which achieves 100% fault coverage with a low cost overhead for the proposed DFT technique. In 2022, the authors in [52] proposed an approach that involves the computation of test vectors, followed by the execution of an algorithm to generate a unique test set. Following the generation of the test set, it is employed to identify any faults present in the circuit. The method has been particularly tested for SMGF, PMGF, and MMGF. In 2025, the authors in [53] discussed basic and modified algorithms for detecting and determining the exact location of SMGFs, PMGFs and MMGFs in k -CNOT circuits. This work demonstrates that the performance of the modified algorithm is improved for detecting and localizing faults, with a 40% reduction in execution time, as compared to the basic algorithm.

From the above literature survey, we discovered minimal work has been conducted taking all MGFs by a single approach. Thus, we are motivated to develop a technique that can identify different categories (SMGF, MMGF, PMGF, and RGF) without modifying the original circuit. In this work, the approach presented herein is intended to simplify the fault detection process and decrease expenses related to overhead with reconstructing or updating original configuration. This technique is primarily intended for detecting all MGFs, along with reducing the test set size by half.

3. PROPOSED METHOD

The method includes a technique for identifying faults present in the reversible circuit. Firstly, a fault-free reversible circuit is taken as an input. The technique initiates by generating all feasible Binary Encoded Patterns (BEP) according to the line count within the reversible circuit. The method proceeds by converting all patterns into Gray codes, and then pairing each with a distinct adjacent Gray code. Afterwards, hierarchical evaluation is conducted for every gate by using selected gray codes and achieves the test vector that identifies all MGFs. Figure 4 demonstrates the conceptual model of the proposed technique. The segments of the pictorial view of AGCP approach are described below:

1. **Generation of BEP:** Firstly, all probable binary encoded patterns are produced for the reversible circuit. Binary encoded pattern generation depends on how many lines are included in the reversible circuit. With n lines, the circuit yields 2^n distinct binary encoded patterns.

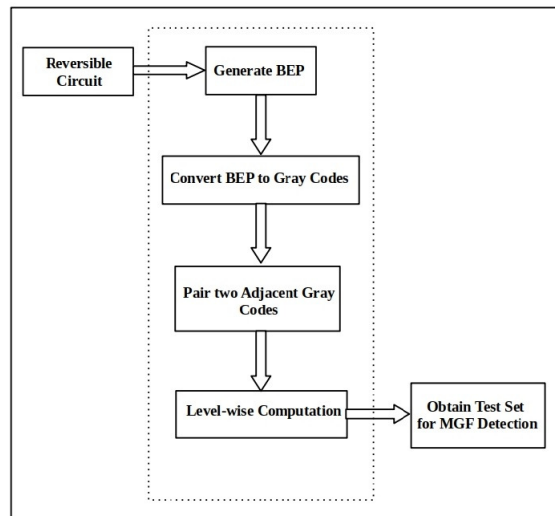


Figure 4. Process flow of an AGCP method

2. **Translation of BEP to their equivalent Gray Codes:** Every generated binary pattern is re-encoded into its corresponding Gray code. The transformation from binary representations to Gray code is carried out using the standard binary-to-Gray conversion technique. The Gray code's MSB remains the same as the MSB of the corresponding binary pattern. An XOR operation between the MSB and the second binary bit yields the subsequent Gray code bit. Then, for each remaining position, the Gray code bit is computed by XORing the current binary bit with its prior bit. Advantages of translating the BEP to gray code are: (i) By limiting changes to just one bit between consecutive numbers, Gray code helps reduce switching activity (ii) Gray code's single bit transitions simplify the circuit while decreasing its cost as well as energy to drive the circuit (iii) The basic operation of reversible circuit is based on XOR operation which is also supported by the gray codes.
3. **Pairing of two adjacent Gray Codes:** The obtained gray codes are made into pairs of two. The pairing of gray codes is done on the non duplicate adjacent gray codes, and each code is used only one time in the pair. Thus, for 2^n binary encoded patterns, there will be 2^n gray codes, where n represents the total source lines of the circuit, and there will be $2^n/2$ quantity of non duplicate adjacent pairs of gray codes.
4. **Perform level-wise computation:** After pairing adjacent gray codes, a level-wise computation is performed. The gray codes containing '1' in the LSB are only considered for the level-wise computation from each pair. It can be figured out that the occurrence of 1 in the LSB position in each pair is for only one gray code; thus, the selection is made from each pair. It is also observed that after selection of the gray codes depending upon the content of '1' in LSB; the selected gray codes produce equal number of even and odd number of ones. The level-wise computation of those selected gray codes is performed. The total count of gray codes is 2^n whereas $2^n/2$ is the count of selected gray codes that perform level-wise computation. The main reasons of selecting '1' in LSB in the method are as follows: (i) LSB changes more frequently as compared to MSB and other bits making it more susceptible to small data alterations and also due to that reason it has more impact on parity bit for effective error detection (ii) Each adjacent gray code pair contains a code with LSB '1' but not by MSB and other bits.

Algorithm 1 presents the method for obtaining the test vectors needed to identify all MGFs. Certain definitions essential to our approach are introduced, and the entire process is depicted in Figure 5.

Definition 1. A variable B_{pi} is the binary encoded patterns $\langle b_1 b_2 \dots b_n \rangle$ in accordance with the n quantity of input lines contained in the circuit, for $0 \leq i \leq 2^n - 1$. Let us consider, $i=2$ and $n=3$, then the binary encoded pattern is presented as $B_{p2}=\langle 010 \rangle$.

