Mixing Test Set Generation for Bridging and Stuck-at Faults in Reversible Circuit

Mousum Handique(✉) and Joinal Ahmed

TSSOT, Department of Computer Science and Engineering, Assam University, Silchar 788011, Assam, India
mousum78@yahoo.co.in, joinalahmed@gmail.com

1 Introduction

The reversible circuit technologies are more promising future alternative as compared to conventional circuit technologies in the scenario of high-performance computation. Rolf Landauer [1], 1961, showed that whenever using a logically irreversible operation, it dissipates energy into the environment. The reversible logic operations are those operations which can reuse a fraction of the signal energy that theoretically can approach near to 100%. Therefore, the reversible logic circuit is the most popular technology to achieve this performance. The properties of the reversible circuit are simpler than a conventional circuit. The basic properties of reversible circuits are (a) number of inputs are equal to the number of outputs, (b) only bijection operation is allowed, and (c) no fanout and feedback connection is allowed.

Due to the reversibility property, the efficient test pattern generation of a reversible circuit is relatively simpler than the traditional logic circuit. The other facet of the reversible circuit is that it has produced unique output vector from each corresponding input vector and vice versa, which gives high controllability and observability [2]. A test set is a collection of the input test vectors which are applied to a reversible circuit to detect and observe that the faults are occurred in the circuit. The test set is called as complete test set if it is capable of detecting all the faults in a given circuit. The relation between different fault models of the reversible circuit has been discussed in [3]. This paper has drawn parallels between the bridging and stuck-at faults for generating the test patterns. The test vectors that set the two lines by the opposite logic values "01" and "10" are used to detect the two lines of bridging faults at the same level [4]. In other way, for detecting the stuck-at faults at any level, the lines at every level are set by 0 and 1. Generated complete test set for individual fault model is already in the

© Springer Nature Singapore Pte Ltd. 2018
S. Bhattacharyya et al. (eds.), *Advanced Computational and Communication Paradigms*, Lecture Notes in Electrical Engineering 475,
https://doi.org/10.1007/978-981-10-8240-5_11

literature but "generated test patterns for one particular fault model is derived such that it is capable of detecting another fault model" is our key concern. Based on this concept, we have generated the test patterns for the input bridging faults and further reconstructed the test vectors for the input stuck-at faults which are obtained from bridging fault model.

The rest of the paper is organized as follows: Sect. 2 provides some basic background on reversible logic circuits and an overview of stuck-at and bridging fault model in the reversible circuit. Sect. 3 describes our proposed method for generating the test set for detecting input bridging and stuck-at faults. The experimental result and conclusion are provided by Sect. 4 and Sect. 5, respectively.

2 Background

2.1 Reversible Logic Circuits

A reversible logic circuit is used to implement the reversible computation and it is formalized in terms of gate-level circuits. The reversible circuit structures are cascade structure [5]. All the operation of the reversible circuit has to be performed in a reverse form. The reversible circuit allows only bijective operations [2] and maintained the bijective operation; the circuit does not include any concept of fan-out and feedback connection [6]. In reversible circuit design, several gates have proposed over the past decades. They are the controlled NOT (CNOT) proposed by Feynman [7], Toffoli [8], Fredkin and Toffoli [9], etc. In this paper, we are using only NCT library that contains NOT, CNOT, Toffoli gate, and GT library containing generalized (n-bit) Toffoli gate. This NCT library was introduced by Toffoli [8] in 1980.

2.2 Fault Models in Reversible Circuit

A fault model has described the different levels of abstraction of physical faults in a system. These levels of abstraction can be defined as behavioral, functional, structural, and geometric [4]. A fault model is a mathematical model which represents various faults possibilities and it helps to generate the tests for detecting and reducing all the possible faults in a given circuit [3]. Numerous fault models have been introduced in the reversible circuit. In this work, we have considered stuck-at fault and bridging fault model in the reversible circuit.

