Detection and Identification of Gate Faults in Reversible Circuit

B. Mondal, C. Bandyopadhyay, A. Bhattacharjee and H. Rahaman

Abstract In recent time, efficient implementation of reversible logic circuits has come out as an important research area before the design industry. With the advancement in reversible logic synthesis, developing mechanism for identification of faults finds importance. Though there exist well-known testing techniques, but developing improved testing algorithms is the need of the hour. Aiming to develop efficient testing technique, here in this work, we show an improved testing scheme based on Boolean logic function. Two different testing approaches are presented here, where in the first work, by using Boolean difference method SMGFs in reversible circuit are tracked successfully, where a test vector generator is derived to find the faults. In the second work, a Reed–Muller (RM) form based testing approach is developed that not only detects the faults but also locates the exact position of the faulty area. A limitation for the second testing scheme is that it can only be employed over a specific type of reversible circuit known as Exclusive-Or Sum-Of-Product (ESOP) design. Both the testing techniques have been executed over different benchmark suites and a comparative study with state-of-the-art testing approaches have been included in the work.

B. Mondal (✉) · C. Bandyopadhyay · A. Bhattacharjee · H. Rahaman
Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur 711103, India
e-mail: bappa.arya@gmail.com

C. Bandyopadhyay
e-mail: chandanb.iiest@gmail.com

A. Bhattacharjee
e-mail: anirbanbhattacharjee330@gmail.com

H. Rahaman
e-mail: hafizur@it.iiests.ac.in

A. K. Singh et al. (eds.), *Design and Testing of Reversible Logic*, Lecture Notes in Electrical Engineering 577, https://doi.org/10.1007/978-981-13-8821-7_10

1 Introduction

Heat dissipation is considered as the essential concern in modern day's VLSI circuit. As per Launder's principles [1], loss of information generates *KTlog$_2$joule* amount of heat, where *k* is Boltzmann constant and *T* is absolute temperature. Hence, alternative technology is required so that heat generation can be minimized in the circuit. Bennet [2] claimed that dissipation of energy can be made zero only when the circuit is constructed with reversible gates. Therefore, reversible logic design is considered as the prerequisite needed to minimize heat dissipation during logic computation. On the other side, as the quantum circuit [3] follows the principle of reversibility, the implementation of quantum functionality using reversible circuit is possible. Reversible circuit not only has the dominance in the field of quantum circuit design, but it too has applications in adiabatic computing [4, 5], Cryptography and Optical Computing. In the last couple of years, several progresses have been made on efficient design strategies of reversible circuit.

Synthesis algorithms have been developed for making the designs of reversible circuit generic. But, not only designing the cost-efficient circuits get the high importance but simultaneously developing testing algorithms [6–11] for checking the correctness of such designs find popularity. In recent time, some promising works on efficient testing strategies have been developed where improved algorithms are formulated to make the testing process easier.

Here, in this work, we show two different approaches to find faults in reversible circuit. In the first work, a Boolean difference-based testing technique is developed, where a Boolean generator is formulated to produce test vectors and then the faults are tracked. This approach is very generic as it can be employed over any type of circuits. The second testing scheme is not very generic like the first one as it can only operate over ESOP-based designs. In this testing scheme, the functional power of Reed–Muller expression is used to find and locate the faults.

The remaining portion of the article is formulated as follows: preliminaries associated with reversible testing are stated in Sect. 2. Section 3 summarizes previous research works on reversible testing. The developed methodologies are presented in Sect. 4. The experimental data of our work are summarized in Sect. 5. Finally, the chapter is concluded in Sect. 6.

2 Preliminaries

2.1 *Reversible Circuits and Gates*

Definition 1 A circuit C_{nf} over a set of circuit lines $L = \{c_1, c_2, \ldots, c_n\}$ is said to be reversible if it satisfies the following three cri*teria*:

(i). input (m) lines are equal with the output (n) lines

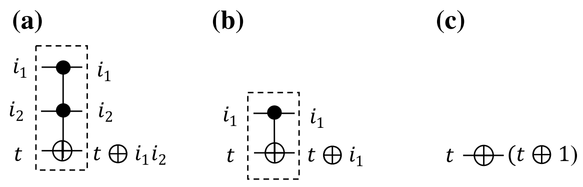


Fig. 1 **a** 2-control Toffoli gate, **b** *CNOT* gate, **c** *NOT* gate

(ii). if the circuit is fan-out free
(iii). circuit consists of reversible gates only.

Definition 2 A reversible gate G can be described as G(C; T), where parameters C, T represents the control and target connection inputs. In that gate G, the control input set C may contain an empty value but the set T must have a minimum of one target line in such that $C \cap T = \Phi$.

In classical circuit different logic gates are used to implement a circuit, similarly there are well-known reversible gates like Toffoli [12], Fredkin [13], Feynman [14] that are used to construct reversible circuits. Some examples of reversible gates are depicted in Fig. 1.

2.2 ESOP-Based Design

A reversible circuit may have different designs and such variations in design depend on the type of algorithm deployed or heuristic employed. Among the several design models, due to the scalable feature property, ESOP (Exclusive Sum-Of-Products) [15]-based representation has been found as one of the widely used design model for reversible circuit. Now in the following, we introduce this special type of circuit.

ESOP can be represented in the form of Sum-Of-Products (SOP) form except that the SOP product terms which are separated by '+' operator, it is separated by "$\oplus$" operator. To express any n-input, m-output reversible function in ESOP representation, it requires $(n + m)$ numbers of input lines in the circuit, where n represents the control set and the rest m lines operate as functional output lines.

Cube list a special data structure from which the ESOP designs are formed. Such cube list contains the detail gate specification and control structure for the ESOP circuit. Each cube in the cube list denotes a gate in the design. For an ease of understanding, cube list and its corresponding ESOP expression are given in Fig. 2a and b, where it can be seen that each of the cubes from the list has been mapped to an equivalent gate and finally a complete design is formed Fig. 2b.

