J Electron Test (2016) 32:175–196
DOI 10.1007/s10836-016-5574-4

An Exact approach for Complete Test Set Generation of Toffoli-Fredkin-Peres based Reversible Circuits

A. N. Nagamani[1] · S. Ashwin[1] · B. Abhishek[1] · V. K. Agrawal[2]

Received: 11 August 2015 / Accepted: 22 February 2016 / Published online: 16 March 2016
© Springer Science+Business Media New York 2016

Abstract Reversible logic has gained interest of researchers worldwide for its ultra-low power and high speed computing abilities in the future quantum information processing. Testing of these circuits is important for ensuring high reliability of their operation. In this work, we propose an ATPG algorithm for reversible circuits using an exact approach to generate CTS (Complete Test Set) which can detect single stuck-at faults, multiple stuck-at faults, repeated gate fault, partial and complete missing gate faults which are very useful logical fault models for reversible logic to model any physical defect. Proposed algorithm can be used to test a reversible circuit designed with k-CNOT, Peres and Fredkin gates. Through extensive experiments, we have validated our proposed algorithm for several benchmark circuits and other circuits with family of reversible gates. This algorithm produces a minimal and complete test set while reducing test generation time as compared to existing state-of-the-art algorithms. A testing tool is developed satisfying the purpose of generating all possible CTS's indicating the simulation time, number of levels and gates in the circuit. This paper also contributes to the detection and removal of redundant faults for optimal test set generation.

Keywords Exact algorithms · Testing · Redundant faults · Stuck-at faults · Missing gate faults

Responsible Editor: B. B. Bhattacharya

✉ A. N. Nagamani
nagamani@pes.edu

S. Ashwin
ashwinsmnth@gmail.com

B. Abhishek
abhishek.b147@gmail.com

V. K. Agrawal
vk.agrawal@pes.edu

[1] Department of ECE, PES University Campus, PES Institute of Technology, Bangalore, Karnataka, India

[2] Department of Information science and Engineering, PES University Campus, PES Institute of Technology, Bangalore, Karnataka, India

1 Introduction

According to Rolf Landauer [12], logical irreversibility is associated with physical irreversibility which serves the purpose of standardizing signals by minimal heat generation, per machine cycle. The standardized signals are independent of their exact logical history and the device is said to be logically irreversible if the output of a device does not uniquely define the inputs. For every bit of information lost in the process, there is an increase in entropy by the factor of $kTln2$ Joules [12] where k is the Boltzmann constant (approximately $1.38 \times (10^{-23})$ J/K), T is the temperature of the circuit in Kelvins, and $ln2$ is the natural logarithm of 2 (approximately 0.69315). Charles H. Bennett argued that for zero power dissipation the computation has to be reversible, but if a computation is reversible it is not zero power [1]. Basically the energy dissipation for each information bit lost

in a reversible circuit is less than the energy limit proposed by Landauer i.e., $E < kTln2$. Hence reversible logic has gained importance with its low power application.

Reversible circuits used with newer technologies such as quantum computing [21], optical computing [28], Quantum Cellular Automata (QCA) [14], trapped-ion technology [23], adiabatic CMOS [18] offers reduction in power dissipation. Testing is one of the most important procedures in accomplishing a chip. The product quality confides in the defect level. A defect in an electronic system is the inadvertent difference between the implemented hardware and its intended design. An embodiment of a defect at the abstract function level is a fault. Detection of these faults before releasing the chip into the market is very essential. Otherwise it would be a huge setback for the manufacturing company. The benefits of testing are quality and economy [22].

Fault detection and fault diagnosis are the two important phases of testing. The role of the former is to detect whether the circuit/chip is functioning properly or not; while the latter determines exactly what went wrong and where the process needs to be altered. Both these phases have their own complexities and the latter depends on the former. Hence fault detection is the basis for fault diagnosis. There are several faults which may occur during fabrication or manufacturing of chips. These physical defects are mapped to fault models such as stuck-at faults, missing gate faults, bridging faults, cross point faults and many more [23, 24]. The detection of these faults in reversible logic circuits is relatively easy compared to irreversible counterpart. Several works have been done in this area with certain assumptions or constraints. The authors in [24] have proposed an ATPG algorithm which detects only stuck-at faults in a circuit comprising of only k-CNOT gates with few gate overheads due to incorporation of DFT technique to detect faults. The work in [7] proposes an ATPG algorithm for detecting Single Missing Gate Fault (SMGF) for k-CNOT circuits with an assumption that at most one gate can be faulty at a time and are detectable only at circuit's primary outputs. As an extension to the work in [7], the same authors in [23] proposed an ATPG for Multiple Missing Gate Fault (MMGF) and Partial Missing Gate Fault (PMGF) considering only k-CNOT gate. The authors in [26] optimized the test set generation for the combination of SMGF and PMGF whereas the optimized test set generation for MMGF is presented in [11] and have also proved that test set for MMGF is sufficient to test SMGF and Repeated Gate Fault (RGF).

Generating complete test set for a combinational circuit with minimal test set is a NP-hard problem and the solution can be obtained using an exhaustive search technique [33]. Exact algorithms aim at computing optimal solution, with the algorithm passing through the same sequence of operations. These exact algorithms (deterministic algorithms) are expensive in terms of run time or memory and hence not suitable for very large input size. On the other hand, the solution can be obtained for this problem with various heuristic approaches. Heuristic approaches use optimization techniques for solving and they may not give minimal solution. In this work we have used an exact approach to determine CTS with minimal test set for reversible combinational circuits. Although the complexity of the problem may become exponential in some rare cases, an optimal solution is found. The detailed analysis of algorithms are presented in next sections.

In this work, we have proposed an ATPG algorithm for reversible circuits using exact approach to generate CTS which can detect single stuck-at faults, multiple stuck-at faults, repeated gate fault, partial and complete missing gate faults either exclusively or combining different fault models. This is the first work in the literature to generate CTS combining both stuck-at fault models and missing gate fault models considering all kinds of standard reversible gates such as *k*-CNOT, Peres and Fredkin gates (we will be referring these gates as the family of reversible gates). The synthesis algorithms for reversible circuits using Toffoli-Fredkin and k-CNOT-Peres-Fredkin gates are proposed in [5, 17, 27] for which test algorithms are not proposed in literature. This is one of the main motivations for this work. A testing tool is developed to generate CTS with minimal test set for reversible circuits which take inputs either in the form of .tfc, .real, .jpg, .bmp, .png or .fig. The output panel displays the minimal test set for that particular circuit and also the simulation time of the CPU, number of gates, and number of lines in a circuit. The tool also generates all possible CTS's for a given circuit with an additional feature of redundancy detection and removal. All the proposed algorithms are validated on the benchmark circuits from the source available in [16].

The main contributions of this work are as follows:

1. An exact ATPG algorithm to generate CTS with minimal test set for a given reversible circuit.
2. Algorithm to test reversible circuits designed with the family of reversible gates.

The rest of the paper is organized as follows. A general discussion on reversible logic, basic reversible gates, fault models and prior work on fault detection in reversible logic is given in Section 2. In Section 3, we present the ATPG algorithms with detailed illustrations and complexity

analysis to detect stuck-at fault, complete missing gate fault and partial missing gate fault in reversible circuits. Performance evaluation and the experimental results are reported in Section 4. Concluding remarks and future work are materialized in Section 5.

2 Preliminaries

2.1 Reversible Logic

Reversible logic is one of the forerunners of post-CMOS technology with ultra-low power applications. For a circuit to be called as reversible, it should satisfy certain conditions: there should be no fan-out and no feedback and also the function should be bijective. The No-fan-out theorem of reversible logic circuits has a correlation with No-cloning theorem of quantum circuits which states that an unknown quantum state cannot be copied [13]. On the other hand, the reversible circuit function should be bijective means that, it maps each input pattern to a unique output pattern. In general sense, for logic function $f : B^n \Rightarrow B^m$ the number of inputs should be equal to number of outputs, i.e., $n = m$.

2.2 Basic Reversible Gates

For a logic gate to be reversible, there must be equal number of inputs and outputs satisfying the bijective property otherwise some information in the input can be lost in the output and vice versa [22]. Some of the reversible gates are NOT gate, CNOT/FEYNMAN gate, TOFFOLI gate, FREDKIN gate and PERES gate. The circuits designed using these gates are called reversible circuits and are governed by performance parameters such as Gate Count (GC), Quantum Cost (QC), Ancilla Inputs (AI), Garbage Outputs (GO) and Delay (Δ).

The k-CNOT gate is a k-input, k-output reversible gate having transformation from $[I_1, I_2, ...I_{k-1}, I_k]$ to $[I_1, I_2, ..., I_{k-1}, (I_1 \cdot I_2 \cdot .... \cdot I_{k-1}) \oplus I_k]$. If $k = 2$, then it is called CNOT gate and if $k = 3$, it becomes TOFFOLI gate. TOFFOLI gate is the most popular reversible gate having a quantum cost of 5. FREDKIN gate is a $3X3$ reversible gate which basically operates as a conditional router or multiplexer [31]. Depending on the bit value in one line (Control Line), the outputs at other two lines (Target Lines) are either

Table 1 Truth table for 3_17tc reversible circuit with faulty and fault-free outputs

Inputs	Fault-free output	Faulty output				
		SMSF	SMGF	RGF	PMGF	MMGF
000	111	010	010	010	111	001
001	000	100	000	000	000	000
010	001	101	001	001	001	010
011	011	110	011	011	110	011
100	100	100	100	100	100	100
101	010	010	111	111	010	101
110	110	110	110	110	011	110
111	101	101	101	101	101	111

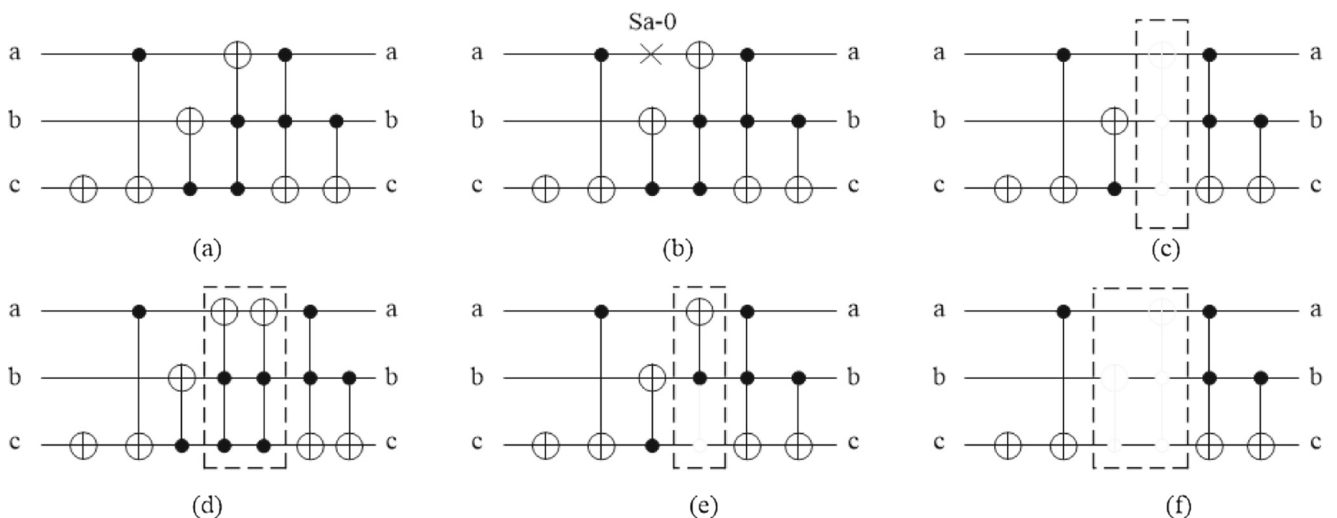


Fig. 1 Illustration of Reversible Circuit 3_17tc for various fault conditions

Table 2 Implementation table of equation (d) of SMSF algorithm

Row of SaFarray	All possible Stuck at Faults of 3_17tc circuit for test set [0 1 6]																	
0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	0	1	1	0
1	0	1	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1
6	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	0	1
$l_0 \vert l_1 \vert l_6$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

passed through unchanged or swapped with each other. The FREDKIN gate is also known as controlled swap(CSWAP) gate [22] having a quantum cost of 5. FREDKIN gate has a special property that it preserves parity. Hence FREDKIN is regarded as Fault Tolerant gate, in other words conservative gate. FREDKIN gate has transformation from $[I_1, I_2, I_3]$ to $[I_1, \bar{I_1} \cdot I_2 \oplus I_1 \cdot I_3, \bar{I_1} \cdot I_3 \oplus I_1 \cdot I_2]$. When $I_1 = 1$, it becomes SWAP gate. The PERES gate is also a $3X3$ reversible gate having transformation from $[I_1, I_2, I_3]$ to $[I_1, I_1 \oplus I_2, I_1 \cdot I_2 \oplus I_3]$. The quantum cost of PERES is 4.