Definition 2. A variable G_{ci} is the gray encoded patterns $\langle g_1 g_2 \dots g_n \rangle$ that is generated by the standard binary to gray code conversion process $B_{pi} \Rightarrow G_{ci}$, where $g_i \in \{0, 1\}$, for $1 \leq i \leq n$ and $|B_{pi}| = |G_{ci}|$. Let us

consider the binary encoded pattern $B_{p2}=\langle 010 \rangle$ and the gray encoded pattern $G_{c2}=\langle 011 \rangle$ is generated by the binary to gray code conversion process, i.e., $010 \Rightarrow 011$.

Definition 3. A variable PG_{ck} is denoted as the paired gray code that is formulated by the non-duplicate consecutive adjacent gray encoded patterns G_{ci} , for $1 \leq k \leq 2^n/2$ and n signifies the aggregate count of input lines. Let us consider the two paired gray codes are represented as $PG_{c1}=\{G_{ci}, G_{cj}\}$ and $PG_{c2}=\{G_{cm}, G_{cn}\}$, where $m=j+1$.

Definition 4. The gray encoded pattern $G_{ci}=\langle g_1g_2 \dots g_n \rangle$ behaves as a select gray encoded pattern that is denoted as SPG_{ci} , where $SPG_{ci} \subseteq PG_{ck}$. The selection of the SPG_{ci} is done when the LSB $\{g_1, g_2, \dots, g_n\} \setminus \{g_1, g_2, \dots, g_{n-1}\} \in \{1\}$. Let us consider the $PG_{c1}=\{G_{c1}, G_{c2}\}$, where $G_{c1}=\langle 011 \rangle$ and $G_{c2}=\langle 010 \rangle$. Thus, the $SPG_{c1}=G_{c1}$, i.e., gray encoded pattern $\langle 011 \rangle$.

Definition 5. The test set $C_{TS}=\{B_{p1}, B_{p2} \dots, B_{pt}\}$ is the complete test set, where $t \leq 2^n/2$ and n denotes input lines of the circuit. Let, $B_{pl}=\langle b_1b_2 \dots b_n \rangle$, for $1 \leq l \leq t$, where each $B_{pl} \in SPG_{ci}$. The test set C_{TS} can identify entire probable faults in SMGFs, MMGFs, PMGFs, and RGFs under the MGF model by the proposed AGCP method.

Algorithm 1: C_{TS} GENERATION USING ADJACENT GRAY CODE PAIRING (AGCP)

Input: A reversible circuit consisting of n amount of source lines and N quantity of k -CNOT gates

$G_1, G_2, \dots, G_N$.

$O_{fault-free}[]$: Stores output of fault-free circuit.

$O_{faulty}[]$: Stores output of faulty circuit.

Output: Generation of C_{TS} for MGF's in a given circuit.

- 1 Analyze *fault-free* circuit
 - 2 **for** $i \leftarrow 0$ **to** $(n-1)$ **do**
 - 3 Translate 2^n amount of B_{pi} for n amount of source lines
 - 4 $G_{ci} \leftarrow B_{pi}$
 - 5 $PG_{ck} \leftarrow G_{ci}$
 - 6 $SPG_{ci} \leftarrow PG_{ck}$
 - 7 **for** $j \leftarrow 1$ **to** N **do**
 - 8 Execute processing $G_1, G_2, \dots, G_N$ by SPG_{ci}
 - 9 Obtain $O_{fault-free}[]$ for each SPG_{ci}
 - 10 Inject MGFs to the circuit
 - 11 Re-execute Step 2 to Step 8 for the faulty circuit
 - 12 Obtain $O_{faulty}[]$ for each SPG_{ci}
 - 13 **if** $(O_{fault-free}[] \neq O_{faulty}[])$ **then**
 - 14 Extract C_{TS}
-

The proposed AGCP approach is presented in Algorithm1. Here, we need to represent the variable n , where n indicates the number of input lines present in the k -CNOT circuits. The circuit size N presents the total number of k -CNOT gates, as mentioned in Algorithm 1. The upper bounds on n and N depend on the maximum number of gates (N) performed in an n -bit k -CNOT reversible circuit of the n -input lines. Due to this, each n -bit of the corresponding n -input line is applied to implement the k -CNOT gate operation of the circuit. Therefore, the upper bound on n associated with N is exponential in the k -CNOT based reversible circuits. The stepwise discussion for analyzing the time complexity of the proposed Algorithm 1 is described below.

Operations related to translating the binary encoded pattern inputs (B_{pi}) and storing Gray codes in different forms, such as G_{ci} , PG_{ck} , and SPG_{ci} , are iterated n times. The generation of B_{pi} for n source input lines implies a time complexity of $O(2^n)$ within each iteration, which is mentioned in Steps 2 to 6 in Algorithm 1. In Steps 7 to 9, computations are performed on the circuit gates $G_1, G_2, \dots, G_N$ by the select gray encoded pattern (SPG_{ci}). This computation involves simulating the circuit, which requires N number of k -CNOT gates and involves an $O(N)$ time complexity. The same process is repeated for the faulty circuit

presented in Steps 11 to 12. The simulated comparison between $O_{fault-free}$ and O_{faulty} is executed for extracting the complete test set C_{TS} . This comparison-based computation incurs an additional constant time C . Thus, the overall time complexity is derived as $O(2^n) + O(N) + O(N) + C = O(2^n)$, which is exponential.