(a)

a_1	a_2	a_3	a_4
–	1	1	–
–	–	–	1
1	–	–	–
0	1	–	1
0	1	1	0

(b)

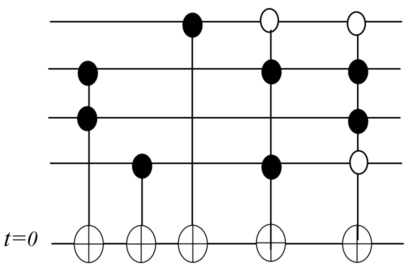


Fig. 2 **a** Cubelist for $f(a_1, a_2, a_3, a_4) = a_2a_3 \oplus a_4 \oplus a_1 \oplus \bar{a}_1a_2a_4 \oplus \bar{a}_1a_2a_3\bar{a}_4$, **b** ESOP expression of Fig. 2a

2.3 Reed–Muller Form

For efficient design and testing of reversible circuit, the concept of Boolean algebra operator is used widely. The modulo-2 arithmetic is applied and any Boolean function can be realized using this algebra. For implementation purpose, Reed–Muller expansion can be represented using sum-of-products expression of modulo-2 arithmetic. Reversible circuit based on module-2 expansion can be realized using only exclusive OR (EXOR) gates. For any Boolean function $f(x_1x_2 \ldots x_n)$, it is expressed in the form of Reed–Muller expansion [16, 17].

PPRM: The positive polarity based Reed–Muller (PPRM) expression can be realized in the form of an EXOR canonical sum-of-products expression in which each variable represents a positive polarity (un-complimented). A PPRM expression of *n* variable can be expressed as

$\mathrm{f}(\mathrm{x}_1\mathrm{x}_2 \ldots \mathrm{x}_\mathrm{n}) = \mathrm{a}_0 \oplus \mathrm{a}_1\mathrm{x}_1 \oplus \mathrm{a}_2\mathrm{x}_2 \oplus \mathrm{a}_3\mathrm{x}_1\mathrm{x}_2 \oplus \ldots \oplus \mathrm{a}_{2^\mathrm{n}-1}\mathrm{x}_1\mathrm{x}_2 \ldots \mathrm{x}_\mathrm{n}$, where $\mathrm{a}_\mathrm{i} \in \{0, 1\}$. The variable a_i in the above expression represents coefficient vector, where x_i denotes input variables. If the coefficient becomes zero, then the corresponding product term is not present in the PPRM expression otherwise the product term is included in the given expression.

FPRM: In fixed polarity Reed–Muller (FPRM) expression, the *n* variable function can be represented as

$\mathrm{f}(\mathrm{x}_1\mathrm{x}_2 \ldots \mathrm{x}_\mathrm{n}) = \mathrm{a}_0 \oplus \mathrm{a}_1\dot{\mathrm{x}}_1 \oplus \mathrm{a}_2\dot{\mathrm{x}}_2 \oplus \mathrm{a}_3\dot{\mathrm{x}}_1\dot{\mathrm{x}}_2 \oplus \ldots \oplus \mathrm{a}_{2^\mathrm{n}-1}\dot{\mathrm{x}}_1\dot{\mathrm{x}}_2 \ldots \dot{\mathrm{x}}_\mathrm{n}$ where $\mathrm{a}_\mathrm{i} \in \{0, 1\}$ and $\dot{\mathrm{x}} \in \{\mathrm{x}, \bar{\mathrm{x}}\}$, where x denoted un-complemented literals and $\bar{\mathrm{x}}$ denotes complemented literals.

GRM/MPRM: The Mixed polarity Reed–Muller (MPRM) expression can be considered as the generalization of FPRM expression in which there is hardly any limitation in the polarity of each variable. The Boolean function having *n* variable can be represented by a number of $2^{\mathrm{n}2^{\mathrm{n}-1}}$ GRM form. The MPRM expression for n variable function is represented as

$f(x_1x_2 \ldots x_n) = a_0 \oplus a_1\dot{x}_1 \oplus a_2\dot{x}_2 \oplus a_3\dot{x}_1\dot{x}_2 \oplus \ldots \oplus a_{2^n-1}\dot{x}_1\dot{x}_2 \ldots \dot{x}_n$ where $a_i \in \{0, 1\}$ and $\dot{x} \in \{x, \bar{x}\}$.

The association among various Reed–Muller configurations can be represented as PPRM ⊂ FPRM ⊂ GRM/MPRM.

2.4 Different Fault Models and Their Properties

Faults in a circuit may originate due to several reasons [18–20]. Depending on the type of errors, there are four types of faults such as—single missing gate fault (SMGF), repeated gate fault (RGF), partial missing gate fault (PMGF) and multiple missing gate fault (MMGF).

Definition 3 Complete disappearance of a gate from a given circuit results in a single missing gate fault.

The Fig. 3a shows an SMGF in a benchmark circuit *ham3\design#*, where the faulty area has been marked with a dotted box. In the dotted region, the first 2-CNOT gates are missing. A SMGF fault can be detected by providing value 1 to all the control line set of the gate and either 0 or 1 at the target node of corresponding gate.

Definition 4 A fault can be considered as repeated gate fault if the same gate consecutively reappears in the design and may change the functionality of the circuit.

In Fig. 3b, the RGF is shown in the first gate of the *ham3\design#1*. It can be ascertained that if a gate reappears even number of times then its effect becomes equivalent to an SMGF fault while odd occurrence makes the RGF fault as redundant indicating the circuit functionality remains unchanged.

Definition 5 Disappearance of control connection input of a gate results in a fault in a circuit known as partial missing gate fault (PMGF).