Table 3 Fault coverage table for 3_17tc benchmark circuit with 2 test vectors

Test vector 1	Test vector 2	Faults covered	% Fault coverage
0	1	15	83.33333
0	2	13	72.22222
0	3	14	77.77778
0	4	13	72.22222
0	5	15	83.33333
0	6	14	77.77778
0	7	15	83.33333
1	2	15	83.33333
1	3	12	66.66667
1	4	13	72.22222
1	5	13	72.22222
1	6	16	88.88889
1	7	15	83.33333
2	3	14	77.77778
2	4	15	83.33333
2	5	15	83.33333
2	6	14	77.77778
2	7	13	72.22222
3	4	16	88.88889
3	5	14	77.77778
3	6	15	83.33333
3	7	14	77.77778
4	5	15	83.33333
4	6	12	66.66667
4	7	15	83.33333
5	6	14	77.77778
5	7	13	72.22222
6	7	14	77.77778

2.3 Fault Models

In reversible logic, in addition to traditional fault models, a few more fault models need to be considered for complete testing of these circuits and testing a circuit for these specific fault models will give high confidence over the designs. For a reversible circuit designed with k-CNOT based gates, several fault models were introduced in literature [23] and [24]. Single and Multiple Stuck-at Faults (SMSF) [23], Single Missing Gate Fault (SMGF) [23], Repeated Gate Fault (RGF) [24], Partial Missing Gate Fault (PMGF) [24] and Multiple Missing Gate Fault (MMGF) [24]. In this section, we discuss these faults in detail and their effects on functionality of reversible circuits. 3_17tc as shown in Fig. 1a is used as the reference in the following sections for explanation.

2.3.1 Single and Multiple Stuck-at Fault (SMSF)

Stuck-at faults are the traditional fault models. A single-bit error or the cell fault which makes a particular signal permanently stuck at a value and thus does not allow a signal transition. (Eg: When the circuit implemented using QCA, or adiabatic CMOS [18] and [4]). Figure 1b shows the effect of single stuck at fault in 3_17tc bench mark circuit. In Fig. 1b, a stuck-at 0 fault is present on line ‘a’ in level 3. In any test pattern generation methods for stuck-at faults, there are three necessary and sufficient steps [2]: *Fault Activation, Fault Propagation and Back Tracing*. On applying these steps, the test vector needed to test this fault is: 000 or 001 or 010 or 011. The truth table for the above circuit with and without Sa-0 fault is as depicted in Table 1.

2.3.2 Single Missing Gate Fault (SMGF)

In this fault model, any one k-CNOT gate completely disappears from the circuit. Technically CNOT gate behaves as a simple wire connection. This also has an impact on circuit functionality. The pulse implementing the gate operation may be short, missing, misaligned, or mistuned. Let us consider the 4th gate that is, TOFFOLI gate is missing from

the circuit as shown in Fig. 1c. If we apply {1 0 1} to the circuit, the output would be {0 1 0}, whereas in the presence of SMGF fault highlighted by a box, the output will be {1 1 1}. Hence the vector {1 0 1} detects this fault. The functional error in the circuit for all other inputs due to this fault is depicted in Table 1. The total number of SMGFs is equal to total number of gates in the circuit.

2.3.3 Repeated Gate Fault (RGF)

If a gate is repeated due to multiple instantiation of a particular gate, then RGF models are used to define the effects of this fault on functionality of reversible circuit. The physical justification for an RGF is the occurrence of long or duplicated pulses [23]. Figure 1d shows the RGF model in 3_17tc reversible circuit. Let {0 0 0} be the input applied to the circuit and the expected fault-free output is {1 1 1}, whereas with the presence of RGF the output evaluates to {0 1 0}. Hence {0 0 0} is one of the test vector to detect RGF in the circuit shown in Fig. 1a. Table 1 shows fault-free and faulty outputs for all other input combinations. From the truth table it is evident that the test patterns for SMGF and RGF are the same. Both these fault types have same characteristics in terms of test patters. Based on this observation, the following theorem holds when generalized:

Theorem 1 (RGF Fault Effect on Reversible Circuit) *Consider a Reversible Circuit RC having RGF fault effect which replaces a reversible gate by r instances of the same gate. Then the*

$$Test\ Set\ for\ RGF = \begin{cases} Test\ Set\ of\ SMGF; & if\ r\ is\ even \\ No\ Test\ Set\ Required; & if\ r\ is\ odd \end{cases}$$

Proof According to the truth table shown in Table 1 for the circuits depicted in Fig. 1a and Fig. 1d, the output values in the presence of RGF (even number of gate instances i.e., for r being even) differs from the actual value for two inputs {0 0 0} and {1 0 1}. For the same inputs, the output values in the presence of SMGF also differs from ideal value. Hence {0 0 0} and {1 0 1} can detect both RGF and SMGF. When r is odd, the fault is redundant since the fault effect gets cancelled and hence there is no need of any test set to detect this redundant fault. □

2.3.4 Partial Missing Gate Fault (PMGF)

This fault models the defects caused in a reversible gate if any control is missing due to physical alignment or tuning of the gate pulses. In case of k-CNOT gate, PMGF reduces the gate to p-CNOT gate, with $p < k$; whereas in case of FREDKIN gate, the presence of PMGF converts the gate functionality from FREDKIN gate to SWAP gate. Let us consider a case where control of the TOFFOLI gate is missing as shown in Fig. 1e. For the applied input {0 1 1}, the output will be {1 1 0} instead of {0 1 1} due to the presence of PMGF. Hence the vector {0 1 1} detects this fault. The corresponding fault-free and faulty outputs for all other inputs are depicted in Table 1. It has been shown in [23] that the test vector which detects first order PMGF is sufficient to detect higher order PMGF as well. This statement is exclusively for *k*-CNOT gates as higher order PMGF occurs only in *k*-CNOT gate.

2.3.5 Multiple Missing Gate Fault (MMGF)

If a physical defect causes multiple gates or a subset of reversible gates to disappear from the circuit, then those type of faults are modeled as MMGF. Even this requires a detailed analysis to detect the fault from the functional analysis of the circuit. According to the authors in [24], a fault is said to be MMGF, if and only if two or more consecutive gates are missing. This is justified by the assumption that the gate operations implemented using laser is more likely to be disturbed for a period of time exceeding one gate operations. Hence always consecutive gates are affected rather than distinct gates here and there. Consider a case where two consecutive gates are missing as shown in Fig. 1f. Let the input applied be {0 1 0}. The fault-free output is {0 0 1} whereas, the faulty output is {0 1 0} which differs from expected output. Hence the vector {0 1 0} detects MMGF. The functional errors for other inputs are described in Table 1.

2.4 Prior Work on Fault Detection in Reversible Logic

2.4.1 Single and Multiple Stuck-at Faults

These are the classical fault models defined for reversible and quantum circuits. Testing can be performed either online or offline or with DFT. Online testing implies that the testing can be performed during the normal mode of operation of the circuit with hardware overhead. But offline testing needs a dedicated test mode without any test hardware overhead, whereas DFT refers to Testing with definite test input and with minimal additional test hardware [2]. The authors in [24] have proposed an ATPG with DFT for detecting all single stuck-at faults in a *k*-CNOT reversible circuit. But the methodology proposed by them does not consider any general n-bit reversible circuit. In [9], an ATPG with DFT methodology is proposed to detect all stuck-at faults. Here stuck-at faults are tested using minimal vectors with the gate replacement technique which increases the quantum cost of a testable circuit.

Algorithm 1 CTS for single and multiple stuck-at fault model

Input: Reversible Circuit RC with 'n' lines and 'N' gates
Output: The minimal CTS and test vectors

Stage I: Input Parameter Extraction
Step 1: Extract the required parameters 'n' and 'N' from RC comprising of n lines $L_1, L_2, ..., L_n$ and N gates $G_1, G_2, ..., G_N$ (here each gate is considered as each level).

Stage II: Generating Fault Array
Step 2: Initialize the stuck-at fault array(SaFarray), temporary array(temp_array) and fault detection array(fault_det).

$$SaFarray = [\,]; \quad temp_array = [\,]; \quad fault_det = [\,];$$

Step 3: Apply the binary input b_i $\{i = 1, 2, ..., n\}$ to the circuit corresponding to the decimal values varying from $d = 0 \; to \; 2^n - 1$.
Step 4: Update the fault detection array for each line L_i such that

$$\textbf{fault_det} = [x \;\; y] \; where \begin{cases} x = 1, y = 0; if \; b_i = 1 \\ x = 0, y = 1; if \; b_i = 0 \end{cases} \quad (a)$$

Step 5: Concatenate the temporary array with fault detection array.

$$temp_array = [temp_array, \quad fault_det]$$

The complexity from step 3 to step 5 is $2^n * nlog_2(n)$.
Step 6: For each target line of the gate G_k $\{k = 1, 2, ..., N\}$, check whether the output GO_i is 1 or 0 and update fault detection array.

$$\textbf{fault_det} = [x \;\; y] \; where \begin{cases} x = 1, y = 0; if \; GO_i = 1 \\ x = 0, y = 1; if \; GO_i = 0 \end{cases} \quad (b)$$

Step 7: Concatenate the temporary array with fault detection array.

temp_array = [temp_array, fault_det]

Step 8: Repeat steps 6 and 7 until N gates $G_1, G_2, ..., G_N$ are covered.
Step 9: Update the Stuck-at Fault Array for each binary input b_i applied, with temporary array.

SaFarray$[b_i]$ = **temp_array**

The complexity from step 6 to step 9 is $2^n * Nlog_2(n)$.
Step 10: Repeat steps 3 to 9 for each b_i and generate final SaFarray.
// The Complete Fault Array is created.
Hence the complexity at the end of stage II is $O([2^n(n + N)log_2(n))$

Stage III: CTS Generation
Step 11: Apply all possible combinations C_i of a set of values l_j (where j = 1 to r and r is the number of rows in SaFarray) taking t_m (where m = 1 to 2^N) at a time such that

$$(l_1 \mid l_2 \mid ....|l_j \,) \,/\, t_m = 1 \quad (c)$$

Here we perform bit-wise ORing of corresponding $l_j's$ taking t_m at a time of SaFarray, results in a row matrix containing all ones. The required CTS is that combination C_i satisfying the above condition c.

$$Test_vector = C_i \; iff \;\; c \; is \; satisfied$$

The complexity of stage III is $\sum_{r=1}^{n} {}^{2^n}C_r)$
Hence total complexity is $O([2^n(n+N)log_2(n)+\sum_{r=1}^{n} {}^{2^n}C_r])$

Various researchers have proposed online testing techniques to detect all single and multiple stuck-at faults which can detect the faults by designing new testable gates [10, 15, 20, 29, 30]. Among these techniques, the work in [20] has 100 % fault coverage with minimum gate overhead and all other techniques have partial fault coverage with high gate overhead. The offline techniques of testing will not add any overhead in terms of hardware complexity but they will have performance overhead since the circuit needs to be worked in test mode for offline testing.

2.4.2 Missing Gate Faults

Missing gate faults were proposed by the authors in [26] for the first time. They proposed a DFT methodology for detection of missing gate faults in *k*-CNOT gates. But this work does not deal with multiple missing gate faults. The complexity of the methodology developed depends on number of gates in the circuit and considers one gate missing at a time. Hence this method does not take multiple missing gate faults into consideration.

The authors in [6] have proposed an ATPG algorithm using linear programming model for detecting SMGF for circuits designed with *k*-CNOT gates. The authors in [25, 26] have proposed a technique of detecting SMGF in *k*-CNOT reversible circuit. An algorithm for CTS generation for all subset of missing gate faults was proposed in [11] for reversible circuits with *k*-CNOT gates. Also this technique has not dealt with stuck-at fault models. All the above mentioned methodologies were offline method of fault detection. The authors in [34] has proposed an online testing methodology which can detect missing gate faults when the circuit is operating in normal mode. But this can detect only odd number of repeated or missing gate faults. The offline testing techniques requires the circuit to operate in test mode for fault detection. The authors in [32] has proposed an offline testing methodology to test reversible circuits using Boolean Satisfiability (SAT) based approach which tests SMGF faults along with Single Missing Control Fault (SMCF) and Single Additional Control Fault (SACF). SMCF and SACF fault models are subsets of PMGF. The work in [35] has proposed a ping pong test to detect SMGF and multiple SMGF in a k-CNOT reversible circuit with 86 % fault coverage in average. In [19], the test set for SMGF is generated using a boolean difference method which has reported both test set and test vector set for detecting 100 % SMGF in a reversible circuit designed with *k*-CNOT gates.