1. **Bridging Faults in a Reversible Circuit**: The bridging faults occur when two signals are connected together but they should not be. If two or more lines

involved in bridging faults, then logical effects of this faults are categorized by wired-OR or wired-AND bridging faults [4]. The single-input bridging faults occur, if two input lines are operated by the opposite logic values [10]. Sarkar and Chakrabarti [11] proved that if any test set is complete for the single-input bridging faults, then that particular test is also completed for detecting the multiple-input bridging faults. The authors also showed that the generated test sets are capable of detecting all the possible single- and multiple-input bridging faults if generated test sets are eligible for detecting all the single-input stuck-at faults in a given reversible circuit.

2. **Stuck-at Faults in a Reversible Circuit**: The stuck-at faults occur in a circuit when one of its inputs and outputs to be fixed at either logic value 0 (stuck-at-0) or logic value 1 (stuck-at-1) without considering the input value. This is a very common fault model used for irreversible circuits. If the stuck-at fault is involved only for one line in the circuit, then it is called as the single stuck-at fault (SSF), and if stuck-at fault is involved more than one line, then it is called as multiple stuck-at faults (MSF). Patel et al. [2] stated that any test set is complete for detecting the stuck-at faults, if and only if each line at every level can be set to both 0 and 1 by the test sets. The paper [2] also showed that the test set is complete for all single stuck-at faults, and then it also becomes complete test set for all multiple stuck-at faults.

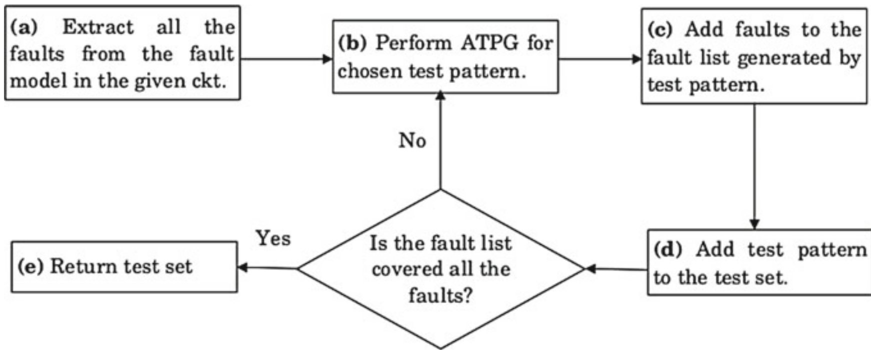

Fig. 1. ATPG and FDL flow

3 Proposed Method

We have divided into two modules in our method. The first module described the test pattern generation of single-input bridging faults. After generating the test patterns, we have derived these patterns for further use of single-input stuck-at faults. Also, we have introduced fault description list (FDL) in our work. The detailed discussion has been given below.

3.1 Fault Description List

Fault description list contains all the possible faults which are generated by the test vectors based on the position of the binary bit sequence present in that test vector. Using the fault description table, we have checked that which test vector is covered by a maximum number of faults and combined the minimum number of test vectors such that it covers all the possible faults in the given reversible circuit.

The construction of ATPG and fault description list (FDL) flow is shown in Fig. 1. In Fig. 1, step (a) extracts all the possible single-input bridging and stuck-at faults in the given reversible circuit. Step (b) generated all the possible test pattern from our proposed method (ATPG). All the faults are stored in the fault list which is generated by test pattern mentioned in step (c). In step (d), each individual test pattern of corresponding faults adds to the test set. Now, we have checked that fault list is covered all the faults, if not then take another combination of test patterns and continue the same process. If some selected test set is able to cover all the faults, then that test set is the final one for detecting all the faults; it has mentioned in step (e) and also this test set is minimized form because the combination of test vectors is growing in increasing order.