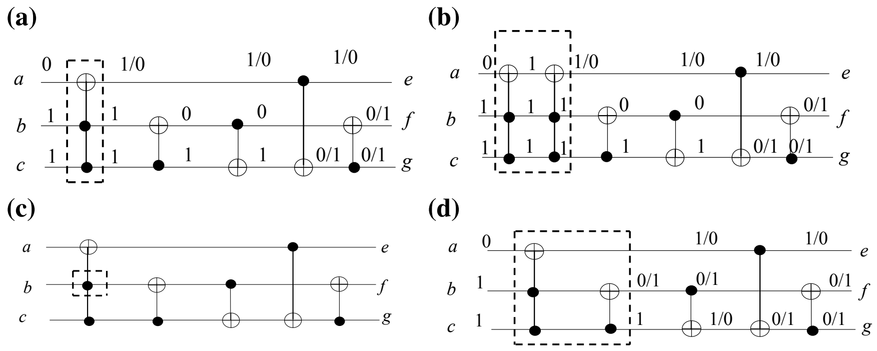


Fig. 3 **a** SMGF fault in circuit*ham3\design#*, **b** RGF in *ham3\design#1* circuit, **c** PMGF in circuit*ham3\design#1*, **d** MMGF in *ham3#design#1*

In Fig. 3c, a PMGF fault is shown in the first gate of the circuit. It is considered that a PMGF fault can only be uncovered by applying value 0 to at the missing control inputs and a value 1 is set for other control lines.

Definition 6 Missing of two or more successive gates from the circuit generates a fault known as multiple missing gate fault (MMGF).

Consider the circuit shown in Fig. 3d, where the first two gates enclosed within the dotted box are missing.

Still so far we have seen the fundamentals related to reversible circuit and its associated fault models. Now, here we are discussing some of the works in testing and highlighting their contributions.

3 Related Works and Contributions

An efficient approach of stuck-at-fault detection by the adaptive tree-based technique is proposed in [6]. Here, in this technique at first the fault in half of the circuit is detected and then by applying the reversible property, a mirror image of the remaining part of the circuit is developed to detect faults in the remaining part of the circuit.

Furthermore, a generalized approach of "stuck-at" fault detection for all k-CNOT based circuit has been reported in [7], where a universal test of size 3 has been used for the fault detection.

Taking one step more, a novel technique for fault detection is shown in [8], where an (n $\times$ n) reversible circuit constructed with k-CNOT gates is tested for possible faults. In this method, a testable design has been developed by copying gates along with an additional line. Though some overhead is incurred to transform the original circuit into the testable form, but the modified testable design becomes very sufficient and easy to detect all existing fault models in the given circuit.

To make the faults detection easier, not only fault specific test vectors have been generated but the way of making simpler testable designs also have been explored and such a work is reported in [9], where extra inputs and additional k-CNOT gates are added in the design to make the design testing friendly. This modified testing design methodology determines a universal test set of size $(n + 2)$ and thereby identifies the said faults in the given circuit.

Instead of applying huge number of test vectors in fault detection and also to reduce the design complexity, testing of SMGFs and RGFs and MMGFs in reversible circuit with minimum number of test vectors is presented in [10].

4 Proposed Testing Methods

Here we state both the testing schemes with examples. The first testing scheme is based on Boolean difference method, whereas the second one relies on Reed–Muller expansion.

4.1 Boolean-Based Testing Method1

Here we state the technique to determine the existence of SMGF in a circuit. The proposed approach is divided into three phases. At first, Boolean difference of the circuit is computed for each of the missing gate. In the second phase, a Boolean generator for the entire circuit is constructed and in the third phase, test vectors are constructed from the Boolean generator which finally checks the presence of SMGF in the circuit. All three phases are stated next in detail.

4.1.1 Computation of the Boolean Difference for Missing Gate Faults

Let us assume a reversible circuit having *n* number of input lines and *N* number of gates.

Definition 7 The Boolean expression generated at the *j*th line of a fault-free circuit is known as the $\mathrm{OriginalExpression}_j$, where $(0 \le j \le n-1)$. For any given reversible circuit with n inputs can be represented by $\mathrm{OriginalExpression}_0$, $\mathrm{OriginalExpression}_1$, $\mathrm{OriginalExpression}_{n-1}$.

Definition 8 For a faulty circuit, the Boolean expression produced at the *j*th line as a result of the gate missing from the *i*th level of circuit can be represented as the $\mathrm{FaultyExpression}_{i,j}$, where $(0 \le i \le N-1)$, $(0 \le j \le n-1)$.

For any reversible circuit with *n* lines can be expressed as $FaultyExpression_{i,0}$, $FaultyExpression_{i,1}$, … , $FaultyExpression_{i,n-1}$ for the detected fault occurring at *i*th gate. The computed Boolean expression of the *j*th line in the faulty circuit may not be identical to the one generated at the *j*th line of the corresponding fault-free circuit.

Boolean difference method [21] is basically used to determine the complete test set for detecting stuck-at faults. We have employed the same it in reversible circuit for identifying SMGF faults.

Definition 9 The Boolean difference $\left(\frac{dF_j}{dG_i}\right)$ estimated at the *j*th line for the gate missing from the *i*th level can be expressed as $\frac{dF_j}{dG_i} = F_j^o \oplus F_j^i$, where F_j^o is the $\mathrm{OriginalExpression}_j$, F_j^i is the $\mathrm{FaultyExpression}_{i,j}$, $(0 \le i \le N-1)$ and $(0 \le j \le n-1)$.

Definition 10 The Boolean difference resulted due to removal of gate located at *i*th level is represented as $\frac{dF}{dG_i}$ which can be determined as follows:

$$\frac{dF}{dG_i} = \sum \frac{dF_j}{dG_i} = \left(\frac{dF_0}{dG_i}\right) + \left(\frac{dF_1}{dG_i}\right) + \cdots + \left(\frac{dF_{n-1}}{dG_i}\right), (0 \le i \le N-1), (0 \le j \le n-1)$$

In this way, at first the Boolean difference for the missing of each gate in the circuit is computed and then the Boolean difference of the circuit is constructed using the Boolean difference for each gate of the given circuit.

Definition 11 The Boolean generator can be defined as the Boolean expression needed for the test set construction so as to identify all the possible SMGFs in the circuit.