So from the literature review, we can conclude that all the existing offline test methodologies have either concentrated on a particular fault type or adopted DFT for generating minimal test set and CTS. In this work, we target both stuck-at and super-set of missing gate fault detection in a reversible circuit designed with family of reversible gates and hence proposed algorithm does not restrict the testing of circuits designed with only *k*-CNOT gates.

3 Proposed Algorithms

3.1 Algorithm for Single and Multiple Stuck-at Fault Detection

In this algorithm, minimal CTS for single and multiple stuck-at faults are generated with Reversible Circuit (RC) as Input. The algorithm follows deterministic approach. We consider all the faults in each level of the circuit and generate minimal CTS and test sets which covers 100 % of faults.

3.2 Illustration/Analysis of Proposed SMSF Algorithm Using an Example

Let 3_17tc benchmark circuit is provided as input to the SMSF algorithm having number of lines $n = 3$ and number of gates $N = 6$. The complete flow of the algorithm to determine CTS for single and multiple Stuck-at faults is as represented in Fig. 2.

SaFarray obtained in Fig. 2 is an $2^n \times p$ matrix where p is the number of all possible Stuck-at Faults (including Sa-0 and Sa-1) for a given circuit. Figure 2 also shows all possible minimal test set to detect SMSF. Test vector is the n-bit binary equivalent of each element in the test set. Let us verify whether the test set [0 1 6] covers all the faults. From c we get

$$(l_0 \mid l_1 \mid l_6) = 1 \tag{d}$$

Table 2 shows the implementation of equation d. The last row indicates that this test set covers all possible SMSF faults in 3_17tc circuit.

Table 4 Comparison of CTS for Single and Multiple stuck-at faults in Reversible circuits containing k-CNOT gates

Bench mark circuits	No. of gates	No. of lines	Work in Ibrahim [8][a]		Work in Chakraborty [3][a]		Proposed work
			No of DFT Gates in [8]	No. of test vectors in [8]	No of DFT gates in [3]	No. of test vectors in [3]	No. of test vectors
2of5d1	18	6	8	2	7	3	2
4_49tc1	16	4	6	2	5	3	3
3_17tc	6	3	5	2	4	3	3
5mod5tc	17	6	7	2	7	3	3
6symd2	20	10	11	2	11	3	3
9symd2	28	12	13	2	13	3	3
Ham 3tc	5	3	4	2	4	3	3
Ham 7	23	7	8	2	8	3	3
Hwb4tc	17	4	5	2	5	3	3
Hwb5tc	55	5	6	2	6	3	4
Hwb6	126	6	7	2	7	3	4
Hwb7tc	289	7	8	2	8	3	4
Hwb8-637	637	8	10	2	9	3	5
Mod 5	8	5	6	2	6	3	3
Rd32	4	4	5	2	5	3	3
Rd53rcmg	30	7	9	2	8	3	2
Rd73d2	20	10	11	2	11	3	3
Rd84d1	28	15	16	2	16	3	3
Xor5	4	5	6	2	6	3	3

[a]with DFT

The same procedure is repeated for any circuit and applying appropriate equations required variables are determined. Figure 3 illustrates the SMSF test set generation for a Reversible circuit comprised of k-CNOT, FREDKIN and PERES gates.

Lemma 1 *Proposed Algorithm generates minimal CTS for a given Reversible circuit with 100 % fault coverage.*

Proof As explained in Section 3.2 and Table 2, for 3_17tc benchmark circuit 100 % fault coverage is achieved with number of test vectors = 3. Let us consider the mathematical induction method of proving this lemma. Hence it is sufficient to prove that 2 vectors are not enough to detect all possible SMSaF.Let the number of test vectors = 2. Total number of SMSF for 3_17tc benchmark circuit are 18. The fault coverage for all possible 2 vectors set is as shown in Table 3. Hence we have proved that 3 vectors are sufficient to detect all SMSaF which is minimal for 3_17tc benchmark circuit. In general, our proposed algorithm generated minimal CTS for all reversible circuits. □

Table 5 Comparison of CTS for single missing gate fault for benchmark circuits

Circuit	N	n	[23]	[7]			Proposed
				Gr.	B&B	DFT gate cost	
2of5d1	18	6	4	4	4	10	4
2of5d2	12	7	2	2	2	5	2
3_17tc	6	3	2	2	2	11	2
4_49tc1	16	4	3	3	3	7	3
5mod5tc	17	6	1	1	1	0	1
6symd2	20	10	2	2	2	10	2
9symd2	28	12	3	3	3	18	3
ham3tc	5	3	2	2	2	1	2
ham7tc	24	7	4	4	4	5	4
hwb4tc	17	4	2	2	2	7	2
hwb5tc	56	5	5	5	5	38	4
hwb6tc	126	6	8	9	8	83	9
hwb7tc	291	7	13	15	>4	190	14
mod5adders	21	6	3	3	3	8	3
mod5d1	8	5	1	1	1	0	1
mod5d2	9	5	1	1	1	0	1
rd32	4	4	2	2	2	1	2
rd53d1	12	7	2	2	2	3	2
rd53d2	12	8	2	2	2	5	2
rd53rcmg	30	7	3	4	3	19	3
rd73d2	20	10	3	3	3	11	3
rd84d1	28	15	3	3	3	14	3
xor5d1	4	5	1	1	1	0	1
ham15tc1	162	15	–	7	>2	47	7

3.3 Redundancy Detection and Removal

In the proposed algorithm, redundancy detection is performed by analyzing the entries in SaFarray, which contains the information on whether a particular vector detects all the fault or not. From the SaFarray, if any of the column contains all zeros then that particular fault is said to be redundant since the condition for fault detection is governed by equation c and hence to remove this redundant fault the algorithm ignores that column and processes further for test vector generation. Total number of redundant faults in the circuit is directly proportional to number of columns with all zero entries.

3.4 Algorithm for Partial and Complete Missing Gate Fault detection

Complete missing gate fault algorithm proposed in Algorithm 2 covers three different types of missing gate faults such as SMGF (Single Missing Gate Fault), MMGF (Multiple Missing Gate Fault) and RGF (Repeated Gate Fault) whereas Algorithm 3 generates minimal CTS for PMGF faults.

Table 6 Comparison of CTS for Single Missing Gate Fault with [19]

Benchmark circuit	N	n	SMGF test vectors [19]	Proposed work
ham3tc	5	3	3 or 4	2
rd32	4	4	6 or 7	2
xor5d1	4	5	2	1
3_17tc	6	3	2	2
mod5d1	8	5	2	1
4_49d3	12	4	4	3
hwb4d1	17	4	7	4
mod5d2	9	5	2	1
rd32d1	4	4	6 or 7	2
mod5d4	5	9	4	1

Algorithm 2 CTS for complete missing gate fault (SMGF / MMGF / RGF) model

Input: Reversible Circuit RC with 'n' lines and 'N' gates
Output: The minimal CTS and test vectors
Stage I: Input Parameter Extraction
Step 1: Extract the required parameters 'n' and 'N' from Reversible Circuit RC comprising of n lines $L_1, L_2, ..., L_n$ and N Gates $G_1, G_2, ..., G_N$ (here each gate is considered as each level).
Stage II: Generating Fault Array
Step 2: Initialize the complete missing gate fault array(CMGFarray), temporary array(temp_array) and fault detection array(fault_det).

$CMGFarray = [\,]; \quad temp_array = [\,]; \quad fault_det = [\,];$

Step 3: Apply the binary input $b_i\{i = 1,2,..., n\}$ to the circuit corresponding to the decimal values varying from $d = 1$ to 2^n.
Step 4: Compute the output at each gate (level) $G_k\{k = 1, 2, ..., N\}$. Let O_i and O_{i-1} be the n-bit binary output and input values for a gate G_k respectively.
Step 5: Update the fault detection array at each gate (level) appearance G_k such that

$$\textbf{fault_det} = [x] \; where \begin{cases} x = 1; if \; O_i \neq O_{i-1} \\ x = 0; if \; O_i = O_{i-1} \end{cases} \quad (e)$$

$$\textbf{fault_det} = [x] \; where \begin{cases} x = 1; if \; O_i \neq O_{i-2} \\ x = 0; if \; O_i = O_{i-2} \end{cases} \quad (f)$$

The fault_det for SMGF is calculated using equation e and for MMGF it is calculated using equation f.
Step 6: Concatenate the temporary array with fault detection array.

$temp_array = [temp_array, \quad fault_det]$

The complexity from step 3 to step 6 is $2^n log_2(n)$
Step 7: Repeat steps 5 and 6 until 'N' gates $G_1, G_2, ..., G_N$ are covered.
Step 8: Update the CMGF Array for each binary input b_i applied, with temporary array.

$\textbf{CMGFarray}[b_i] = \textbf{temp_array}$

Step 9: Repeat steps 3 to 8 for each b_i and generate final CMGFarray.
// The Complete Fault Array is created. The complexity from step 7 to step 9 is $2^n * Nlog_2(n)$ Hence the complexity at the end of stage II is $O(2^n(N + 1)log_2(n))$
Stage III: CTS Generation
Step 10: Apply all possible combinations C_i of a set of values l_j (where j = 1 to r and r is the number of rows in CMGFarray) taking t_m (where m = 1 to 2^N) at a time such that

$$(l_1 \mid l_2 \mid ....|l_j \,) \,/\, t_m = 1 \quad (g)$$

i.e., bit-wise ORing of corresponding l_js taking t_m at a time of CMGFarray, results in a row matrix containing all ones. The required CTS is that combination C_i satisfying the above condition g.

$Test_vector \;=\; C_i \; iff \;\; g \; is \; satisfied$

The complexity of stage III is $\sum_{r=1}^{n} \; {}^{2^n}C_r$
Hence total complexity is $O([2^n(N{+}1)log_2(n){+}\sum_{r=1}^{n} \; {}^{2^n}Cr])$

Algorithm 3 CTS for partial missing gate fault (PMGF) model

Input: Reversible Circuit RC with 'n' lines and 'N' gates
Output: The minimal CTS and test vectors
Stage I: Input Parameter Extraction
Step 1: Extract the required parameters n and N from Reversible Circuit RC comprising of n lines $L_1, L_2, ..., L_n$ and N gates $G_1, G_2, ..., G_N$ (here each gate is considered as each level).
Stage II: Generating Fault Array
Step 2: Initialize the partial missing gate fault array(PMGFarray), temporary array(temp_array) and fault detection array(fault_det).

$PMGFarray = [\,]; \quad temp_array = [\,]; \quad fault_det = [\,];$

Step 3: Apply the binary input $b_i\{i = 1, 2, ..., n\}$ to the circuit corresponding to the decimal values varying from $d = 1$ to 2^n.
Step 4: Compute the output at each gate (level) $G_k\{k = 1, 2, ..., N\}$. Let O_i and O_{i-1} be the n-bit binary output and input values for a gate G_k respectively.
Step 5: Update the fault detection array at each gate (level) appearance G_k such that

$$\textbf{fault_det} = [x] \; where \begin{cases} x = 0; if \; O_i \neq O_{i-1} \\ x = 1; if \; O_i = O_{i-1} \end{cases} \quad (h)$$

Step 6: Concatenate the temporary array with fault detection array.

$temp_array = [temp_array, \quad fault_det]$

The complexity from step 3 to step 6 is $2^n log_2(n)$
Step 7: Repeat steps 5 and 6 until 'N' gates $G_1, G_2, ..., G_N$ are covered.
Step 8: Update the PMGF Array for each binary input b_i applied, with temporary array.

$\textbf{PMGFarray}[b_i] = \textbf{temp_array}$

Step 9: Repeat steps 3 to 8 for each b_i and generate final PMGFarray.
// The Complete Fault Array is created.
The complexity from step 7 to step 9 is $2^n * Nlog_2(n)$
Hence the complexity at the end of stage II is $O(2^n(N + 1)log_2(n))$
Stage III: CTS Generation
Step 10: Apply all possible combinations C_i of a set of values l_j (where j = 1 to r and r is the number of rows in PMGFarray) taking t_m (where m = 1 to 2^N) at a time such that

$$(l_1 \mid l_2 \mid ....|l_j \,) \,/\, t_m = 1 \quad (i)$$

i.e., bit-wise ORing of corresponding l_js taking t_m at a time of PMGFarray, results in a row matrix containing all ones. The required CTS is that combination C_i satisfying the above condition i.