3.2 Test Set Generation Algorithm for Single-Input Bridging Fault

In this section, we have arranged our proposed method into two algorithms. Algorithm 1 has explained initially how to extract all the test vectors (TV) based on the bridging faults property. To obtain the binary bit sequences, we are starting from n = 3, where n is the number of input lines of the reversible circuit. In Algorithm 2, the FDL has ensured that which test vector(s) is capable of detecting all single-input bridging faults $(l_1, l_2), (l_1, l_3) \dots (l_i, l_j)$, where $i < j \leq n$

and $(l_i, l_j) \in F_{B_i}$ in the given circuit. Finally, select the minimized fault detecting test set T_B of N number of input lines in a reversible circuit.

Algorithm 1: Extracting the TV from bridging fault property

Input: S is the set of all binary sequences of 3-input lines.
Output: T is minimized form of test vectors (TV).
Step 1: List out all the opposite logic value of two-bit binary sequence for n=3 from set S and P is the set of opposite logic values.
Step 2: Assign 'X' at missing bit position in P and replaced X by 0 and 1.
Step 3: P with duplicate terms then removes until no duplicate terms are exist and then remove complement terms at P.
Step 4: P is the set of final minimized binary sequences and T$\leftarrow$ P.
Step 5: return T.

Algorithm 2: Complete test set generation for detecting single input bridging faults

---

Input: A set of test vectors T from Algorithm 1.
Output: Final minimized fault detecting test set T_B using FDL.
Step 1: Compute T_B, initially empty.
Step 2: All the faults F_{B_1},F_{B_2}, F_{B_3} ...... F_{B_i} are assigned to F_B in a given n-input reversible circuit.
Step 3: Each test vector of T is identified single input bridging faults F_B using the binary bit position.
Step 4: FDL stores the information of each fault identified by TV in T.
Step 5: Choose the increasing order combination of test vectors in T.
Step 6: If T is cover all the faults F_B using FDL then $TS_B \leftarrow T$ and **return $\mathbf{T_B}$**, otherwise continue from Step 5 until covers all the faults.
Step 7: Update the value of n = 4, 5, 6, ...
Step 8: Appending the "0" or "1" at the LSB of each test vector in T_B.
Step 9: T $\leftarrow$ current T_B for n-input lines.
Step 10: Continue the same process from Step 2.

Lemma 1. *Detecting single- and multiple-input bridging faults in a given n-input reversible circuit, the* $(\lceil n/2 \rceil)$ *number of test vectors are sufficient.*

Proof. Here, each of the input lines can be set to both opposite logic values 0 and 1. The test vector TV_i=$(\langle b_{0i}\ b_{1i}\ b_{2i}\ \ldots\ b_{(n-1)i}\rangle)$, where $1 \leq i \leq \lceil n/2 \rceil$, $b_{(n-1)i}$ is the $(n-1)$th bit of ith test vector and $b_{(n-1)i} \in \{0, 1\}$. As explained in Algorithm 1, the test set T=$\{TV_1, TV_2, TV_3\}$ is the test set for n = 3-input lines

(initially) in the given reversible circuit, where no duplicate and complement test vectors are present. The total single-input bridging faults are C(n, 2) = 3 in the three-input reversible circuit. In the binary bit position for three-input lines, any one of the test vectors is capable of detecting the number of faults $C(n, 2)/2$ or more than $C(n, 2)/2$ of the total faults but only one test vector is unable to detect all the faults. Because at least one logic value for each test vector is similar to the other logic value, to maintain the opposite logic value for each test vector, the test set T_B has exactly produced $\lceil n/2 \rceil$ test vectors. Therefore, according to Algorithm 2, we formed the test set $T_B = \{TV_1, TV_2\}$ that is capable of detecting all the faults in the given three-input reversible circuit. If we go for (n + 1), i.e., four-input lines in the reversible circuit, then adding extra input line is created new faults $(l_1, l_4), (l_2, l_4), (l_3, l_4)$. For detecting new faults, we are adding 0 or 1 to the least significant bit (LSB). We have only checked the new form of faults because existing faults are automatically covered by the existing test vector (for n = 3). Hence, $\lceil n/2 \rceil$ test vectors are required to detect all the single-input bridging faults.