Lemma 1 *For a reversible circuit of n number input lines and N number of gates, then computation of individual Boolean difference* $\left(\frac{dF}{dG_i}\right)$ *enables to identify SMGF at the* *i*th *level.*

Proof For the reversible circuit with *n* lines and *N* gates, the Boolean difference $\left(\frac{dF}{dG_i}\right)$ for detecting the gate missing at the *i*th level is computed as:

$$\frac{dF}{dG_i} = \sum \frac{dF_j}{dG_i}, (0 \le i \le N-1), (0 \le j \le n-1) \tag{1}$$

Σ denotes the OR operation and

$$\frac{dF_j}{dG_i} = F_j^o \oplus F_j^i, (0 \le i \le N-1), (0 \le j \le n-1) \tag{2}$$

From the expression given at Eq. (1), it can be noticed that the possible values for $\frac{dF}{dG_i}$ can be either 0 or some other Boolean representation. Furthermore, it can be determined that the estimated result of $\frac{dF}{dG_i}$ becomes zero provided each of the terms $\frac{dF_j}{dG_i}$ evaluates to "0" as logical OR is being performed between any two successive terms of $\frac{dF}{dG_i}$.

By analyzing the expression given in Eq. (2), it can be observed that the term $\frac{dF_j}{dG_i}$ returns the value "0", if both F_j^o and F_j^i are becomes identical only if the circuit is fault free. This in turns suggests that, if the expression $\frac{dF}{dG_i}$ returns 0 then the circuit is said to be fault free.

For any SMGF in the circuit, it is obvious that the output expression obtained at any of the n input lines varies with the one derived for fault-free circuit. It means that at least a single line say j must be present for which the terms F_j^o and F_j^i turns out to be different due to existence of fault. It implies that $\frac{dF}{dG_i}$ cannot be equal to 0 in the faulty reversible circuit as logical OR is implemented between the consecutive terms $\frac{dF}{dG_i}$ and $\frac{dF_j}{dG_i}$.

4.1.2 Implementation of Boolean Generator for Detection of Single Missing Gate Fault

In the proposed method, the Boolean generator of the given circuit is estimated using the function $\mathrm{B_{Gen}} = \frac{\mathrm{dF}}{\mathrm{dG_0}} \wedge \frac{\mathrm{dF}}{\mathrm{dG_1}} \wedge \ldots \wedge \frac{\mathrm{dF}}{\mathrm{dG_{N-1}}}$, where $\wedge$ represents the logical AND operation and N represents number of gates in the circuit. For BGen $\neq 0$, the BGen is minimized to construct the Boolean generator of the circuit. For BGen $= 0$, the expressions $\mathrm{B^1_{Gen}}$ and $\mathrm{B^2_{Gen}}$ need to be computed to determine the Boolean generator of the given circuit. The computation method of the expressions $\mathrm{B^1_{Gen}}$ and $\mathrm{B^2_{Gen}}$ are as follows:

Let $\mathrm{S = (so, s1, \ldots , sN - 1)}$ be the set of all $\frac{\mathrm{dF}}{\mathrm{dG_i}}$, where $\mathrm{si} = \frac{\mathrm{dF}}{\mathrm{dG_i}}$ for $(0 \leq \mathrm{i} \leq \mathrm{N} - 1)$. Let $\mathrm{S1} \subseteq \mathrm{S}$ contains maximum number of si's such that $\mathrm{B^1_{Gen}} = \mathrm{Si1 \wedge Si2 \wedge \cdots \wedge Sik} \neq 0$, where $\mathrm{Sik} \in \mathrm{S1}$. $\mathrm{B^2_{Gen}}$ are remaining si's of S. For each of the $\mathrm{B^2_{Gen}}$, the $\mathrm{B^1_{Gen}} = 0$.

After the formation $\mathrm{B^1_{Gen}}$ and $\mathrm{B^2_{Gen}}$, the expression $\mathrm{B^1_{Gen}}$ is upgraded to compute the final form of the Boolean generator as follows:

(i) If any term of B^2_{Gen} completely matches or is subset of any term of B^1_{Gen}, no need to upgrade the B^1_{Gen}, else compare different terms of the B^1_{Gen} with each term of the B^2_{Gen} and upgrade the B^1_{Gen} as: $B^1_{Gen} = B^1_{Gen}+$ highest matching term [i], i = 0 to (number of highest matching term – 1). Repeat the same procedure between upgraded B^1_{Gen} and other available B^2_{Gen}, if exist.
(ii) The minimized form of B^1_{Gen} is the Boolean generator of the circuit.

4.1.3 Test Vector Construction from the Boolean Generator of the Circuit

The minterms and their corresponding decimal values for the generator are calculated and the collection of those decimal values is the test set of the circuit that will be used for the detection of SMGF.

Example 1 The *ham3tc* benchmark circuit of Fig. 4a is considered here. The circuits consisting of three lines ($n = 3$) and five gates ($N = 5$).

As per the first phase of the proposed technique, output expressions *OriginalExpression$_0$*, *OriginalExpression$_1$* and *OriginalExpression$_2$* are computed from the fault-free circuit of Fig. 4a.

The output expression generated for each circuit line is *Origi*nalExpre*ssion*$_0$ $= (a \oplus b \cdot c)$, *OriginalExpression*$_1$ $= ((b \oplus c) \oplus ((c \oplus (b \oplus c)) \oplus (a \oplus b \cdot c)))$, OriginalExpression$_2$ $= ((c \oplus (b \oplus c)) \oplus (a \oplus b \cdot c))$.

Now, let us assume that the gate *G$_0$* at the *level$_0$* is removed from the circuit. Hence, the circuit *ham*3*tc* becomes a faulty circuit (as shown in Fig. 4b) and its output expressions are FaultyExpression0,0, FaultyExpression0,1, FaultyExpression0,2.