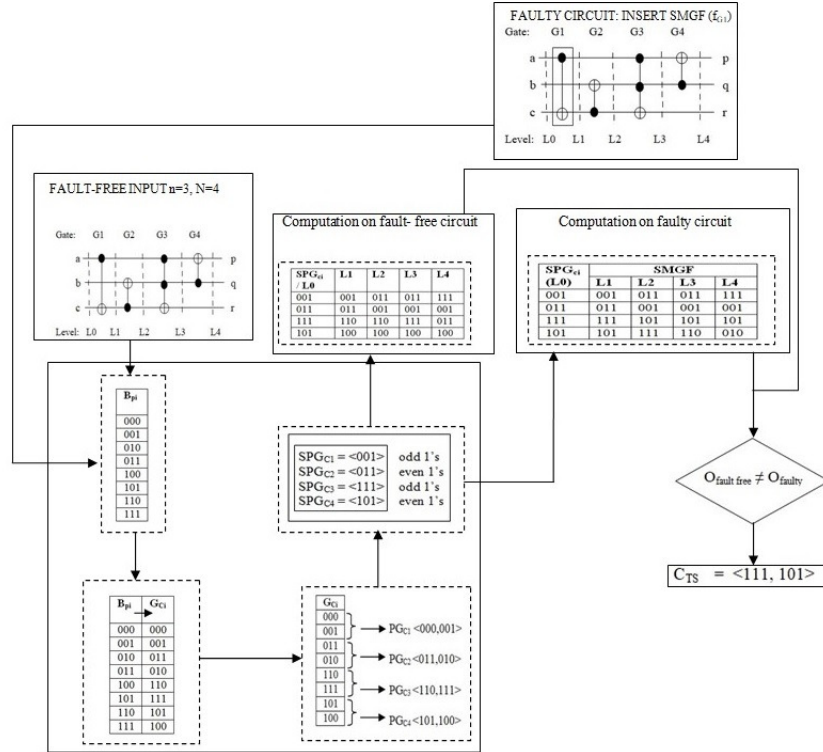


Figure 5. Demonstration of Algorithm 1 by using reversible standard n th_prime circuit

Example 2. To demonstrate our approach in Algorithm 1, we have used a standard reversible circuit n th_prime, as presented in Figure 5. **Step 1** is initiated by the consideration of fault-free circuit that consists of input lines a, b , and c ($n=3$), and four k -CNOT gates ($N=4$) namely G_1, G_2, G_3 and G_4 . Hence, there are five levels from L_0 to L_4 . **Step 2** initiates the 'for loop' which iterates over the circuit lines ($n=3$). **Step 3** is initiated by the translation of all possible binary encoded patterns B_{pi} from 0 to (2^n-1) . Thus, a total number of B_{pi} is 8 for the n th_prime circuit. **In Step 4**, all the generated B_{pi} is converted to their respective gray encoded patterns G_{ci} , as per the Definition 2. **In Step 5**, the paired gray code PG_{ck} is formulated by the non-duplicate consecutive adjacent gray encoded patterns G_{ci} . There are four number of paired gray codes PG_{ck} for eight G_{ci} , namely $PG_{c1} = \{\langle 000 \rangle, \langle 001 \rangle\}$, $PG_{c2} = \{\langle 011 \rangle, \langle 010 \rangle\}$, $PG_{c3} = \{\langle 110 \rangle, \langle 111 \rangle\}$ and $PG_{c4} = \{\langle 101 \rangle, \langle 100 \rangle\}$. **In Step 6**, the select gray encoded pattern SPG_{ci} is determined. Here, the $SPG_{c1} = \langle 001 \rangle$, $SPG_{c2} = \langle 011 \rangle$, $SPG_{c3} = \langle 111 \rangle$ and $SPG_{c4} = \langle 101 \rangle$ are extracted from the paired gray codes $PG_{c1}, PG_{c2}, PG_{c3}$ and PG_{c4} , respectively, as mentioned in the Definition 4. **Step 7** initiates another loop that reads the gate count ($N=4$). **In Step 8**, the level-wise computation is performed by applying each of SPG_{ci} and **Step 9** obtains the primary output at level L_4 of the fault-free circuit and stores in $O_{fault-free}$ [$\langle 111 \rangle, \langle 001 \rangle, \langle 011 \rangle, \langle 100 \rangle$]. **Step 10** is showing the injection of MGFs including SMGF, MMGF, PMGF, and RGF and injection of SMGF at the first gate is outlined in Figure 5. **Step 11** illustrates the reiteration of the same process for the faulty circuit, which are shown in Step 2 to Step 8. **Step 12** extracts the faulty primary output at the primary level L_4 of the circuit with fault and stores in O_{faulty} [$\langle 111 \rangle, \langle 001 \rangle, \langle 101 \rangle, \langle 010 \rangle$] for the SMGF. **In Step 13** performs the fault simulation between $O_{fault-free}$ and O_{faulty} . Finally, complete test set $C_{TS} = \{\langle 111 \rangle, \langle 101 \rangle\}$ is extracted, which is shown in **Step 14**.

Lemma 1. The size of the C_{TS} achieves 50% reduction in the proposed AGCP method for detecting 100% MGFs in the MCT-based reversible circuit.

Proof. The proposed method achieves comprehensive coverage of reversible circuit states through systematic generation of binary encoded patterns B_{pi} and subsequent conversion to gray codes depending upon the amount

of source lines n exist in the circuit. A non-duplicate consecutive adjacent paired gray code PG_{ck} is presented in the suggested method, and the existence of entire feasible PG_{ck} is $2^n/2$. A select gray encoded pattern SPG_{ci} is selected in accordance with the existence of 1 in the LSB position. SPG_{ci} is selected from each paired gray code PG_{ck} . As per the basic characteristics of gray codes, the appearance of $SPG_{ci} \in PG_{ck}$ is only once that containing the 1 at the LSB position. In our method, each $B_{pi} \in SPG_{ci}$ is assigned to the C_{TS} and the size of the $|C_{TS}| = 2^n/2$. Hence, the size of the C_{TS} is achieved by a 50% reduction in the AGCP method. Also the method detects all MGFs including SMGF, MMGF, PMGF, and RGF and thus concludes 100% fault detection of MGFs. $\square$