3.3 Test Set Generation Algorithm for Single-Input Stuck-at Fault

In this section, we have introduced our algorithm for detecting single-input stuck-at faults. Algorithm 3 extracted the test set T_B which is derived from the single-input bridging fault model and generated the final test set T_S for detecting the single-input stuck-at faults.

Algorithm 3: Test set generation for detecting single input stuck-at faults

Input: T_B is the current test set obtained from Algorithm 2.
Output: Test set T_S for detecting single input stuck-at faults.
Step 1: Compute test set T_S, initially $T_S = T_B$.
Step 2: All faults F_{S_1}, F_{S_2}, $F_{S_3}, \ldots, F_{S_i}$ of the given circuit assign to F_S.
Step 3: FDL stores the information of each fault identified by the T_S.
Step 4: If T_S is covered all the faults of F_S from FDL then **return $\mathbf{T_S}$** else goto next Step.
Step 5: Complement of any one test vector of T_S and update T_S with adding new complement test vector and goto Step 3.

Lemma 2. *Detecting all the single- and multiple-input stuck-at faults in a given n-input reversible circuit, the* $(\lceil n/2 \rceil) + 1$ *number of test vectors are sufficient enough.*

Proof. In Lemma 1, it has proved that test set $T_B = \{TV_1, TV_2\}$ is capable of detecting all the single-input bridging faults in three-input reversible circuit. Any combination of test vectors (excluding complement of test vector), at least one-bit position, is occurred with same logic value. Due to this fact, it is unable to cover either one (or more) stuck-at-0 or stuck-at-1 fault(s). If we are introducing new test vector which is the complement of any test vector (e.g., TV_1 or TV_2),

then that bit position of complement test vectors must occur at least one opposite logic value. Therefore, extra one more test vector is needed for detecting the single-input stuck-at faults in our proposed method, i.e., the test has required exactly $(\lceil n/2 \rceil) + 1$ number of test vectors.

Example 1. Consider the reversible benchmark circuit *ham*3 consisting of three-input and three-output lines as shown in Fig. 2a. The number of single-input bridging faults is $C(3, 2) = 3$. Algorithm 1 generated the minimized test vectors in $T = \{001, 010, 011\}$. Then, using Algorithm 2, the final minimized test set $T_B = \{001, 010\}$ with the help of FDL, which is the complete test set in a *ham*3 benchmark circuit for detecting the input bridging faults. In this circuit, the total numbers of single-input stuck-at faults are $3 \times 2 = 6$. It has observed that the test set T_B is not capable of capturing all the single-input stuck-at faults. Hence, we apply Algorithm 3 and generated the complete test set $T_S = \{001, 010, 110\}$, where "110" is the complement form of "001". The test set T_S covered all the possible single-input faults in *ham*3 benchmark circuit.

Example 2. In this example, we are considering *mpsk*_4_49_13 reversible benchmark circuit consisting of four-input and four-output lines as shown in Fig. 2b. The number of input bridging faults is $C(4, 2) = 6$. The Step 9 in Algorithm 2 generated the minimized test set $T_B = \{0011, 0101\}$ with the help of FDL, which is the complete test set for detecting the single-input bridging faults in a *mpsk*_4_49_13 benchmark circuit. Now if we consider the single-input stuck-at faults of this circuit, then a total number of single-input stuck-at faults occur $4 \times 2 = 8$. Similarly, the test set T_B is not capable to detect all the input stuck-at faults. Therefore, Algorithm 3 is generated the test set $T_S = \{0011, 0101, 1100\}$, where "1100" is the complement form of "0011". The test set T_S is the complete test set for detecting single-input stuck-at faults in *mpsk*_4_49_13 benchmark circuit.