$Test_vector \;=\; C_i \; iff \;\; i \; is \; satisfied$

The complexity of stage III is $\sum_{r=1}^{n} \; {}^{2^n}C_r$
Hence total complexity is $O([2^n(N{+}1)log_2(n){+}\sum_{r=1}^{n} \; {}^{2^n}C_r])$

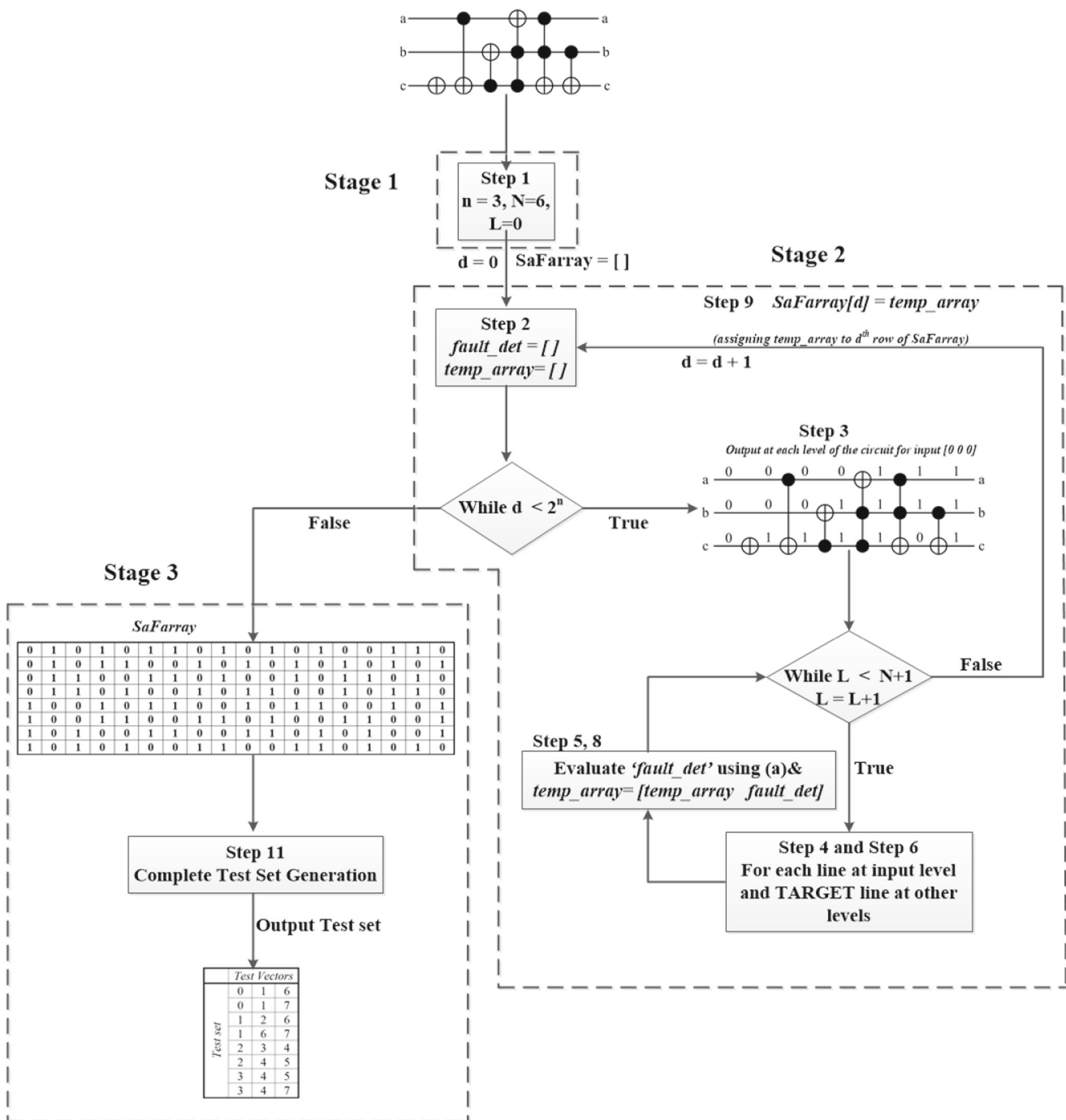
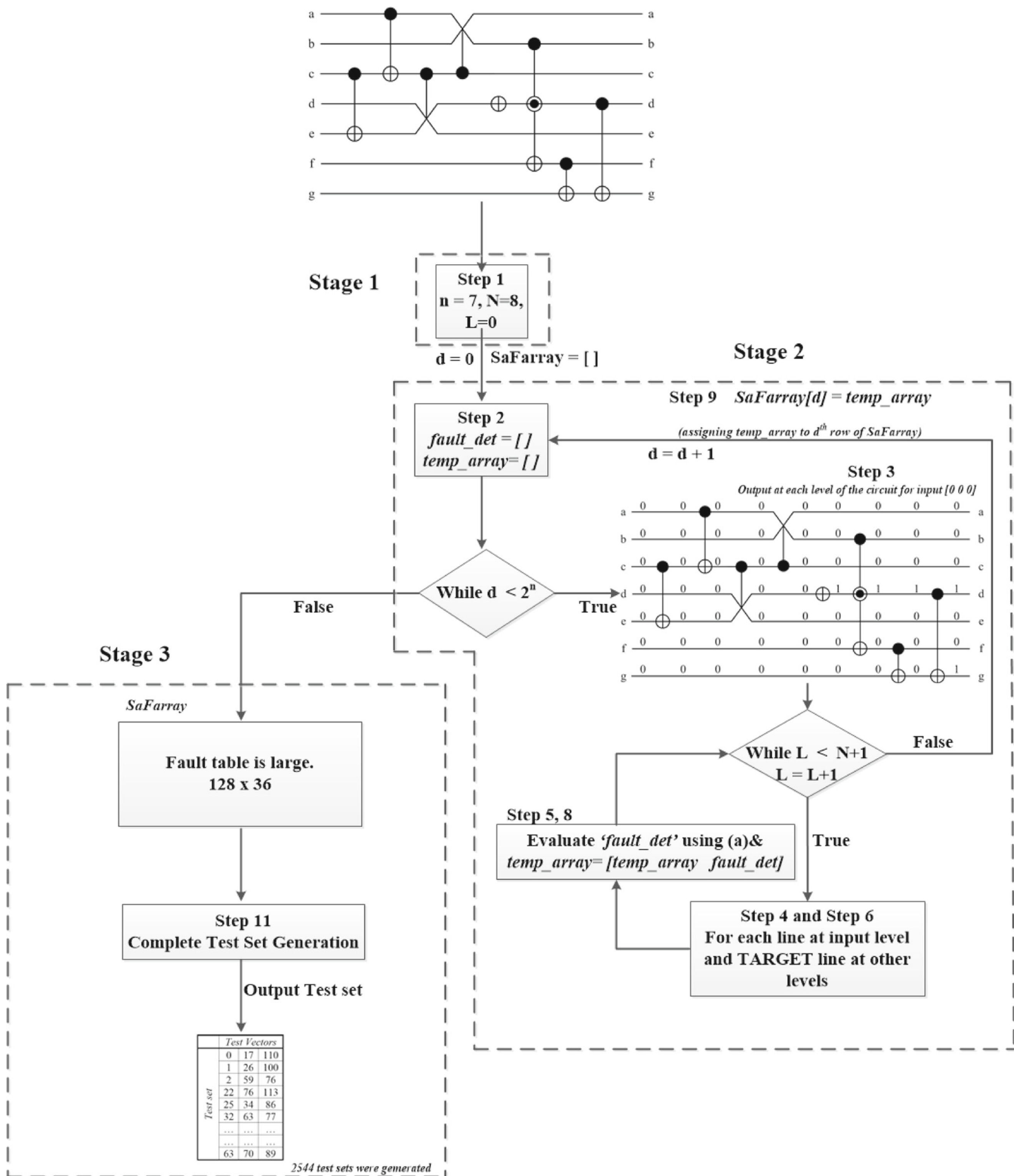


Fig. 2 Demonstration of SMSF algorithm for 3_17tc benchmark circuit

3.5 Illustration/Analysis of Proposed Complete Missing Gate Fault Algorithm with an Example

Let us consider 3_17tc benchmark circuit as an input circuit for the proposed algorithm to generate CTS to detect partial and complete missing gate faults. The algorithm flow remains the same as explained in Figs. 2 and 3 for SMSF algorithm. The same flow holds good for all proposed fault models but fault_det is evaluated using equation d for CMGF and equation h for PMGF. Hence Figs. 2 and 3 can be considered as a general flow diagram for all fault models but the final evaluation equation differs from one fault model to other.



Test set	Test Vectors		
	0	17	110
	1	26	100
	2	59	76
	22	76	113
	25	34	86
	32	63	77
	...	...	...
	...	...	...
	63	70	89

Fig. 3 Demonstration of SMSF algorithm for MBRC_2 circuit

4 Results and Discussions

The ATPG algorithms proposed in this work detects Single and Multiple stuck-at faults, SMGF, MMGF, PMGF and RGF exclusively or combining different fault models. The test sets are generated using exact approach and is found to be minimal compared to existing state-of-the-art work. The proposed SMSF has the computation complexity of

$O([2^n(n+N)log_2(n)+\sum_{r=1}^{n} {}^{2^n}C_r])$ and SMGF, RGF and PMGF has the complexity of $O([2^n(N+1)log_2(n)+\sum_{r=1}^{n} {}^{2^n}C_r])$. The complexity of an exact algorithm is exponential - polynomial and in our work, the exponential complexity is due to the generation of fault table and searching of minimal test set with CTS from this fault table. Since the size of fault table is $2^n \cdot L$ the search also has complexity of $O(2^n)$.

4.1 Single and Multiple Stuck-at Faults

Couple of attempts are made in the literature to generate test sets for stuck-at-fault models using DFT approach. In all these approaches there is an extra overhead due to adaptation of DFT method. In this work, there is no overhead as the design is completely exact and is found to be more optimal approach satisfying the requirements. The comparison of the proposed work with the existing works in [8] and [3] is described in Table 4. It is found that though the numbers of test vectors are one or two more than those proposed in [8], there is no extra overhead in terms of gate cost. Hence there is no area overhead in proposed work. Similarly, the required number of test vector in [3] is 3 for few benchmark circuits with an additional gate cost due to incorporation of DFT approach. Hence the proposed method generates minimal test vectors to detect all single and multiple stuck-at faults with no extra overhead. The proposed algorithm is applied to all the benchmark circuits and the results are tabulates along with the CPU simulation time as shown in Table 12.