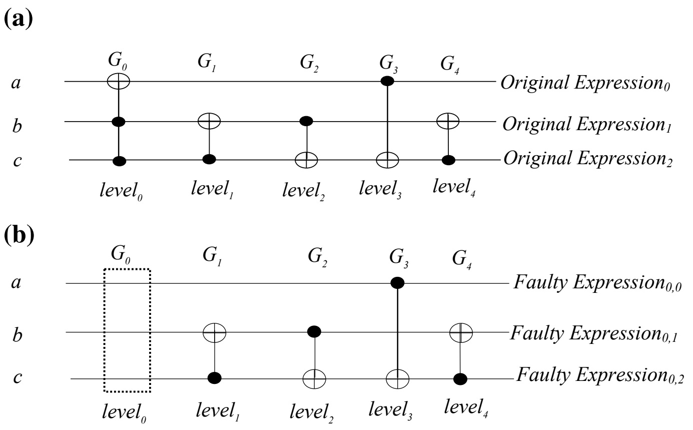


Fig. 4 **a** Fault free *ham3tc* circuit, **b** testable circuit for *SMGF* (Faulty *ham3tc* circuit where dotted box indicates missing of that gate)

Each expression of the faulty circuit are FaultyExpression$_{0,0}$ = (a), *FaultyExpression$_{0,1}$* = ((b ⊕ c) ⊕ ((c ⊕ (b ⊕ c) ⊕ (a))), FaultyExpression0$_{,2}$ = ((c ⊕ (b ⊕ c)) ⊕ (a)).

After finding out fault free expression of each line and the faulty expression of each line of the circuit, the Eqs. 1 and 2 as discussed earlier are used to compute the Boolean difference of the circuit. The Boolean difference of the circuit for the missing of the gate G0 is as $\frac{\mathrm{dF}}{\mathrm{dG_0}} = \left(\frac{\mathrm{dF_0}}{\mathrm{dG_0}}\right) + \left(\frac{\mathrm{dF_1}}{\mathrm{dG_0}}\right) + \left(\frac{\mathrm{dF_2}}{\mathrm{dG_0}}\right)$, $\frac{dF}{dG_0}$ = (OriginalIExpression0 ⊕ FaultyExpression0,0) + (OriginalExpression1 ⊕ FaultyExpression$_{0,1}$) + (OriginalExpression2 ⊕ FaultyExpression0,2) = ((a⊕b·c)⊕a) + (((b⊕c)⊕((c⊕(b⊕c)) ⊕(a ⊕ b · c))) ⊕ ((b ⊕ c) ⊕ ((c ⊕ (b ⊕ c) ⊕ (a))))) +(((c ⊕ (b ⊕ c)) ⊕ (a ⊕ b · c)) ⊕((c ⊕ (b ⊕ c)) ⊕ (a))) = bc

Similarly, each gate is removed at a time and the Boolean difference of the given circuit for the remaining gates is computed.

So, $\frac{dF}{dG_1} = c$, $\frac{dF}{dG_2} = b\bar{c} + \bar{b}c$, $\frac{dF}{dG_3} = \bar{a}bc + a\bar{b} + a\bar{c}$, $\frac{dF}{dG_4} = \bar{a}b\bar{c} + a\bar{b} + ac$.

Now as per the second phase of the proposed testing method, the Boolean generator of the circuit is computed by the statement B_{Gen}, where $B_{Gen} = (\frac{dF}{dG_0})AND(\frac{dF}{dG_1})AND(\frac{dF}{dG_2})\ AND(\frac{dF}{dG_3})AND(\frac{dF}{dG_4}) = (bc)AND(c)\ AND(b\bar{c} + \bar{b}c)AND(\bar{a}bc + a\bar{b} + a\bar{c})\ AND(\bar{a}b\bar{c} + ab + ac) = 0$. As B_{Gen} is 0, we have to find out the B^1_{Gen} and B^2_{Gen} to calculate the final form of the Boolean generator.

For this example, as gate count is $N = 5$, we need 5 iterations to generate the resultant Boolean generator of the circuit.

First Iteration: Let us assume $B^1_{Gen} = \frac{dF}{dG_0} = bc$.

Second Iteration: Here B^1_{Gen} is upgraded as follows: $B^1_{Gen} = B^1_{Gen}AND(\frac{dF}{dG_1}) = (bc)AND(c) = bc$.

Third Iteration: Similarly, $B^1_{Gen} = B^1_{Gen} AND(\frac{dF}{dG_2}) = (bc)AND(b\bar{c} + \bar{b}c) = 0$. That means the term $(b\bar{c}+\bar{b}c)$ converts the term B^1_{Gen} to 0 and therefore, the $B^2_{Gen}(0)$ is computed and B^1_{Gen} will not be upgraded. Now, $B^2_{Gen}(0) = (b\bar{c} + \bar{b}c)$.

Fourth Iteration: $B^1_{Gen} = B^1_{Gen} AND(\frac{dF}{dG_3}) = (bc)AND(\bar{a}bc + a\bar{b} + a\bar{c}) = \bar{a}bc$.

Fifth Iteration: $B^1_{Gen} = B^1_{Gen} AND(\frac{dF}{dG_4}) = (\bar{a}bc)AND(\bar{a}b\bar{c} + a\bar{b} + ac) = 0$. Once again as per the third iteration, the B^2_{Gen} (1) is computed and the function B^1_{Gen} will not be modified. Now, $B^2_{Gen}(1) = (\bar{a}b\bar{c} + a\bar{b} + ac)$.

Now, we need to compare B^1_{Gen} with B^2_{Gen} (0) and B^2_{Gen} (1) once again to upgrade the B^1_{Gen}. At First, the expression $B^1_{Gen}(\bar{a}bc)$ is compared with both the terms of B^2_{Gen} (0). The term $(\bar{a}bc)$ of B^1_{Gen} contains a single literal matching with both the terms of B^2_{Gen} (0) and hence, $B^1_{Gen}(0) = (\bar{a}bc + b\bar{c})$ and $B^1_{Gen}(1) = (\bar{a}bc + \bar{b}c)$.

Now, we need to compare both the B^1_{Gen} with the $B^2_{Gen}(1)$ to determine the updated value of B^1_{Gen} (0) and B^1_{Gen} (1). The B^1_{Gen} (0) is compared with the B^2_{Gen} (1) and the term $(\bar{a}bc)$ of B^1_{Gen} (0) and the term $(\bar{a}b\bar{c})$ of B^2_{Gen} (*1*) has the highest literal matching. So, the B^1_{Gen} (0) is upgraded to $B^1_{Gen}(0) = (\bar{a}bc + b\bar{c} + \bar{a}b\bar{c})$. Similarly, the B^1_{Gen} (1) is upgraded to $B^1_{Gen}(1) = (\bar{a}bc + \bar{b}c + \bar{a}b\bar{c})$.