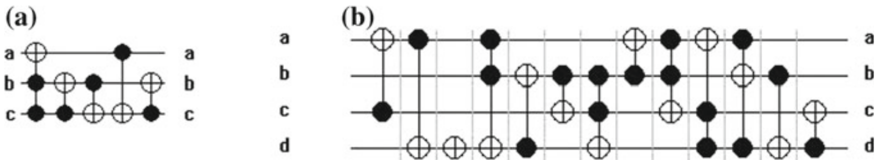

Fig. 2. a *ham*3 benchmark circuit **b** *mpsk*_4_49_13 benchmark circuit

4 Experimental Results

The proposed algorithms are generating the mixing test vectors applied to various reversible benchmark circuits [12] with NCT and GT models. The test sets are generated by our proposed method that is compared with the existing method [11] which is shown in Table 1. It has observed that the proposed method gives good performance in both the fault models and also covered 100% faults.

Table 1. Detection of input faults (BF = input bridging faults and SAF = input stuck-at faults)

Benchmark circuits	Gate model	No. of input/output	No. of input stuck- at faults	No. of input bridging faults	No. of test vectors [11]	No. of test vectors [Proposed]	% fault coverage [Proposed]
					BF+SAF	BF+SAF	BF+SAF
3_17	NCT	3/3	6	8	3	3	100
6sym	NCT	6/1	12	90	6	4	100
9sym	NCT	9/1	18	352	9	6	100
hwb7	GT	7/7	14	152	7	5	100
hwb8	GT	8/8	16	238	8	5	100
hwb6	GT	6/6	12	90	6	4	100
rd73	NCT	7/3	14	152	7	5	100
rd84	NCT	8/4	16	238	8	5	100
ham7	GT	7/7	14	152	7	5	100
ham15	GT	15/15	30	1848	15	9	100
mod1024 Adder	GT	20/20	40	4598	20	11	100
mod10485 76 adder	GT	40/40	80	21222	40	21	100

5 Conclusion

This paper observed that there is a close relation between stuck-at and bridging faults in the reversible circuit. This paper concludes that ($\lceil n/2 \rceil$) test vectors are generated at first for detecting the input bridging faults and reconstructed the test vectors from the bridging fault model such that adding only one test vector is sufficient to detect input stuck-at faults in the n-input reversible circuit. There will be a possibility to design a technique such that similar type of test vectors can be detected which may be some other fault models. This may lead to the future work.

References

1. Landauer R (1961) Irreversibility and heat generation in the computing process. IBM J Res Dev 5(8):183–191
2. Patel KN, Hayes JP, Markov IL (2004) Fault testing for reversible circuits. IEEE Trans Comput Aided Design Integ Circuits Syst 23(8):1220–1230
3. Rice J (2013) An overview of fault models and testing approaches for reversible logic. In: 2013 IEEE pacific rim conference on communications, computers and signal processing (PACRIM). IEEE, pp 125–130
4. Jha NK, Gupta S (2003) Testing of digital systems. Cambridge University Press
5. Maslov D (2003) Reversible logic synthesis. PhD Dissertation, University of New Brunswick
6. Nielson MA, Chuang IL (2000) Quantum computation and quantum information. Monograph Collection (Matt-Pseudo)
7. Feynman RP (1986) Quantum mechanical computers. Found Phys 16(6):507–531
8. Toffoli T (1980) Reversible computing. Springer
9. Fredkin E, Toffoli T (2002) Conservative logic. Springer
10. Rahaman H, Kole DK, Das DK, Bhattacharya BB (2007) Optimum test set for bridging fault detection in reversible circuits. In: 16th Asian test symposium, ATS07. IEEE, pp 125–128
11. Sarkar P, Chakrabarti S (2008) Universal test set for bridging fault detection in reversible circuit. In: 3rd international design and test workshop, IDT 2008. IEEE, pp 51–56
12. Maslov D, Dueck G, Scott N (2005) Reversible logic synthesis benchmarks page