4.2 Missing and Repeated Gate Faults

There are various variants of missing gate fault model. A complete gate or any control line or two or more consecutive gates may disappear to cause fault effect on the circuit. Contrary to the missing gate, there may be gates which are repeated two or more times causing functional errors in the circuit. The test vector to detect all these variants of fault types is not found in the literature. This work is supposed to

Table 7 Comparison of CTS for single missing gate fault with [35]

Benchmark circuit	N	n	SMGF test vectors in [35]	% fault coverage with [35]	Test vectors from proposed work with 100 % fault coverage
ham3tc	5	3	3	80.02	2
3_17tc	6	3	2	78.44	2
mod5d1	8	5	1	47.09	1
2of5d2	12	7	3	82.84	2
mod5adders	17	6	3	86.25	3
5mod5tc	17	6	1	48.63	1
ham15tc1	70	15	9	98.57	7
5mod5_10_10_71a	10	6	2	64.65	2
mspk_nth_primes4_11	11	4	4	96.92	2
mspk_4_49_14	14	4	6	84.86	2
mspk_nth_prime4_14	14	4	4	89.41	2
4b15g_3	15	4	4	90.78	3
cycle10_2	19	12	12	94.74	2
gf2 4mult_19_83	19	12	1	88.89	1
ham7_21_69	21	7	3	94.39	3
nth_prime5_inc_25_103	25	5	4	93.84	3
mspk_hwb5_31_91_opt_38_80	38	5	7	97.28	3
hwb6_47_107	47	6	7	97.78	3
nth_prime6_inc_55_667	55	6	33	99.98	6
nth_prime7_inc_1427_3172	1427	7	27	99.99	11
nth_prime8_inc_3346_7618	3346	8	73	99.99	–

Table 8 Comparison of CTS for SMGF,MMGF and PMGF for benchmark circuits

Circuit	N	n	MMGF		PMGF		SMGF, MMGF and PMGF		SMGF and PMGF	
			[23]	Proposed	[23]	Proposed	[23]	Proposed	[26]	Proposed
2of5d1	18	6	5	5	8	4	11	6	7	6
2of5d2	12	7	2	2	3	2	4	4	8	4
3_17tc	6	3	2	2	2	2	3	2	4	2
4_49tc1	16	4	3	3	5	3	5	4	5	4
5mod5tc	17	6	6	6	5	1	7	2	7	2
6symd2	20	10	2	3	3	2	4	4	11	4
9symd2	28	12	3	3	–	3	–	5	13	5
ham3tc	5	3	2	2	3	2	3	3	4	3
ham7tc	24	7	4	4	4	4	4	6	8	6
hwb4tc	17	4	4	4	5	2	6	4	5	4
hwb5tc	56	5	5	5	9	4	9	6	6	6
hwb6tc	126	6	8	9	15	9	16	13	7	13
hwb7tc	291	7	14	14	24	15	–	–	–	–
mod5adders	21	6	4	4	6	3	8	4	7	4
mod5d1	8	5	4	4	2	1	4	2	6	2
mod5d2	9	5	2	2	2	1	3	2	6	2
rd32	4	4	2	2	3	2	3	4	5	4
rd53d1	12	7	3	3	8	2	8	4	8	4
rd53d2	12	8	2	2	3	2	4	4	9	4
rd53rcmg	30	7	4	4	8	3	10	6	8	6
rd73d2	20	10	3	3	4	3	4	4	11	4
rd84d1	28	15	3	3	4	3	–	4	16	4
xor5d1	4	5	1	1	1	1	2	2	6	2

be first in the literature to generate test vectors for all these types of faults.

The authors in [7] have used Greedy and Branch & Bound algorithm to generate test vectors to detect SMGF with some additional gates due to DFT method. Our proposed algorithm also generates the test sets whose cardinality matches with those proposed in [7] with an advantage of zero overhead. The results are tabulated and compared in Table 5.

Table 9 Comparison of CTS for SMGF, MMGF and RGF for benchmark Circuits

Circuit	N	n	[9]	Proposed
2of5d1	18	6	5	4
4_49tc1	16	4	3	3
hwb4tc	17	4	4	2
rd53d1	12	7	3	2
rd53rcmg	30	7	4	3
mod5adders	21	6	4	3
5mod5tc	17	6	3	1
ham3tc	5	3	2	2

The authors in [23] have proposed an algorithm which do not uses any DFT approach and detects SMGF, MMGF and PMGF. The test set cardinality for the proposed SMGF algorithm also matches with the one proposed in [23], but for few benchmark circuits the proposed algorithm gives better test sets. The comparison of CTS for SMGF are as described in Table 5. The proposed MMGF and PMGF algorithms are found to give minimal test set when compared to the results in [23]. For $5mod5tc$ benchmark circuit, the number of test vectors needed to test PMGF is reduced by 80 % compared to [23].

The work in [19] targets for determining test set for SMGF. Table 6 gives the comparison of our work with [19]. On an average, test set cardinality is reduced by 60 %. Table 7 shows the comparison of the proposed work with [35]. The test set proposed in [35] has variations in fault coverage for different benchmark circuits. Compared to [35] the test vectors required to test a given reversible circuit is reduced by an average of 40 % with 100 % fault coverage.

Table 10 Comparison of CTS for SMGF, MMGF and PMGF for benchmark circuits

Circuit	N	n	PMGF			SMGF	
			SMCF- SAT [32]	SACF - SAT [32]	Proposed	SAT [32]	Proposed
One-two-three-v0_97	11	5	3[a]	3[a]	2	3	2
4gt4-v0_78	13	5	7	5	3	4	3
4gt12-v0_86	14	5	10	5[a]	2	3	2
Mini-alu_84(mini_alu_305)	20	10	5	7[a]	1	5	1
Rd53_131	28	7	11	9	5	5	5
Sym6_63(sym6_316)	29	14	10	16[a]	2	7	2
Sym9_148	210	10	120	57	1	32	1

[a]Test set with Un-testable faults

The comparison of the proposed work with the existing work is tabulated in Table 8 for MMGF and for PMGF. The number of test vectors needed to detect the combined effect of SMGF, MMGF and PMGF are compared with the work done in [23] and is tabulated in Table 8. It is found that, for $5of5d1$ benchmark circuit, there is an improvement of 45.45 % wrt [23] in number of test vectors needed to detect SMGF, MMGF and PMGF. The authors in [23] does not discusses on repeated gate faults and stuck-at faults.

Not much work have been found in the literature which discusses all types of fault models which takes combinations of all fault types. The authors in [26] discusses on test vector generation for the combined fault model of SMGF and PMGF. The proposed algorithm for combined effect of SMGF and PMGF is compared with [26] and tabulated as depicted in Table 8. The proposed algorithm generates minimal test sets compared to the counterpart with an improvement of 66.66 % for $xor5d1$ and 71.42 % for

Table 11 CTS with simulation time for reversible circuits designed with k-CNOT, Fredkin, and Peres gates

Circuits with Toffoli-Fredkin-Peres gates	# Test Vectors -,#Test Set,-,#Lines,-,#Gates Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
RFA_4	3-7974-13-16 81.5341	2-32-13-16 23.2026	2-32-13-16 24.0136	3-24-13-16 50.0567	3-32-13-16 52.2644	4-480-13-16 27.8082	4-344-13-16 49.7823
PBC_2	3-328-5-2 0.1291	1-16-5-2 0.0144	1-16-5-2 0.0167	3-328-5-2 0.1343	3-328-5-2 0.1302	2-256-5-2 0.0203	3-328-5-2 0.1365
RBC	3-44-4-4 0.0351	1-2-4-4 0.0157	1-2-4-4 0.0153	3-20-4-4 0.0426	3-24-4-4 0.0431	2-8-4-4 0.0174	3-8-4-4 0.0417
PBC_1	3-20-3-2 0.0115	1-2-3-2 0.0072	1-2-3-2 0.0067	3-10-3-2 0.0124	3-10-3-2 0.0124	2-4-3-2 0.0069	3-4-3-2 0.0125
MBRC_2	3-2560-7-8 6.7564	2-540-7-8 0.3357	2-540-7-8 0.3066	3-608-7-8 6.9523	3-900-7-8 6.9617	2-168-7-8 0.3199	3-270-7-8 6.8944
MBRC_4	3-9332-14-18 458.9209	2-48-14-18 (2 RF) 47.0389	2-48-14-18 (2 RF) 46.0430	4-48-14-18 (2 RF) 129.0482	3-44-14-18 (2 RF) 134.9654	2-256-14-18 (4 RF) 53.7973	3-40-14-18 (4 RF) 144.6939

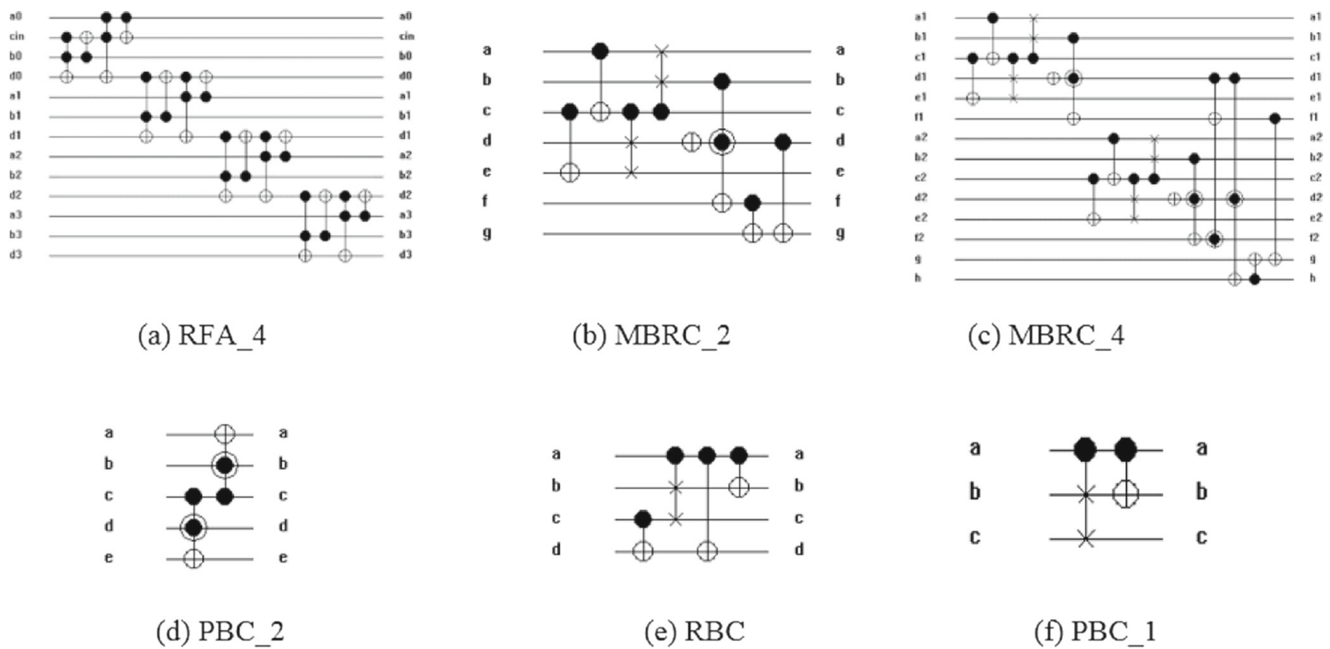


Fig. 4 Reversible Circuits designed using standard gates for validating the algorithm

$5mod5tc$ benchmark circuit. Though for one or two of the benchmark circuits the number of test vector are slightly more compared to [26], for most of the cases, the proposed algorithm is found to be optimal.

The author in [11] discusses on the test vector generation for combined effect of SMGF, MMGF and RGF whose results are compared with the proposed work as depicted in Table 9. It is found that the test set cardinality is improved by 50 % and 66.66 % for $hwb4tc$ and $5mod5tc$ benchmark circuits respectively when compared to [11]. Other fault models are not discussed and also it is applicable only for circuits containing k-CNOT gates.

The authors in [32] has defined variants of missing control faults as a Single Missing Control Fault (SMCF), Single Additional Control Fault (SACF) and Single Missing Gate Fault (SMGF). In the proposed work, both SMCF and SACF are subsets of PMGF and are detected using Algorithm 3. SMGF faults are detected using Algorithm 2. It is evident from Table 10 that the number of test vectors are reduced by 98.24 % for PMGF and 96.875 % for SMGF for $Sym9_148$ benchmark circuit. For some of the circuits in [32], there are some un-testable faults present which is not the case in the proposed work which detects all the faults present in the circuit with minimal test set cardinality.

The main objective of this work is to test circuits containing other reversible gates such as Fredkin, Peres and k-CNOT. Some of the reversible circuits designed using all these gates are considered and tested for all kinds of fault models. The results obtained are tabulated in Table 11. The reversible circuits for validating the proposed algorithms are as shown in Fig. 4.

The CTS for most of the benchmark circuits along with the simulation time for Single/Multiple Stuck-at Fault, SMGF, MMGF, PMGF and their combinations are tabulated in Table 12. # Test Vectors indicates the number of minimal test vectors needed to test a particular fault and #Test Set indicates CTS i.e., all possible minimal test vector sets. #Lines and #Gates indicate number of lines and gates present in a circuit under test.