4. EXPERIMENTAL RESULTS AND DISCUSSION

The empirical findings derived from experiments on multiple reversible MCT circuits for fault detection based on the MGF model are discussed in this section. The proposed AGCP algorithm has been executed in Python 3.4 and runs on a *core-i3* device equipped with 4 gigabytes of RAM, which is operational on Ubuntu v16.04 (64-bit). In our experimental results, we have considered small, medium, and large-sized circuits and a total of 68 standard reversible circuits are taken from [54] and [55]. These circuits are applied in our experimental results to produce a comprehensive test set, along with the required CPU time by the AGCP algorithm for detecting all possible MGFs, which are reported in Table 2. The comparative analysis of the proposed AGCP method with some of the existing approaches outlined in [33, 45] and [48, 51] is reported in Table 3 and Table 4, respectively, with regard to the total number of test vectors and the CPU time to identify the MGFs in reversible circuits. Figure 6 presents a visual representation of the circuits based on fault count and test vector quantity, as determined from the experimental data in Table 2. The value ‘-’ indicates that the result of the circuit is not included in the existing works, which is shown in Table 3 and Table 4.

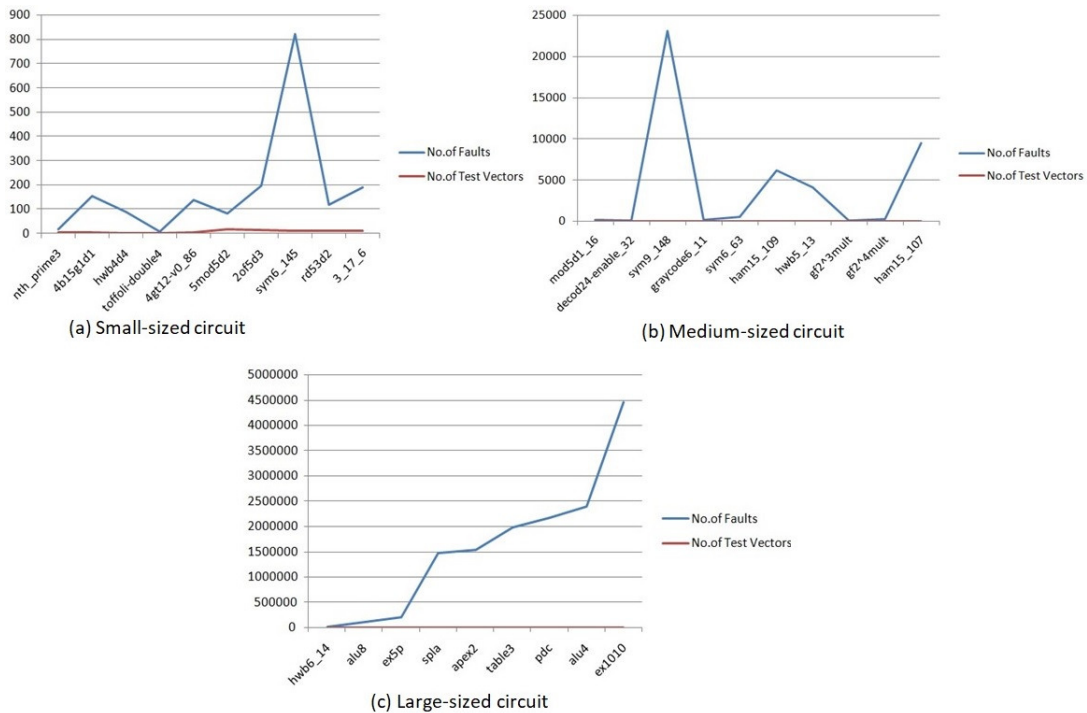


Figure 6. Illustration of various reversible circuits in terms number of faults and test vectors, as reported in Table 2

Table 2 begins with the names of the reversible circuits in the first column. Details of the line count (n) and gate count (N) per circuit are shown in columns 2 and 3. Listed in the 4th column is the total fault count, broken down into SMGF, MMGF, RGF, and PMGF categories. The table's column 5 displays the overall number of test vectors in the C_{TS} designed to detect all MGFs through the AGCP approach. Column

Table 2. Experimental results for identifying all MGFs by AGCP method with CPU Time