After the minimization, the $B^1_{Gen}(0) = (\bar{a}b + b\bar{c})$ and $B^1_{Gen}(1) = (\bar{a}b + \bar{b}c)$.

As both B^1_{Gen} (0) and B^1_{Gen} (1) contains similar number of literals, there will be only two generators to identify all the possible *SMGF* in the given circuit. generator(0) = $B^1_{Gen}(0) = (\bar{a}b + b\bar{c})$ and generator(1) = $B^1_{Gen}(1) = (\bar{a}b + \bar{b}c)$

Now as per the third phase, the test vector formation from the Boolean generator is explained as follows:

$Minterm_{generator(0)} = \bar{a}b(c + \bar{c}) + b\bar{c}(a + \bar{a}) = \bar{a}bc + \bar{a}b\bar{c} + ab\bar{c} + \bar{a}b\bar{c} = \bar{a}bc + \bar{a}b\bar{c} + ab\bar{c} = \{3, 2, 6\} = \{2, 3, 6\}$. The test set derived from the generator(0) is $\{2, 3, 6\}$. Similarly, $Minterm_{generator(1)} = \bar{a}b(c + \bar{c}) + \bar{b}c(a + \bar{a}) = \bar{a}bc + \bar{a}b\bar{c} + a\bar{b}c + \bar{a}\bar{b}c = \{3, 2, 5, 1\} = \{3, 2, 5, 1\}$.

The test set derived from the generator(1) is {1, 2, 3, 5}. As the generator(0) contains lesser number of test vectors, the generator(0) will be used to uncover the *SMGF* fault.

4.2 Boolean Based Testing Method2

In this method, the ESOP-based reversible circuit is considered for the detection of SMGF fault and also to diagnosis the detected fault. Let us assume that C_{test} is the testable circuit in which the test has to be performed. To test the C_{test} circuit, initially a fault-free ESOP-based circuit (C_{true}) is read from a given specification files (T_{spec}) and then the logical XOR is performed between C_{true} and C_{test}.

But the design of fault-free ESOP circuit from the T_{spec} creates problem because number of distinct ESOP circuits can be generated from the T_{spec} and due to this reason MPRM can be considered as the subclass of an ESOP expression. A fault-free ESOP circuit (C_{true}) can be identified from a number of distinct ESOP circuits by the help of some complex calculation. To solve the said problem, we have used the PPRM class from which only one circuit can be designed.

The proposed testing method is segmented into two stages. At first, fault detection is performed in the C_{test} and after that in second stage, identification of the detected fault in the C_{test} is performed and proper diagnosis is done in the C_{test} to transform the circuit from faulty to fault-free circuit.

4.2.1 Detection of SMGF

Here in this phase, a PPRM expression f_{true}^{PPRM} from the given circuit specification file$\mathrm{T_{spec}}$ is obtained for the circuit, $\mathrm{C_{test}}$. After that the MPRM expression f_{test}^{MPRM} is derived from testable ESOP circuit $\mathrm{C_{test}}$. Now, for each variable contained within MPRM expression ($\mathrm{f_{test}^{MPRM}}$), the polarity of such variables is converted to positive form and thereby a FPRM form (f_{test}^{FPRM}) can be derived. Now, the $\mathrm{L_{XOR}}$ is computed as follows: $\mathrm{L_{XOR}} = f_{true}^{PPRM} \oplus f_{test}^{FPRM}$, where $\oplus$ denotes the XOR operation. If $\mathrm{L_{XOR}}$ is zero that means fault is not detected in the $\mathrm{C_{test}}$ else SMGF is detected in the $\mathrm{C_{test}}$.

4.2.2 Detection and Correction of SMGF

Here, both fault identification and correction of the specified fault in a given circuit. The output expression obtained from L_{XOR} is considered as the specification details of the corresponding missing gate and represented in the form of a Boolean expression and the variables contained within such expression L_{XOR} designates the control inputs for the gate Mg, where Mg denotes the missing gate in C_{test} circuit.

To make the circuit function correctly, if the identified missing gate (M_g) is attached to the input circuit C_{test} to convert it to a fault-free circuit.

Now, the proposed method of SMGF identification and medication in an ESOP based circuit has been discussed with supportive examples below.

Example 2 The benchmark ESOP circuit 4gt4 [22] is used here for testing the proposed methodology. The $\mathrm{T_{spec}}$ of 4gt4 is represented in Fig. 5a. The testable circuit, $\mathrm{C_{test}}$ for the given circuit 4gt4 is also shown in Fig. 5c. Now, the PPRM cover of the fault-free circuit is obtained from file $\mathrm{T_{spec}}$ employing [23]. In Fig. 5b, the corresponding PPRM cube can be observed and the derived PPRM expression is $f_{true}^{PPRM} = \mathrm{a} \oplus \mathrm{bd} \oplus \mathrm{bc} \oplus \mathrm{bcd} \oplus \mathrm{abd} \oplus \mathrm{abc} \oplus \mathrm{abcd}$. Now, the obtained expression from the circuit $\mathrm{C_{test}}$ is $f_{test}^{MPRM} = \mathrm{a} \oplus \bar{a}\mathrm{b}\bar{c}\bar{d} \oplus \bar{a}\mathrm{b}$ and changing the polarity of each literal in the expression f_{test}^{MPRM} to positive value is required to derive the corresponding FPRM expression (($f_{test}^{FPRM} = \mathrm{a} \oplus \mathrm{bd} \oplus \mathrm{bc} \oplus \mathrm{bcd} \oplus \mathrm{abd} \oplus \mathrm{abc} \oplus \mathrm{abcd}$). Now $\mathrm{L_{XOR}}$ is computed as $\mathrm{L_{XOR}} = f_{true}^{PPRM} \oplus f_{test}^{FPRM} = \emptyset$.

Hence, it can be confirmed that there an SMGF does not exist in the circuit C_{test} as L_{XOR} is null.