In this work, we have proposed an ATPG algorithm using an exact approach to generate minimal test vectors with CTS for a given reversible circuit consisting of family of reversible gates. Based on these proposed algorithms, a testing tool is developed which can take inputs either in the form of circuit image (.bmp, .jpg, .png, .fig) or in .tfc or .real format. This tool is capable of generating test vectors for different fault models such as Single and Multiple stuck-at fault, Missing gate fault model and its variants. The tool generates the all possible test vectors needed to test a particular fault in a circuit indicating the number of lines and number of gates in the circuit along with CPU time required for processing. There is an option to write the results in to

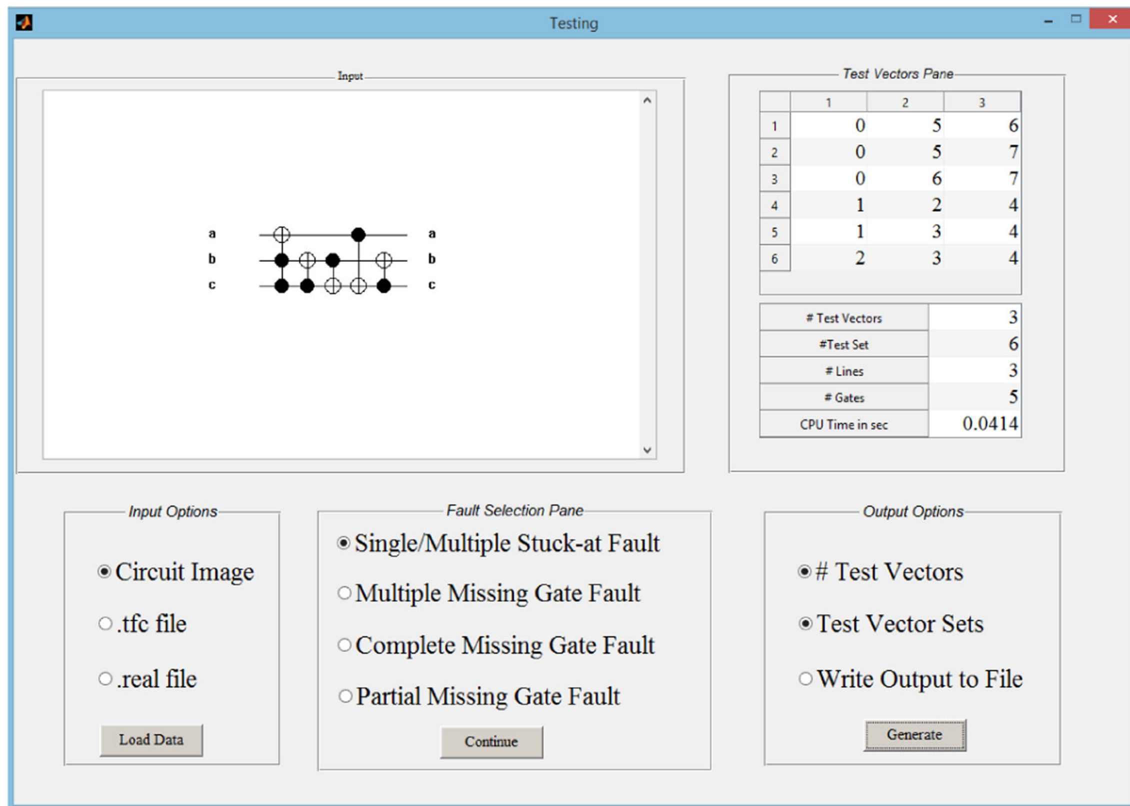


Fig. 5 Snapshot of Testing Tool for $ham3tc$ as input circuit

a text file for future reference. The tool also display the circuit under test at the left pane. The snapshots of the tool for $ham3tc$ benchmark circuit as input is as shown in Fig. 5.

5 Conclusion and Future Work

In this paper, we have developed a generalized platform to generate CTS for various fault models in a reversible circuit comprising of family of reversible gates. This is the first attempt in literature which combines all types of fault models and for circuits designed using family of reversible gates. So far in the existing state-of-the-art approaches, only k-CNOT gate is taken into consideration and any one or two fault models are combined for offline testing. In this work, we have also considered other standard reversible gates such as FREDKIN and PERES gates along with k-CNOT gates. All variants of stuck-at faults and missing Gate Faults are tested either exclusively or by grouping. A testing tool is developed based on the proposed ATPG algorithms. The tool is developed with flexibility in providing inputs and detailed analysis in output. The proposed algorithm is implemented in MATLAB 2011a on an Intel Core $i7-3612QM$ processor with $2.10GHz$ processing speed. The CPU simulation time quoted in this work is strictly based on the processor specifications mentioned. The algorithm can be used independent of the implementation technology. Because of the exponential complexity of algorithm, for some benchmark circuits that have large gate count and lines, we are not able to use our tool successfully. In our future work we will report some heuristic approach for reducing the time complexity of test generation based on a divide-and-conquer strategy using circuit partitioning. Hence our future work includes reducing the time complexity of test generation by using Heuristic approaches and to work on circuit partitioning methods for large circuits which will divide the exponential complexity of the algorithm. Our future work also include detection of other quantum circuit fault models.

Acknowledgments Authors like to thank PES Institute of Technology, Bengaluru, INDIA for the support provided during this work.

Appendix

Table 12 CTS and minimal test set with Simulation Time for the benchmark Circuits

MASLOV	# Test Vectors -,#Test Set,-,#Lines,-,#Gates						
benchmark Circuits	Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
2-4dec	2-4-6-3	1-2-6-3	1-2-6-3	3-320-6-3	3-522-6-3	2-8-6-3	3-56-6-3
	0.3130	0.1315	0.1205	0.9859	0.9893	0.1616	0.9884
2of5d1	2-5-6-18	4-23-6-18	4-23-6-18	4-10-6-18	4-13-6-18	6-1-6-18	6-16-18
	0.2338	14.8834	14.7497	15.9838	17.1745	16.4448	16.6221
2of5d1s	2-5-6-15	3-22-6-15	3-22-6-15	3-5-6-15	3-2-6-15	4-33-6-15	4-19-6-15
	0.1893	0.9580	0.9711	1.1526	1.1602	14.9549	17.6501
2of5d2	3-564-7-12	2-12-7-12	2-12-7-12	4-2-7-12	3-30-7-12	4-25-7-12	4-9-7-12
	6.8528	0.3958	0.3889	7.2525	7.1275	6.8231	7.4051
2of5d3	3-126-6-17	3-64-6-17	3-64-6-17	3-2-6-17	3-2-6-17	6-9-6-17	6-7-6-17
	1.0346	0.9904	1.0045	1.2268	1.1925	15.3202	16.6217
3_17tc	3-8-3-6	2-5-3-6	2-5-3-6	3-2-3-6	3-4-3-6	2-1-3-6	3-1-3-6
	0.0182	0.0174	0.0164	0.0267	0.0261	0.0158	0.0302
4_49_fc	3-9-4-12	2-3-4-12	2-3-4-12	4-38-4-12	3-1-4-12	4-12-4-12	4-2-4-12
	0.0700	0.0500	0.0529	0.1567	0.1018	0.1250	0.1571
4_49-12-32	3-13-4-12	3-38-4-12	3-38-4-12	4-102-4-12	4-108-4-12	4-29-4-12	4-15-4-12
	0.0589	0.0585	0.0568	0.1526	0.1491	0.1070	0.1411
4_49tc1	3-1-4-16	3-15-4-16	3-15-4-16	4-24-4-16	4-36-4-16	4-7-4-16	4-3-4-16
	0.0796	0.0724	0.0701	0.1678	0.1654	0.1201	0.1650
4b15g_1	3-2-4-15	3-7-4-15	3-7-4-15	4-46-4-15	4-46-4-15	4-5-4-15	4-2-4-15
	0.0627	0.0614	0.0593	0.1694	0.1585	0.1210	0.1570
4b15g_2	3-17-4-15	4-37-4-15	4-37-4-15	4-12-4-15	4-13-4-15	6-76-4-15	6-76-4-15
	0.0612	0.1212	0.1121	0.1538	0.1624	0.6048	0.6851
4b15g_3	3-9-4-15	3-11-4-15	3-11-4-15	4-39-4-15	4-38-4-15	3-2-4-15	4-22-4-15
	0.0785	0.0647	0.0592	0.1529	0.1571	0.0593	0.1622
4b15g_4	3-21-4-15	3-7-4-15	3-7-4-15	4-55-4-15	4-59-4-15	5-70-4-15	5-57-4-15
	0.0875	0.0620	0.0606	0.1515	0.1597	0.2682	0.3130
4b15g_5	3-10-4-15	2-1-4-15	2-1-4-15	3-2-4-15	4-53-4-15	4-17-4-15	4-6-4-15
	0.0626	0.0469	0.0445	0.0970	0.1549	0.1112	0.1552
5bitadder	3-451-11-29	2-2-11-29	2-2-11-29	4-2-11-29	3-2-11-29	2-4-11-29	4-4-11-29
	29.6312	9.3405	9.5173	18.5539	18.7116	9.2762	19.1209
5mod5_fc	3-912-6-10	2-64-6-10	2-64-6-10	3-72-6-10	3-80-6-10	2-4-6-10	3-8-6-10
	0.9418	0.1425	0.1432	1.0534	1.0319	0.1426	1.0875
5mod5-8	3-2184-6-8	1-2-6-8	1-2-6-8	3-152-6-8	3-198-6-8	2-4-6-8	3-44-6-8
	0.9142	0.0906	0.0837	0.9970	0.9913	0.1154	1.0155
5mod5-10-71a	3-1328-6-10	2-64-6-10	2-64-6-10	3-80-6-10	3-120-6-10	3-128-6-10	3-24-6-10
	0.9500	0.1426	0.1426	1.0529	1.0600	0.9230	1.0627
5mod5tc	3-472-6-17	1-2-6-17	1-2-6-17	3-4-6-17	3-26-6-17	2-4-6-17	3-4-6-17
	1.0532	0.1963	0.1955	1.2351	1.2639	0.2221	1.2763
6symd2	3-649-10-20	2-2-10-20	2-2-10-20	4-4-10-20	4-4-10-20	4-20-10-20	4-8-10-20
	9.8886	3.3090	3.2846	6.4955	6.6713	3.5032	6.8259

Table 12 (continued)

MASLOV	# Test Vectors -,#Test Set,-,#Lines,-,#Gates						
benchmark Circuits	Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
9symd2	3-1369-12-28	3-2-12-28	3-2-12-28	4-2-12-28	4-2-12-28	5-42-12-28	5-50-12-28
	48.7271	18.4493	18.4373	37.2094	36.5676	18.8827	38.1186
cycle10_2	3-3582-12-19	2-4-12-19	2-4-12-19	4-4-12-19	3-1-12-19	4-15-12-19	5-13-12-19
	34.5715	14.9016	14.8557	29.3721	29.2128	14.5255	29.7550
gf2^4mult_19_83	3-3524-12-19	1-2-12-19	1-2-12-19	3-2-12-19	3-2-12-19	2-8-12-19	3-4-12-19
	38.6134	12.4453	12.4530	25.4651	26.1726	12.8029	26.0375
gf2^5mult_29_129	3-26877-15-29	1-2-15-29	1-2-15-29	3-2-15-29	3-2-15-29	2-16-15-29	3-16-15-29
	1218.7000	176.9768	170.4569	395.4593	398.9618	162.6005	398.2599
ham3_fc	3-9-3-4	1-1-3-4	1-1-3-4	3-2-3-4	3-4-3-4	2-3-3-4	3-2-3-4
	0.4213	0.1543	0.1917	0.1364	0.1390	0.1290	0.1389
ham3tc	3-6-3-5	2-4-3-5	2-4-3-5	3-4-3-5	3-3-3-5	3-8-3-5	3-2-3-5
	0.0207	0.0162	0.0139	0.0265	0.0269	0.0169	0.0264
ham7-21-69	3-96-7-21	3-2856-7-21	3-2856-7-21	4-1-7-21	4-1-7-21	5-4-7-21	5-4-7-21
	7.2371	6.7703	6.7480	7.6630	7.6881	6.9831	7.8969
ham7-25-49	3-48-7-25	3-4676-7-25	3-4676-7-25	4-1-7-25	4-1-7-25	4-4-7-25	4-1-7-25
	7.4618	7.0233	6.9395	7.9281	7.9289	7.1432	8.2473
ham7tc	3-160-7-23	4-3-7-23	4-4-7-23	4-2-7-23	4-3-7-23	6-12-7-23	6-12-7-23
	7.1260	6.9357	6.8309	8.0462	7.9583	7.2269	8.1977
ham15-70	–	8-12-15-70	8-12-15-70	–	–	12-64-15-70	–
		404.5184	411.1473	–	–	443.9390	
ham-15-109-214	–	4-2-15-109	4-4-15-109	–	–	7-24-15-109	–
		626.3074	634.1340	–	–	645.9290	
ham15tc1	–	7-3-15-132	7-7-15-132	–	–	13-24-15-132	–
		794.1442	804.5426	–	–	823.1376	
hwb4_fc	2-1-4-9	2-4-4-9	2-4-4-9	4-33-4-9	3-2-4-9	3-8-4-9	4-6-4-9
	0.6674	0.2061	0.1653	0.2774	0.2080	0.1718	0.2665
hwb4-11-21	3-5-4-11	2-8-4-11	2-8-4-11	4-156-4-11	3-3-4-11	3-16-4-11	4-47-4-11
	0.0717	0.0416	0.0436	0.1475	0.0895	0.0619	0.1452
hwb4-11-23	3-4-4-11	2-3-4-11	2-3-4-11	3-1-4-11	4-157-4-11	3-16-4-11	4-57-4-11
	0.0640	0.0382	0.0369	0.0891	0.1437	0.0544	0.1468
hwb4tc	3-12-4-17	2-1-4-17	2-1-4-17	3-1-4-17	4-56-4-17	4-28-4-17	4-13-4-17
	0.0839	0.0580	0.0540	0.1162	0.1754	0.1370	0.1723
hwb5_31_91	4-25-5-31	3-13-5-31	3-13-5-31	4-25-5-31	4-20-5-31	4-21-5-31	5-483-5-31
	1.8237	0.3717	0.4166	1.6166	1.5967	1.4056	9.0708
hwb5_fc	2-1-5-16	3-3-5-16	3-3-5-16	4-2-5-16	5-126-5-16	4-15-5-16	5-9-5-16
	0.3043	0.3025	0.2677	1.4759	8.3768	1.2578	8.3818
hwb5-24-114	3-6-5-24	3-2-5-24	3-2-5-24	4-20-5-24	4-12-5-24	4-17-5-24	4-1-5-24
	0.2784	0.2810	0.2699	1.4557	1.4624	1.3020	1.4726
hwb5ps	3-66-8-23	4-2-8-23	4-2-8-23	5-1-8-23	5-2-8-23	5-3-8-23	5-1-8-23
	2.7389	1.2392	1.2289	2.4192	2.4199	1.2221	2.4304
hwb5tc	4-327-5-55	4-1-5-55	4-1-5-55	5-4-5-55	5-6-5-55	6-6-5-55	8-2-5-55
	1.5777	1.4777	1.4656	9.5898	9.5420	45.3169	55.7468
hwb6_47_107	4-4272-6-47	3-144-6-47	3-144-6-47	4-193-6-47	4-63-6-47	4-207-6-47	4-1-6-47
	17.6966	1.5457	1.5443	18.9772	19.1182	17.3831	20.5470