Benchmark Circuit	n	N	Total no. of MGFs				C_{TS}	CPU Time (sec)
			SMGF	MMGF	RGF	PMGF		
fredkin-6	3	3	3	3	3	6	2	0.00
ham3d1	3	5	5	10	5	6	4	0.01
nth_prime3	3	4	4	6	4	5	4	0.00
3_17_13	3	6	6	15	6	7	3	0.00
toffoli-double4	4	2	2	1	2	4	3	0.00
mod10-176	4	7	7	21	7	13	7	0.01
hwb4d1	4	17	17	136	17	27	6	0.00
hwb4d3	4	11	11	55	11	14	4	0.06
hwb4d4	4	11	11	55	11	14	2	0.04
hwb4d5	4	13	13	78	13	16	3	0.02
4_49d3	4	12	12	66	12	15	4	0.00
rd32d1	4	4	4	6	4	5	5	0.01
4_49_16	4	16	16	120	16	24	5	0.03
4b15g1d1	4	15	15	105	15	19	6	0.05
4b15g1d2	4	15	15	105	15	20	8	0.00
4mod5d1	5	8	8	28	8	12	8	0.00
4mod5d2	5	9	9	36	9	11	6	0.00
4mod5d3	5	5	5	10	5	6	10	0.01
xor5d1	5	4	4	6	4	4	7	0.01
4gt4-v0.78	5	13	13	78	13	18	11	0.02
4gt12-v0.86	5	14	14	91	14	20	4	0.02
One-two-three -v0.97	5	11	11	55	11	23	3	0.18
mod5d2	5	9	9	36	9	16	6	0.00
rd32_271	5	9	9	36	9	11	5	0.04
mod5d1	5	8	8	28	8	12	7	0.07
5mod5d1	6	17	17	136	17	37	10	0.02
5mod5d2	6	10	10	45	10	18	16	0.05
2of5d1	6	18	18	153	18	24	16	0.13
2of5d3	6	17	17	136	17	29	12	0.09
2of5d2	7	12	12	66	12	19	3	0.04
permanent2x2	6	3	3	3	3	8	6	0.00
majority	6	8	8	28	8	17	6	0.01
3_17_6	7	17	17	136	17	20	9	0.22
rd53d1	7	30	30	435	30	41	7	0.04
rd53d2	7	12	12	66	12	28	10	0.18
rd53d3	8	12	12	66	12	20	3	0.07
rd53d4	7	28	28	378	28	24	11	0.04
sym6_145	7	36	36	630	36	120	9	0.42
ham7d1	7	23	23	253	23	34	3	0.44
ham7d2	7	21	21	210	21	30	6	0.36
ham7d3	7	25	25	300	25	31	9	0.32
mod5d1_16	8	15	15	105	15	19	4	0.00
milller_5	8	16	16	120	16	24	6	0.00
gf2^3mult	9	11	11	55	11	20	5	0.26
gf2^4mult	12	19	19	171	19	35	23	0.77
gf2^5mult	15	29	29	406	29	54	30	0.91
gf2^6mult	18	41	41	820	41	77	19	0.87
decod24- enable_32	9	14	14	91	14	17	7	0.00
mini-alu_84	10	20	20	190	20	27	7	0.02
Sym9_148	10	210	210	21945	210	756	12	0.73
graycode6.11	11	15	15	105	15	10	3	0.27
sym6_63	14	29	29	406	29	43	5	0.03
0410184_169	14	46	46	1035	46	49	5	0.33
rd84_142	15	28	28	378	28	49	9	0.05
4_49_7	15	42	42	861	42	61	11	0.07
ham15_107	15	132	132	8646	132	352	13	0.50
ham15_108	15	70	70	2415	70	125	9	0.07
ham15_109	15	109	109	5886	109	126	18	0.01
hwb5_13	28	88	88	3828	88	131	6	0.32
hwb6_14	46	159	159	12561	159	241	4	0.60
alu_8	91	453	453	102378	453	730	33	2.45
ex5p	206	647	647	208981	647	904	15	2.87
spla	489	1709	1709	1459486	1709	2711	32	3.04
apex2	498	1746	1746	1523385	1746	2787	83	4.04
table3	554	1988	1988	1975078	1988	2997	57	4.19
pdcc	619	2080	2080	2162160	2080	3135	63	5.06
alu4	541	2186	2186	2388205	2186	3390	44	4.37
ex1010	670	2982	2982	4444671	2982	4543	23	5.72
Average			227	210715	227	356	11.95	0.59

6 indicates the CPU processing time (in seconds) needed to generate the C_{TS} for each circuit. In Table 2, the average test vectors and CPU time from the proposed AGCP method obtained are 11.95 and 0.59, respectively. The findings presented in Table 2 highlight that the fault quantity is notably larger (average of SMGFs, MMGFs, RGFs, and PMGFs are found as 227, 210715, 227, 356, respectively) than the set of test vectors used to detect the associated missing gate faults. It is because the computation mechanism is reduced to 50%. Thus, the test set size is reduced to half of the original test set. After analyzing Table 2, it is found in average cases that the generated C_{TS} increases with the increasing number of lines of the circuit. Also, the CPU time increases with the number of lines of the circuit. The highest CPU time is observed in *ex1010* (where $n = 670$) with 5.72 seconds. Figure 6 (a), (b), and (c) demonstrate the fault instances and test vectors for some small-sized circuits, medium-sized circuits, and large-sized circuits, respectively. Table 2 contains a total of 40 small-sized circuits that have been tested, and it is found that the number of faults contained in *sym6_145* is very high. A total of 19 numbers of medium-sized circuits have been tested and found the highest fault content for *sym9_148*. Similarly, a total of 9 large-sized circuits have been tested and the highest fault count is observed in the *ex1010* reversible circuit.

Table 3. Experimental results of comparison of total number of faults and test vectors with [33] and [45]

Benchmark Circuit	n	N	Total No. of Faults			Total No. of Test Vectors		
			[33]	[45]	[Proposed]	[33]	[45]	[Proposed]
4_49_16	4	16	40	-	176	8	-	5
4gt4-v0.78	5	13	31	18	122	16	5	11
4gt12-v0.86	5	14	34	20	139	16	4	4
One-two-three_v0.97	5	11	-	23	100	-	3	3
3_17_6	7	17	37	20	190	4	5	9
rd53_131	7	28	52	24	458	16	11	11
mod5d1_16	8	15	34	19	154	4	5	4
miller_5	8	16	40	24	176	6	5	6
decod24-enable 32	9	14	31	17	136	4	3	7
mini-alu_84	10	20	47	27	257	4	6	7
Sym9_148	10	210	-	756	23121	-	23	12
graycode6_11	11	15	-	10	145	-	1	3
sym6_63	14	29	72	43	507	6	11	5
rd84_142	15	28	77	49	483	5	16	9
4_49_7	15	42	103	61	1006	6	7	11
ham15_107	15	132	484	-	9482	16	-	13
ham15_108	15	70	195	-	2680	16	-	9
ham15_109	15	109	235	-	6230	16	-	18
hwb5_13	28	88	219	131	4135	6	11	6
hwb6_14	46	159	400	241	13120	6	13	4
alu_8	91	453	-	730	104014	-	27	33
ex5p	206	647	-	904	211179	-	42	32
spla	489	1709	-	2711	1465615	-	42	32
apex2	498	1746	-	2787	1531969	-	70	83
table3	554	1988	-	2997	1982051	-	49	57
pdc	619	2080	-	3135	2169455	-	49	63
alu4	541	2186	-	3390	2395967	-	38	44
ex1010	670	2982	-	4543	4455178	-	27	23