(a)

```
# Function: 4gt4
.input a b c d
.output f
a b c d f
0101 1
 0110 1
0111 1
 1000 1
1001 1
1010 1
1011 1
1100 1
1101 1
1110 1
11111
```

(b)

a	b	c	d
–	1	–	1
–	1	1	–
–	1	1	1
1	–	–	–
1	1	–	1
1	1	1	–
1	1	1	1

(c)

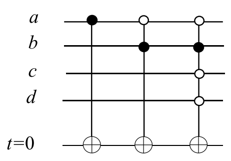


Fig. 5 Illustration of Example 2. **a** Input specification file of *4gt4* **b** Equivalent PPRM cube list of *4gt4* **c** Testable input ESOP circuit for function *4gt4*

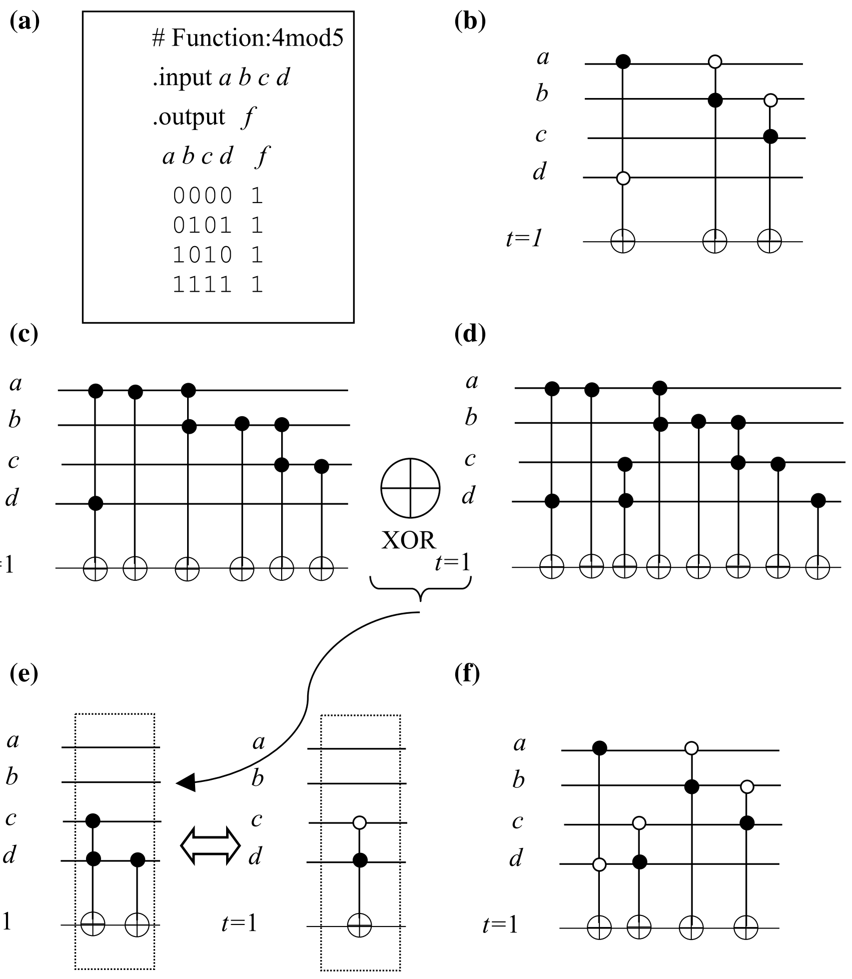
Example 3 The benchmark circuit 4mod5 [22] is considered here once again to explain the proposed technique. The specification file T_{spec} for 4mod5 and C_{test} circuit are represented in Fig. 6a and b, respectively. Initially, $f_{true}^{PPRM} = 1 \oplus ad \oplus ab \oplus bc \oplus cd \oplus a \oplus b \oplus c \oplus d$ is obtained from T_{spec}. Then, MPRM expression ($f_{test}^{MPRM} = 1 \oplus a\bar{d} \oplus \bar{a}b \oplus \bar{b}c$) is derived from the circuit C_{test}. Now, an FPRM logic expression ($f_{test}^{FPRM} = 1 \oplus ad \oplus ab \oplus bc \oplus a \oplus b \oplus c$) is formed. The equivalent ESOP structure of derived f_{test}^{FPRM} and f_{true}^{PPRM} expressions are represented in Fig. 6c and d, respectively. Now, $L_{XOR} = f_{true}^{PPRM} \oplus f_{test}^{FPRM} = cd \oplus d = \bar{c}d$.

Hence, it is confirmed that a SMGF fault is present in C_{test} exists as L_{XOR} is equal to some Boolean value. As per the proposed method, the expression L_{XOR} represents the control lines of the gate M_g (as depicted in Fig. 6e) which is completely missing in the testable circuit. Thereafter, it can be described that the M_g is having a negative control and positive control at lines *c* and *d,* respectively.

The circuit can be made fault free if the corresponding missing gate (M_g) is adjoined to the input circuit C_{test} and the resultant original ESOP structure of *4mod5* is depicted in Fig. 6f.

5 Experimental Results

We have tested both of our approaches against different benchmark suites [22]. The results obtained from first testing approach is summarized in Table 1, where first three columns represent the circuit name, the number of lines (n) and the number of gates (N) present in a benchmark function. The Boolean generator is shown in



```
# Function:4mod5
.input a b c d
.output f
a b c d  f
0000 1
0101 1
1010 1
1111 1
```

Fig. 6 Illustration of Example 3. **a** Specification file T_{spec} for circuit *4mod5* **b** Testable input circuit *4mod5* **c** f_{test}^{FPRM} function equivalent ESOP circuit **d** f_{test}^{FPRM} function equivalent ESOP circuit **e** Detected missing gate details **f** Fault free circuit of *4mod5*

column 4 and the set of test patterns produced from the generator are tabulated in column 5 of Table 1.

The second testing technique is also checked over several benchmark circuits and the effectiveness of RM form also has been verified for successful detection and localization of faults.