Table 12 (continued)

MASLOV	# Test Vectors -,#Test Set,-,#Lines,-,#Gates						
benchmark Circuits	Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
hwb6_fc	2-1-6-25	4-2-6-25 (3 RF)	4-2-6-25 (3 RF)	6-2-6-25 (3 RF)	5-2-6-25 (3 RF)	5-8-6-25 (6 RF)	6-2-6-25 (6 RF)
	0.5425	16.2640	16.0012	18.2514	18.4219	16.3194	18.6992
hwb6-42-150	4-2519-6-42	4-223-6-42	4-223-6-42	4-1-6-42	4-1-6-42	5-2-6-42	6-2-6-42
	17.2462	16.1501	16.0682	18.5645	18.4655	16.9252	19.9843
hwb6ps	3-118-9-27	4-1-9-27	4-1-9-27	5-1-9-27	6-1-9-27	5-1-9-27	6-2-9-27
	5.5933	2.5563	2.5970	5.0624	5.0522	2.5891	5.0988
hwb6tc	4-449-6-126	9-1-6-126	9-1-6-126	9-1-6-126	10-1-6-126	13-1-6-126	12-1-6-126
	22.1752	18.7732	18.6450	26.8373	27.1740	22.5011	32.1969
hwb7_fc	2-1-7-38	4-1-7-38	4-2-7-38	6-3-7-38	6-2-7-38	5-2-7-38	7-7-7-38
	1.3112	7.7146	7.8096	10.4030	10.2875	8.0551	10.9789
hwb7-236	4-12-7-236	12-1-7-236	13-1-7-236	13-1-7-236	12-1-7-236	21-1-7-236	19-1-7-236
	19.2195	16.5462	16.2939	33.8152	31.4848	955.4152	1075.6300
hwb7-331-2609a	5-74-7-331	12-1-7-331	13-2-7-331	12-1-7-331	13-2-7-331	18-1-7-331	19-2-7-331
	25.5628	19.4453	20.3498	39.7577	37.0916	571.5173	1574.3589
hwb7ps	3-86-10-35	5-2-10-35	5-2-10-35	6-2-10-35	6-2-10-35	6-1-10-35	8-1-10-35
	14.7413	6.5048	6.4758	12.8333	12.8146	6.5518	14.3243
hwb7tc	4-13-7-289	14-1-7-289	15-1-7-289	14-1-7-289	15-1-7-289	–	–
	22.3727	16.8538	18.8977	31.0356	39.2918	–	–
hwb8-749-6197a	5-5-8-749	18-1-8-749	18-1-8-749	19-1-8-749	19-1-8-749	–	–
	56.8533	181.7448	106.3085	500.0458	485.4238	–	
hwb8ps	3-315-12-38	5-2-12-38	5-2-12-38	6-2-12-38	6-2-12-38	6-2-12-38	7-5-12-38
	86.1770	31.8000	30.9391	63.7427	64.1434	30.2390	64.5245
hwb9ps	3-255-13-57	5-2-13-57	6-4-13-57	6-1-13-57	7-4-13-57	7-4-13-57	8-4-13-57
	315.4491	96.7060	102.0703	206.2935	203.3331	96.4637	201.5791
hwb10ps	3-480-14-62	5-1-14-62	5-1-14-62	7-3-14-62	7-3-14-62	8-4-14-62	7-1-14-62
	917.0504	209.4136	211.1549	460.6636	464.0232	210.2953	463.9036
hwb11ps	3-574-15-73	7-1-15-73	6-1-15-73	9-1-15-73	7-1-15-73	8-4-15-73	8-1-15-73
	3302.0000	557.6415	492.8986	1162.6000	1148.4000	576.6186	1291.8000
mod5adder-15	2-1-6-15	2-4-6-15	2-4-6-15	3-3-6-15	3-11-6-15	4-457-6-15	4-122-6-15
	0.1978	0.1916	0.2021	1.1892	1.2373	15.4477	16.5944
mod5adder-17-81	3-1131-6-17	2-8-6-17	2-8-6-17	3-6-6-17	3-8-6-17	3-16-6-17	4-392-6-17
	1.0706	0.2177	0.2256	1.2481	1.2754	1.0534	17.0354
mod5adders	3-1064-6-21	3-57-6-21	3-57-6-21	4-645-6-21	3-4-6-21	4-344-6-21	4-136-6-21
	1.1206	1.0567	1.0768	17.0445	1.3358	16.1636	17.0355
mod5d1	3-134-5-8	1-2-5-8	1-2-5-8	3-16-5-8	3-26-5-8	2-4-5-8	3-10-5-8
	0.1866	0.0479	0.0486	0.2225	0.2447	0.0844	0.2070
mod5d2	3-176-5-9	1-2-5-9	1-2-5-9	3-66-5-9	3-86-5-9	2-4-5-9	3-20-5-9
	0.1792	0.0604	0.0504	0.2148	0.2482	0.0580	0.2207
mod5d4	3-192-5-5	2-72-5-5	2-72-5-5	3-128-5-5	3-128-5-5	2-32-5-5	3-128-5-5
	0.1730	0.0448	0.0449	0.1689	0.2025	0.0445	0.2036
mod5mils	3-432-5-5	1-2-5-5	1-2-5-5	3-160-5-5	3-176-5-5	2-16-5-5	3-64-5-5
	0.1516	0.0384	0.0498	0.1799	0.1775	0.0429	0.2059

Table 12 (continued)

MASLOV benchmark Circuits	# Test Vectors -,#Test Set,-,#Lines,-,#Gates						
	Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
mspk_4_49_12	3-13-4-12	3-38-4-12	3-38-4-12	4-102-4-12	4-108-4-12	4-29-4-12	4-15-4-12
	0.2868	0.1739	0.1627	0.2388	0.2561	0.2203	0.2602
mspk_4_49_13	3-6-4-13	2-1-4-13	2-1-4-13	3-1-4-13	3-1-4-13	3-2-4-13	4-23-4-13
	0.0781	0.0510	0.0551	0.1058	0.1075	0.0736	0.1708
mspk_4_49_14	3-3-4-14	2-4-4-14	2-4-4-14	4-89-4-14	4-87-4-14	3-6-4-14	4-21-4-14
	0.0767	0.0544	0.0571	0.1757	0.1668	0.0724	0.1659
mspk_4b15g_1	4-258-4-15	2-1-4-15	2-1-4-15	4-50-4-15	4-58-4-15	4-11-4-15	4-7-4-15
	0.1401	0.0585	0.0552	0.1818	0.1808	0.1290	0.1680
mspk_4b15g_2	3-2-4-15	3-25-4-15	3-25-4-15	4-69-4-15	4-67-4-15	3-4-4-15	4-28-4-15
	0.0772	0.0727	0.0734	0.1689	0.1722	0.0773	0.1728
mspk_4b15g_3	3-2-4-15	2-2-4-15	2-2-4-15	4-77-4-15	3-1-4-15	3-2-4-15	4-27-4-15
	0.0821	0.0544	0.0579	0.1707	0.1106	0.0759	0.1762
mspk_4b15g_4	3-2-4-15	3-13-4-15	3-13-4-15	4-52-4-15	4-47-4-15	4-57-4-15	4-9-4-15
	0.0814	0.0744	0.0721	0.1741	0.1788	0.1390	0.1756
mspk_4b15g_5	3-1-4-15	3-19-4-15	3-19-4-15	4-56-4-15	4-50-4-15	3-2-4-15	4-14-4-15
	0.0816	0.0775	0.0789	0.1719	0.1756	0.0729	0.1753
mspk_hwb4_12	4-335-4-12	2-1-4-12	2-1-4-12	4-135-4-12	4-114-4-12	3-12-4-12	4-49-4-12
	0.1366	0.0504	0.0459	0.1642	0.1529	0.0648	0.1612
mspk_hwb4_13	3-1-4-13	2-1-4-13	2-1-4-13	4-120-4-13	4-91-4-13	3-6-4-13	4-38-4-13
	0.0718	0.0547	0.0477	0.1620	0.1604	0.0688	0.1549
mspk_nth_prime_inc_29_91_opt_36_80	4-433-5-36	3-28-5-36	3-28-5-36	4-23-5-36	4-22-5-36	4-8-5-36	5-178-5-36
	1.4706	0.5002	0.4984	1.7081	1.7118	1.4573	8.6296
mspk_nth_primes4_11	3-12-4-11	3-5-4-11	3-5-4-11	4-10-4-11	4-30-4-11	5-4-4-11	5-4-4-11
	0.0687	0.0622	0.0669	0.1464	0.1439	0.2748	0.3216
mspk_nth_primes4_12	3-5-4-12	2-3-4-12	2-3-4-12	3-1-4-12	4-88-4-12	2-1-4-12	4-38-4-12
	0.0666	0.0482	0.0516	0.0869	0.1597	0.0629	0.1561
mspk_nth_primes4_13	3-3-4-13	2-1-4-13	2-1-4-13	4-75-4-13	4-56-4-13	4-31-4-13	4-7-4-13
	0.0673	0.0476	0.0476	0.1555	0.1814	0.1248	0.1476
mspk_nth_primes4_14	4-266-4-14	2-2-4-14	2-2-4-14	4-84-4-14	4-61-4-14	3-4-4-14	4-17-4-14
	0.1504	0.0474	0.0486	0.1820	0.1818	0.0727	0.1824
nth_prime3_inc	3-6-3-4	1-1-3-4	1-1-3-4	3-4-3-4	3-6-3-4	2-3-3-4	3-4-3-4
	0.223	0.0118	0.0110	0.0270	0.0188	0.0115	0.0199
nth_prime4_inc_15_51	3-1-4-15	3-20-4-15	3-20-4-15	4-43-4-15	4-41-4-15	3-2-4-15	4-7-4-15
	0.0686	0.0603	0.0698	0.1708	0.1931	0.0634	0.1862
nth_prime4_inc_d1	3-9-4-12	3-11-4-12	3-11-4-12	4-36-4-12	3-1-4-12	4-2-4-12	4-1-4-12
	0.0551	0.0574	0.0519	0.1421	0.1093	0.1370	0.1430
nth_prime4_inc_d2	3-13-4-11	3-5-4-11	3-5-4-11	4-10-4-11	4-31-4-11	5-4-4-11	5-4-4-11
	0.0611	0.0498	0.0516	0.1611	0.1356	0.2911	0.3084
nth_prime5_inc_25_103	3-6-5-25	3-4-5-25	3-4-5-25	4-13-5-25	4-23-5-25	5-34-5-25	5-16-5-25
	0.2827	0.2458	0.2511	1.3585	1.4237	7.0901	7.8481
nth_prime5_inc_29_91	3-1-5-29	3-6-5-29	3-6-5-29	4-10-5-29	4-23-5-29	5-188-5-29	5-69-5-29
	0.2813	0.2716	0.2927	1.4132	1.4425	7.0739	8.0254
nth_prime6_inc_55_667	4-9607-6-55	6-2-6-55	6-3-6-55	7-2-6-55	7-1-6-55	10-1-6-55	10-1-6-55
	17.9921	16.2976	16.3334	19.5197	19.4009	17.2807	21.1762