Table 3 presents a comparative evaluation between the formulated AGCP method and the existing approaches outlined in [33] and [45]. Columns 1, 2, and 3 of Table 3 show a similar presentation to Table 2. As shown in Table 3, columns 4 through column 6 list the number of faults evaluated in [33, 45], and the approach presented in this study. Correspondingly, columns 7, 8 and 9 of Table 3 show the total test vector counts used for identifying faults in [33, 45], and the proposed AGCP method. It is observed that the fault coverage of our method is higher compared to [33, 45], as it considers all types of faults under the MGF model. Moreover, the magnitude of the test vectors of the formulated method is less than or equal to detect the faults in maximum cases, as compared to existing methods. It is observed that the *apex2* circuit has the peak test vector count required to detect all MGFs using the suggested method, and the performance of *4gt - 12_v0.86* and *sym6_63* is found better than [33, 45], as per the consideration of test vector requirement. Almost 33 times and 980 times more faults are considered in our work compared to work in [33] for the circuit *hwb6_14* and the work in [45] for the circuit *ex1010*. After comparing the works of [33] and [45], we observed that our proposed method is more effective in detecting faults and various types of MGFs. Additionally, the test vector count is less as the reduction of computation overhead is introduced in our method. Therefore, these are the two main strengths of

this proposed AGCP approach.

Table 4. Experimental results of comparison of total number of faults and test vectors with [48] and [51]

Benchmark Circuit	n	N	Total No. Faults			Total No. of Test Vectors		
			[48]	[51]	[Proposed]	[48]	[51]	[Proposed]
ham3d1	3	5	26	21	26	3	10	4
nth_prime3	3	4	-	15	19	-	7	4
3_17_13	3	6	34	-	34	3	-	3
hwb4d1	4	17	197	-	197	6	-	6
hwb4d3	4	11	91	-	91	4	-	4
hwb4d4	4	11	91	-	91	4	-	2
hwb4d5	4	13	120	-	120	4	-	3
4_49_16	4	16	176	-	176	6	-	5
4b15g1d1	4	15	-	139	154	-	17	6
4b15g1d2	4	15	-	140	155	-	16	8
4b15g2d1	4	15	-	141	156	-	18	7
4b15g2d2	4	15	-	139	154	-	16	7
4b15g3d1	4	15	-	140	155	-	18	5
4b15g3d2	4	15	-	138	153	-	16	4
4b15g4d1	4	15	-	139	154	-	17	3
4b15g4d2	4	15	-	139	154	-	16	5
4b15g5d1	4	15	-	139	154	-	16	5
4b15g5d2	4	15	-	140	155	-	15	5
4mod5d1	5	8	-	48	56	-	4	8
4mod5d2	5	9	-	56	65	-	4	6
4mod5d3	5	5	-	21	26	-	6	10
4mod5d4	5	5	-	21	26	-	7	8
4mod5bdd	7	8	-	32	38	-	13	4
4mod5esop	5	5	-	21	26	-	7	9
xor5d1	5	4	18	14	18	2	6	7
mod5d2	5	9	65	-	65	4	-	2
rd32_271	5	9	-	56	65	-	6	5
mod5d1	5	8	56	48	56	5	7	7
5mod5d1	6	17	207	190	207	7	6	10
5mod5d2	6	10	-	73	83	-	10	16
5mod5d3	6	8	-	50	58	-	12	15
5mod5d4	6	10	-	69	79	-	13	11
mod5adder	6	21	279	-	279	9	-	6
2of5d1	6	18	213	195	213	11	13	16
2of5d3	6	17	199	182	199	7	12	12
2of5d2	7	12	109	97	109	4	17	3
permanent2x2	6	3	-	14	17	-	5	6
majority	6	8	-	53	61	-	8	6
rd53d1	7	30	536	506	536	10	29	7
rd53d2	7	12	118	106	118	10	11	10
rd53d3	8	12	110	98	110	4	17	3
rd53d4	7	28	458	430	458	10	11	11
rd53d5	7	16	179	163	179	6	12	9
sym6_145	7	36	-	786	822	-	6	9
ham7d1	7	23	333	310	333	5	21	3
ham7d2	7	21	282	261	282	5	19	6
ham7d3	7	25	381	356	381	4	18	9
rd53esop	8	8	-	56	64	-	8	7
gf2 ³ mult	9	11	97	86	97	4	16	5
gf2 ⁴ mult	12	19	-	225	244	-	24	23
rd53bdd	13	34	-	673	707	-	31	25

Table 4 represents the comparison of the result of the proposed AGCP method with [48] and [51]. The first three column of Table 4 depicts the same representation as Table 2 and Table 3. Columns 4, 5 and 6 show the total number of faults considered in [48], [51] and the proposed method, and the last three columns present the total number of test vectors required for detecting the respective faults in [48], [51] and the proposed method, respectively. In this table, the experimental findings of a total of 53 reversible circuits are mentioned. It is observed that the total fault count considered in [48] is exactly similar to the fault count considered in the proposed method for every circuit shown in Table 4; however, it is less in the existing work [51]. It is found that the number of test vectors required to detect all MGFs by our method is less in *hwb4d4*, *hwb4d5*, *4_49_16*, *mod5d2*, *mod5adder* as compared to [48] and it is also less for a total of 32 circuits as compared

to [51]. Test vector count in the proposed method is less or equal, as compared to both [48] and [51] in the circuits namely *2of5d2*, *rd53d1*, *rd53d2*, *rd53d3*, *rd53d4*, *ham7d1*, *gf2³mult*. Thus, the performance of the proposed method is found to be good as compared to [51] despite considering a larger number of faults. Though the fault consideration is similar in [48] to our method, it is observed that only small-sized circuits have been considered in [48]. In our method, we have considered experimental findings for small, medium, and large circuits, as shown in Table 2.