Table 1 Boolean generator for the detection of SMGF

Name of benchmark function	Number of lines (n)	Number of gates (N)	Boolean generator of the circuit	Derived test set from the boolean generator

rd32d1	4	4	$(ab + \bar{a}bc)$
			or
			$(ab + a\bar{b}c)$
				{6, 7, 12, 13, 14, 15}
				or
				{10, 11, 12, 13, 14, 15}

xor5d1	5	4	$(a\bar{b}\bar{c}\bar{d})$	{16, 17}

ham3tc	3	5	$(\bar{a}b + b\bar{c})$
			or
			$(\bar{a}b + \bar{b}c)$
				{2, 3, 6}
				or
				{1, 2, 3, 5}

3_17tc	3	6	$(a\bar{b}c + ab\bar{c})$
			or
			$(a\bar{b}c + \bar{a}\bar{b}\bar{c})$
				{5, 6}
				or
				{5, 0}

mod5d1	5	8	$(abcd)$	{30, 31}
mod5d2	5	9	$(\bar{a}\bar{b}cd)$	{6, 7}
4_49d3	4	12	$(abd + \bar{a}cd)$	{3, 7, 13, 15}
hwb4d1	4	17	$(a\bar{b}cd + \bar{c}d + ac\bar{d})$	{1, 5, 9, 10, 11, 13, 14}
2of5d1	6	15	$(\bar{a}c\bar{e} + a\overline{bd}e + \bar{a}bd\bar{e} + \bar{a}b\bar{c}df)$	{8, 9, 12, 13, 20, 21, 23, 24, 25, 28, 29, 34, 35, 42}
mod5adders1	6	21	$(abc\bar{e}\bar{f} + ab\bar{c}\bar{d}\bar{e} + a\bar{d}\bar{e}\bar{f} + \bar{b}\bar{c}\bar{e}\bar{f})$	{0, 4, 32, 36, 40, 48, 49, 56, 60}

6 Conclusions

This work has presented two Boolean based approaches for testing of SMGF faults in reversible circuit. In the first method, a Boolean generator has developed for a reversible circuit and in later time, from this generator test vectors are constructed to a test a circuit. The second testing technique has mainly targeted to test a special class of reversible circuit known as ESOP designs. But, the first testing technique is very generic and can be employed over any type of circuits. In both the testing approaches, we have addressed SMGF only, but other types of faults like RGF, MMGF also can be tracked by following the same strategy as used to find SMGF. The presented techniques have successfully tested over a wide spectrum of benchmarks also.

References

1. Landauer R (1961) Irreversibility and Heat Generation in the Computing Process. IBM J Res Dev 5:183–191
2. Bennett CH (1973) Logical Reversibility of Computation. IBM J Res Dev 17:525–532

3. Nielsen M, Chuang I (2000) Quantum Computation and Quantum Information. Cambridge University Press, Cambridge
4. Wille R, Keszocze O, Hillmich S, Walter M, Ortiz AG (2016) Synthesis of approximate coders for on-chip interconnects using reversible logic. In: Design, automation and test in Europe
5. Rauchenecker A, Ostermann T, Wille R (2017) Exploiting reversible logic design for implementing adiabatic circuits. In: International conference mixed design of integrated circuits and systems
6. Ramasamy K, Tagare R, Perkins E, Perkowski M (2004) Fault localization in reversible circuits is easier than for classical circuits. In: Proceedings of international workshop on logic and synthesis
7. Chakraborty A (2005) Synthesis of reversible circuits for testing with universal test set and C-testability of reversible iterative logic arrays. In: Proceedings of VLSI design, pp 249–254
8. Rahaman H, Kole DK, Das DK, Bhattacharya BB (2011) Fault diagnosis for missing-gate fault (SMGF) model in reversible quantum circuits. Int J Comput Electr Eng (Elsevier) 37:475–485
9. Mondal J, Das DK, Kole DK, Rahaman H, Bhattacharya BB (2013) On designing testable reversible circuits using gate duplication. In: International symposium on VLSI design and test, pp 322–329
10. Kole DK, Rahaman H, Das DK, Bhattacharya BB (2010) Derivation of optimal test set for detection of multiple missing-gate faults in reversible circuits. In: Proceedings of Asian test symposium, pp 33–38
11. Mahammad SN, Hari SKS, Shroff S, Kamakoti V (2006) Constructing online testable circuits using reversible logic. In: International symposium on VLSI design and test, pp 373–383
12. Toffoli T (1980) Reversible computing. Technical memo MIT/LCS/TM-151, MIT Lab for Computer Science
13. Fredkin E, Toffoli T (1982) Conservative logic. Int J Theor Phys 21(3–4):219–253
14. Feynman RP (1996) Feynman lectures on computation. Perseus books
15. Fazel K, Thornton MA, Rice JE (2007) ESOP-based Toffoli gate cascade generation. In: Pacific rim conference on communications, computers and signal processing (PacRim). Victoria, Canada, pp 206–209
16. Zhang YZ, Rayner PJW (1984) Minimization of Reed-Muller polynomials with fixed polarity. IEE Proc Comput Digit Tech 131(5):177–186
17. Harking B (1990) Efficient algorithm for canonical Reed-Muller expansion of Boolean function. IEE Proc Comput Digit Tech 137(5):366–377
18. Hayes JP, Polian I, Becker B (2004) Testing for missing-gate faults inreversible circuits. In: Proceedings of Asian test symposium, pp 100–105
19. Perkowski M, Biamonte J, Lukac M (2005) Test generation and fault localization for quantum circuits. In: Proceedings of international symposium on multi-valued logic, pp 62–68
20. Polian I, Hayes JP, Fienn T, Becker B (2005) A family of logical fault models for reversible circuits. In: Proceedings of Asian test symposium, pp 422–427
21. Kohavi Z (1978) Switching and finite automata theory, 2nd edn. Tata McGraw- Hill, New York
22. Wille R, Grosse D, Teuber L, Dueck GW, Drechsler R (2008) Revlib: an online resources for reversible functions and reversible circuits. IEEE ISMVL 24:220–225
23. Bandyopadhyay C, Rahaman H (2014) Synthesis of ESOP-based reversible logic using positive polarity reed-muller form. In: IEEE emerging trends in computing and communication (ETCC-2014), Calcutta, India