Table 12 (continued)

MASLOV benchmark Circuits	# Test Vectors -,#Test Set,-,#Lines,-,#Gates / Test Vector Generation Time (sec)						
Name	Single/Multiple Stuck-at Fault [i]	Complete Missing Gate Fault [ii]	Partial Missing Gate Fault [iii]	[i] and [ii]	[i] and [iii]	[ii] and [iii]	[i], [ii] and [iii]
nth_prime7_inc_1427_3172	7-84-7-1427 107.9474	11-2-7-1427 58.7779	11-1-7-1427 55.3596	12-1-7-1427 130.7828	11-1-7-1427 138.2252	13-3-7-1427 84.6795	14-3-7-1427 192.7920
Rd32	3-48-4-4 0.0408	2-16-4-4 0.0202	2-16-4-4 0.0196	3-16-4-4 0.0442	3-32-4-4 0.0463	4-256-4-4 0.0958	4-160-4-4 0.1003
Rd53-16-67	3-672-7-16 6.9771	2-48-7-16 0.4741	2-48-7-16 0.4497	3-8-7-16 7.3917	3-16-7-16 7.3146	4-7-7-16 6.8503	4-7-7-16 7.6004
Rd53d1	3-1888-7-12 6.8874	2-32-7-12 0.3837	2-32-7-12 0.3958	4-4-7-12 7.1870	3-64-7-12 7.1461	4-15-7-12 6.7558	4-17-7-12 7.3302
Rd53d1mils	3-692-7-16 6.8708	2-112-7-16 0.4530	2-112-7-16 0.4603	3-24-7-16 7.5159	3-28-7-16 7.4095	2-16-7-16 0.4618	3-8-7-16 7.5010
Rd53d2	3-190-8-12 1.2279	2-2-8-12 0.5007	2-2-8-12 0.5040	4-2-8-12 0.9970	3-2-8-12 1.0094	4-14-8-12 0.6018	4-14-8-12 1.0865
Rd53d15	3-2536-7-28 7.3589	5-31-7-28 6.9239	5-18-7-28 6.9174	5-25-7-28 8.1426	5-22-7-28 8.1461	6-14-7-28 7.3177	6-9-7-28 8.5092
Rd53d15s	3-1496-7-20 7.1606	4-8-7-20 6.7383	4-8-7-20 6.7833	5-5-7-20 7.7242	5-8-7-20 7.6860	6-14-7-20 7.0316	6-13-7-20 7.8442
Rd53rcmg	2-2-7-30 0.6912	3-3-7-30 7.1627	3-3-7-30 7.1263	4-5-7-30 8.2370	4-4-7-30 8.2662	6-1-7-30 7.1952	6-2-7-30 8.3856
Rd73d2	3-622-10-20 9.6205	3-4-10-20 3.5469	3-4-10-20 3.4140	4-4-10-20 6.8204	4-4-10-20 6.7236	4-12-10-20 3.2534	4-4-10-20 6.6281
Rd84d1	3-14695-15-28 1160.7234	3-8-15-28 155.3735	3-9-15-28 160.2758	4-8-15-28 367.3788	4-8-15-28 379.2576	4-12-15-28 158.1981	5-16-15-28 368.6052
Xor5d1	3-256-5-4 0.1376	1-2-5-4 0.0256	1-2-5-4 0.0245	3-256-5-4 0.1988	3-256-5-4 0.1562	2-72-5-4 0.0323	3-256-5-4 0.1658

References

1. Bennett C (1973) Logical reversibility of computation. IBM J Res Dev 17(6):525–532. doi:10.1147/rd.176.0525
2. Bushnell M, Agrawal VD (2000) Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits, vol 17. Springer Science & Business Media
3. Chakraborty A (2005) Synthesis of reversible circuits for testing with universal test set and c-testability of reversible iterative logic arrays. In: Proceedings of 18th international conference on VLSI design, VLSID. IEEE, pp 249–254
4. Chaves JF, Silva DS, Camargos VV, Vilela Neto OP (2015) Towards reversible QCA computers: reversible gates and ALU. In: Proceedings of IEEE 6th Latin American symposium on circuits & systems, (LASCAS). IEEE, pp 1–4
5. Donald J, Jha NK (2008) Reversible logic synthesis with fredkin and peres gates. J Emerg Technol Comput Syst 4(1):2:1–2:19
6. Fang-ying X, Han-wu C, Wen-jie L, Zhi-giang L (2008) Fault detection for single and multiple missing-gate faults in reversible circuits. In: Proceedings of IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), pp 131–135. doi:10.1109/CEC.2008.4630787
7. Hayes J, Polian I, Becker B (2004) Testing for missing-gate faults in reversible circuits. In: Proceedings of 13th Asian test symposium, pp 100–105. doi:10.1109/ATS.2004.84
8. Ibrahim M, Chowdhury A, Babu H (2008) Minimization of cts of k-cnot circuits for ssf and msf model. In: Proceedings of IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, DFTVS'08, pp 290–298. doi:10.1109/DFT.2008.38
9. Ibrahim M, Chowdhury A, Babu H (2008) On the minimization of complete test set of reversible k-cnot circuits for stuck-at fault model. In: Proceedings of 11th international conference on computer and information technology, ICCIT 2008, pp 7–12. doi:10.1109/ICCITECHN.2008.4803009
10. Kole DK, Rahaman H, Das DK, Bhattacharya BB (2010) Synthesis of online testable reversible circuit. In: Proceedings IEEE 13th international symposium on design and diagnostics of electronic circuits and systems (DDECS). IEEE, pp 277–280
11. Kole DK, Rahaman H, Das DK, Bhattacharya BB (2013) Derivation of test set for detecting multiple missing-gate faults in reversible circuits. Comput Electr Eng 39(2):225–236
12. Landauer R (1961) Irreversibility and heat generation in the computing process. IBM J Res Dev 5(3):183–191

13. Lo HK, Popescu S, Spiller T (eds) (2002) Introduction to quantum computation information. World Scientific Publishing Co., Inc., River Edge
14. Ma X, Huang J, Metra C, Lombardi F (2008) Reversible gates and testability of one dimensional arrays of molecular QCA. J Electron Test 24(1-3):297–311
15. Mahammad S, Veezhinathan K (2010) Constructing online testable circuits using reversible logic. IEEE Trans Instrum Meas 59(1):101–109. doi:10.1109/TIM.2009.2022103
16. Maslov D (2015) Reversible logic synthesis benchmarks page. Online: http://webhome.cs.uvic.ca/dmaslov/
17. Maslov D, Dueck G, Miller D (2005) Synthesis of fredkin-toffoli reversible networks. IEEE Trans Very Large Scale Integr VLSI Syst 13(6):765–769. doi:10.1109/TVLSI.2005.844284
18. Mishchenko A, Perkowski M (2002) Logic synthesis of reversible wave cascades. In: Proceedings of international workshop on logic synthesis, pp 197–202
19. Mondal B, Kole D, Das D, Rahaman H (2014) Generator for test set construction of smgf in reversible circuit by boolean difference method. In: Proceedings of IEEE 23rd Asian test symposium (ATS), pp 68–73. doi:10.1109/ATS.2014.24
20. Nayeem N, Rice J (2011) A simple approach for designing online testable reversible circuits. In: Proceedings of IEEE pacific rim conference on communications, computers and signal processing (PacRim). IEEE, pp 85–90
21. Nielsen MA, Chuang IL (2010) Quantum computation and quantum information. Cambridge University Press
22. Perumalla KS (2013) Introduction to reversible computing
23. Polian I, Fiehn T, Becker B, Hayes JP (2005) A family of logical fault models for reversible circuits. In: Proceedings of 14th Asian test symposium. IEEE, pp 422–427
24. Polian I, Hayes JP (2010) Advanced modeling of faults in reversible circuits. In: Proceedings of East-West design & test symposium (EWDTS). IEEE, pp 376–381
25. Rahaman H, Kole DK, Das DK, Bhattacharya BB (2008) On the detection of missing-gate faults in reversible circuits by a universal test set. In: Proceedings of 21st international conference on VLSI design, VLSID. IEEE, pp 163–168
26. Rahaman H, Kole DK, Das DK, Bhattacharya BB (2011) Fault diagnosis in reversible circuits under missing-gate fault model. Comput Electr Eng 37(4):475–485
27. Soeken M, Chattopadhyay A (2015) Fredkin-enabled transformation-based reversible logic synthesis. In: Proceedings of IEEE international symposium on multiple-valued logic (ISMVL), pp 60–65. doi:10.1109/ISMVL.2015.37
28. Taraphdar C, Chattopadhyay T, Roy JN (2010) Mach–zehnder interferometer-based all-optical reversible logic gate. Opt Laser Technol 42(2):249–259
29. Thapliyal H, Vinod AP (2007) Designing efficient online testable reversible adders with new reversible gate. In: Proceedings of IEEE international symposium on circuits and systems, ISCAS. IEEE, pp 1085–1088
30. Vasudevan DP, Lala PK, Di J, Parkerson JP (2006) Reversible-logic design with online testability. IEEE Trans Instrum Meas 55(2):406–414
31. Wille ADVR (2010) Reversible computation. Springer
32. Wille R, Zhang H, Drechsler R (2011) Atpg for reversible circuits using simulation, boolean satisfiability, and pseudo boolean optimization. In: Proceedings of IEEE computer society annual symposium on VLSI (ISVLSI). IEEE, pp 120–125
33. Woeginger GJ (2003) Exact algorithms for NP-hard problems: a survey. Springer
34. Zamani M, Tahoori MB (2011) Online missing/repeated gate faults detection in reversible circuits. In: Proceedings of IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT). IEEE, pp 435–442
35. Zamani M, Tahoori MB, Chakrabarty K (2012) Ping-pong test: Compact test vector generation for reversible circuits. In: Proceedings of IEEE 30th VLSI test symposium (VTS). IEEE, pp 164–169

A. N. Nagamani received M.Tech degree from VTU, Belagavi, INDIA and currently persuing Ph.D from VTU, Belagavi in the area of Reversible circuit optimization and Testing. She is working with PES Institute of Technology as Assistant Professor in Department of Electronics and Communication Engineering. Her area of research includes, Reversible logic optimization and testing, low power Digital VLSI, Analog and mixed signal design, VLSI architecture for optimized performance. She has published several publications in the above mentioned areas in the peer reviewed conferences and Journals.

S. Ashwin is in graduating class of Bachelors degree in Electronics and Communication Engineering from PES Institute of Technology, Bengaluru, India and will be graduating in the year 2015. His research interest includes design, synthesis and testing of reversible logic circuits, cryptography, fault-tolerant computing and VLSI design. He has published 4 research articles in refereed conference proceedings.

B. Abhishek is in graduating class of Bachelors degree in Electronics and Communication Engineering at PES Institute of Technology, Bangalore India. His research interests include Synthesis and optimization of reversible and irreversible circuits using Genetic Algorithms, Reversible logic design, Cryptographic Algorithms, Testing of Reversible circuits and Image processing.

V. K. Agrawal received the M.Tech. degree from IIT, Kanpur, and the Ph. D. degree from the Department of Computer Science, Indian Institute of Science, Bangalore, in 1986. He joined the ISRO Satellite Center in 1978. He has been largely responsible for the development of microprocessor-based on-board computer systems for the satellites developed at ISRO Satellite Center, Bangalore. He has also worked as Group Director for the Control Systems Group at ISRO. His research interests are in the area of Petri nets and evolutionary computing. Presently, he is the Director for CORI (Crusible of Research and Innovation Lab) at PES University, Bangalore, India. He has published several peer reviewed conference and journal articles in his fields of specialization.