5. CONCLUSION

The main purpose of this study can be summarized as developing a single technique for single, multiple, repeated, and partial missing gate fault models and reducing the computation speed of the working of the circuits for this detection technique. It holds the importance of the study. This work outlines an approach for recognizing various faults based on the MGF model of reversible circuit by performing computation by half of the selected test vectors, which reduces the effort of the computation process. The computation overhead is reduced by using the concept of pairing and selecting the most efficient one from the pairs. The consideration of all the different types of missing gate faults helps in increasing the number of detectable faults, which is another challenging task.

REFERENCES

- [1] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.
- [2] H. F. Chau and H. K. Lo, "One-way functions in reversible computations," *Cryptologia*, vol. 21, no. 2, pp. 139–148, 1997.
- [3] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, 1961.
- [4] C. H. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.
- [5] M. B. Plenio and V. Vitelli, "The physics of forgetting: Landauer's erasure principle and information theory," *Contemporary Physics*, vol. 42, no. 1, pp. 25–60, 2001.
- [6] M. P. Frank, "The physical limits of computing," *Computing in Science & Engineering*, vol. 4, no. 3, pp. 16–26, 2002.
- [7] A. Sarker, H. M. Hasan Babu, and S. M. M. Rashid, "Design of a dna-based reversible arithmetic and logic unit," *IET nanobiotechnology*, vol. 9, no. 4, pp. 226–238, 2015.
- [8] M. Sarkar, P. Ghosal, and S. P. Mohanty, "Minimal reversible circuit synthesis on a dna computer," *Natural Computing*, vol. 16, pp. 463–472, 2017.
- [9] M. Rofail and A. Younes, "Synthesis strategy of reversible circuits on dna computers," *Symmetry*, vol. 13, no. 7, p. 1242, 2021.
- [10] S. Kotiyal, H. Thapliyal, and N. Ranganathan, "Design of reversible adder-subtractor and its mapping in optical computing domain," in *transactions on Computational Science XXIV: Special Issue on Reversible Computing*. Springer, 2014, pp. 37–55.
- [11] D. G. S. Rao, S. Swarnakar, and S. Kumar, "Design of all-optical reversible logic gates using photonic crystal waveguides for optical computing and photonic integrated circuits," *Applied Optics*, vol. 59, no. 35, pp. 11 003–11 012, 2020.
- [12] H. Thapliyal and M. Zwolinski, "Reversible logic to cryptographic hardware: a new paradigm," in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, vol. 1. IEEE, 2006, pp. 342–346.

- [13] V. Raj, S. Janakiraman, S. Rajagopalan, and R. Amirtharajan, "Security analysis of reversible logic cryptography design with lfsr key on 32-bit microcontroller," *Microprocessors and Microsystems*, vol. 84, p. 104265, 2021.
- [14] 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, pp. 297–311, 2008.
- [15] T. N. Sasamal, A. K. Singh, and A. Mohan, "Efficient design of reversible alu in quantum-dot cellular automata," *Optik*, vol. 127, no. 15, pp. 6172–6182, 2016.
- [16] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jezwiak, A. Coppola, and B. Massey, "Regular realization of symmetric functions using reversible logic," in *Proceedings Euromicro Symposium on Digital Systems Design*. IEEE, 2001, pp. 245–252.
- [17] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "Scaffcc: A framework for compilation and analysis of quantum computing programs," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, 2014, pp. 1–10.
- [18] 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, 2020, pp. 126–131.
- [19] M. Arabzadeh, M. Saheb Zamani, M. Sedighi, and M. Saeedi, "Depth-optimized reversible circuit synthesis," *Quantum information processing*, vol. 12, pp. 1677–1699, 2013.
- [20] M. Mohammadi and M. Eshghi, "Heuristic methods to use don't cares in automated design of reversible and quantum logic circuits," *Quantum Information Processing*, vol. 7, pp. 175–192, 2008.
- [21] R. Montaser, A. Younes, and M. Abdel-Aty, "Improving the quantum cost of nct-based reversible circuit," *Quantum Information Processing*, vol. 14, pp. 1249–1263, 2015.
- [22] K. Datta, G. Rath, I. Sengupta, and H. Rahaman, "Exact synthesis of reversible circuits using a* algorithm," *Journal of The Institution of Engineers (India): Series B*, vol. 96, pp. 121–130, 2015.
- [23] H. Jayashree, H. Thapliyal, H. R. Arabnia, and V. K. Agrawal, "Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier," *The Journal of Supercomputing*, vol. 72, pp. 1477–1493, 2016.
- [24] H. M. Gaur, A. K. Singh, and U. Ghanekar, "Offline testing of reversible logic circuits: an analysis," *Integration*, vol. 62, pp. 50–67, 2018.
- [25] M. Abramovici, M. A. Breuer, A. D. Friedman *et al.*, *Digital systems testing and testable design*. Computer science press New York, 1990, vol. 2.
- [26] 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.
- [27] M. Ibrahim, A. R. Chowdhury, and H. M. H. Babu, "Minimization of cts of k-cnot circuits for ssf and msf model," in *2008 IEEE international symposium on defect and fault tolerance of VLSI systems*. IEEE, 2008, pp. 290–298.
- [28] M. Handique, J. K. Deka, S. Biswas, and K. Dutta, "Minimal test set generation for input stuck-at and bridging faults in reversible circuits," in *TENCON 2017-2017 IEEE Region 10 Conference*. IEEE, 2017, pp. 234–239.
- [29] H. M. Gaur, A. K. Singh, and U. Ghanekar, "Design for stuck-at fault testability in multiple controlled toffoli-based reversible circuits," *Defence Science Journal*, vol. 68, no. 4, 2018.
- [30] H. Rahaman, D. K. K. D. K. 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.

- [31] P. Sarkar and S. Chakrabarti, "Universal test set for bridging fault detection in reversible circuit," in *2008 3rd international design and test workshop*. IEEE, 2008, pp. 51–56.
- [32] I. Hayes, John P and B. Becker, "Testing for missing-gate faults in reversible circuits," in *13th Asian test symposium*. IEEE, 2004, pp. 100–105.
- [33] A. Kole, R. Wille, K. Datta, and I. Sengupta, "Test pattern generation effort evaluation of reversible circuits," in *Reversible Computation: 9th International Conference, RC 2017, Kolkata, India, July 6-7, 2017, Proceedings 9*. Springer, 2017, pp. 162–175.
- [34] 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.
- [35] M. Handique and H. K. Sarma, "Testable design for positive control flipping faults in reversible circuits," *Indonesian Journal of Electrical Engineering and Informatics (IJEET)*, vol. 11, no. 2, pp. 416–430, 2023.
- [36] 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.
- [37] 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.
- [38] M. Zamani and M. B. Tahoori, "Online missing/repeated gate faults detection in reversible circuits," in *2011 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems*. IEEE, 2011, pp. 435–442.
- [39] T. Toffoli, "Reversible computing," in *International colloquium on automata, languages, and programming*. Springer, 1980, pp. 632–644.
- [40] R. P. Feynman and T. Hey, "Quantum mechanical computers," in *Feynman Lectures on Computation*. CRC Press, 2023, pp. 169–192.
- [41] 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.
- [42] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," *Phys. Today*, vol. 54, no. 2, p. 60, 2001.
- [43] H. Rahaman, D. K. Kole, D. K. Das, and B. B. Bhattacharya, "On the detection of missing-gate faults in reversible circuits by a universal test set," in *21st international conference on VLSI design (VLSID 2008)*. IEEE, 2008, pp. 163–168.
- [44] H. Zhang, S. Frehse, R. Wille, and R. Drechsler, "Determining minimal testsets for reversible circuits using boolean satisfiability," in *IEEE Africon'11*. IEEE, 2011, pp. 1–6.
- [45] R. Wille, H. Zhang, and 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.
- [46] M. Zamani, M. B. Tahoori, and K. Chakrabarty, "Ping-pong test: Compact test vector generation for reversible circuits," in *2012 IEEE 30th VLSI Test Symposium (VTS)*. IEEE, 2012, pp. 164–169.
- [47] A. P. Surhonne, A. Chattopadhyay, and R. Wille, "Automatic test pattern generation for multiple missing gate faults in reversible circuits: Work in progress report," in *Reversible Computation: 9th International Conference, RC 2017, Kolkata, India, July 6-7, 2017, Proceedings 9*. Springer, 2017, pp. 176–182.
- [48] J. Mondal, B. Mondal, D. K. Kole, H. Rahaman, and D. K. Das, "Boolean difference technique for detecting all missing gate and stuck-at faults in reversible circuits," *Journal of Circuits, Systems and Computers*, vol. 28, no. 12, p. 1950212, 2019.

- [49] 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, no. 1, pp. 105–122, 2020.
- [50] 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, p. 370, 2021.
- [51] 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. 06, p. 2150094, 2021.
- [52] B. Mondal, C. Bandyopadhyay, A. Bhattacharjee, D. Roy, S. Parekh, and H. Rahaman, "An approach for detection and localization of missing gate faults in reversible circuit," *IETE Journal of Research*, vol. 68, no. 5, pp. 3607–3627, 2022.
- [53] B. Mondal, S. Datta, and S. Sarkar, "A modified technique for the detection and exact localization of missing gate faults in reversible circuits," *IETE Journal of Research*, pp. 1–17, 2025.
- [54] D. Maslov, "Reversible logic synthesis benchmarks page," <http://www.cs.uvic.ca/maslov/>, 2005.
- [55] 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.

BIOGRAPHIES OF AUTHORS



Dimpimoni Kalita completed her Bachelor's degree in Computer Science and Engineering (CSE) from Gauhati University, Guwahati, Assam with honours in the year 2015. She earned her Master's degree in CSE from Assam University, Silchar, Assam, with Gold Medal in the year 2018. Currently, she is pursuing PhD from the Department of CSE, TSSOT, Assam University, Silchar, Assam, India. Her research focuses on reversible and quantum computing.



Mousum Handique     obtained his Bachelor of Engineering (B.E.) in Computer Technology and Master of Technology in Information Technology from Nagpur University, India, in 2002 and Tezpur University, Tezpur, Assam, India, in 2005, respectively. Subsequently, he obtained his Ph.D. in VLSI Testing and Reversible computing from Indian Institute of Technology Guwahati (IITG), Assam, India. Presently, he is working as an Associate Professor in the Department of Computer Science and Engineering, TSSOT, Assam University, Silchar, India. His research focuses on Formal System Verification, Machine Intelligence, and Testing and Synthesis of Reversible and Quantum Circuits. He has served as a reviewer in many journals, like IEEE, IETE, IJEEI, JETA etc.