Research Journal of Applied Sciences, Engineering and Technology 4(18): 3470-3475, 2012
ISSN: 2040-7467

Submitted: April 23, 2012 Accepted: May 18, 2012 Published: September 15, 2012

Fault Model for Testable Reversible Toffoli Gates

Yu Pang, Shaoquan Wang, Zhilong He and Qiangbing Zhang
Chongqing University of Posts and Telecommunications, Chongqing, China

Abstract: Techniques of reversible circuits can be used in low-power microchips and quantum communications. Current most works focuses on synthesis of reversible circuits but seldom for fault testing which is sure to be an important step in any robust implementation. In this study, we propose a Universal Toffoli Gate (UTG) with four inputs which can realize all basic Boolean functions. The all single stuck-at faults are analyzed and a test-set with minimum test vectors is given. Using the proposed UTG, it is easy to implement a complex reversible circuit and test all stuck-at faults of the circuit. The experiments show that reversible circuits constructed by the UTGs have less quantum cost and test vectors compared to other works.

Keywords: Quantum cost, reversible logic, stuck-at fault model, test vector, toffoli gate

INTRODUCTION

Recently, reversible circuits started to emerge as an important topic, bringing alternative solutions to classical networks. The motivation behind reversible computation comes from its two important properties: information lossless computation with less energy dissipation and close relation to several emerging technologies such as quantum circuits and optical computing (Landauer, 1961).

The testing properties and test generation for reversible circuits are initially proposed by (Patel *et al*., 2003). The authors shown that only few vectors are necessary to fully test a reversible circuit under the multiple stuck-at fault model with the number growing at most logarithmically both in the number of inputs and the number of gates. The authors (Hayes *et al*., 2004) present a method to make any circuit fully testable for single missing gate fault with single test vector. In particular, to test a circuit with a missing gate fault, it is necessary to apply a vector of the form 111……1 and ensures that any signal change or error at any node will propagate to the circuit's primary outputs. A design-for-testability method has been proposed to make any reversible logic circuit composed of *n*-bit Toffoli gates fully testable for single stuck-at faults and single intra-level bridging faults (Bubna *et al*., 2007). In this case bridging fault between a pair of lines is assumed to occur one at a time and function as a classical AND or OR bridging type. However, the overhead caused by added circuitry for testability is significant, since adding extra control point in Toffoli gate increases the overall quantum cost. Although these off-line testing methods are simple to adopt, the on-line testing of reversible circuits is still a challenging issue.

New online fault testing methods for reversible circuits have been presented. A universal dual rail reversible gate is proposed, which can implement any reversible circuit and provide detection of any single fault on inputs, outputs and inside the gate (Farazmand and Mehdi, 2010). Two reversible gates R1 and R2 are used to construct online-testable reversible circuits. The parity outputs of R1 and R2 are monitored to detect a single bit error (Vasudevan *et al*., 2004a). An extended version, i.e., a new two pair two rail checker is presented (Vasudevan *et al*., 2004b). However the computational complexity due to excessive number of gates limits its application. Recently a concurrent error detection methodology is proposed (Patel *et al*., 2003). The main feature of this method is its capability to detect multi-bit error at the outputs. However, added circuitry for inversing the original gate and constructing a comparator, as well as extra Feynman gates for duplicating primary outputs and inputs increases the area and quantum cost of the overall reversible circuit (Ketan *et al*., 2004).

In this study, we propose a new reversible gate, i.e., a Universal Toffoli Gate (UTG), which can be used to construct any testable reversible circuit. Since a stuck-at fault is a particular fault model used by fault simulators and automatic test pattern generation tools to mimic a manufacturing defect within an integrated circuit, in this study we investigate stuck-at faults of UTG. A small set of test vectors is capable of finding any single fault on inputs, outputs and inside the gate can be found. Further, an algorithm is proposed to find test vectors for a complex circuit constructed by UTGs. Compared to existing methods, the testable reversible circuits generated by UTGs have smaller quantum costs and high testing efficiency.

METHODOLOGY

Proposed UTG:

Definition 1: The network quantum cost is the sum of the quantum cost of all gates. The quantum cost of a gate is

Corresponding Author: Yu Pang, Chongqing University of Posts and Telecommunications, Chongqing, China

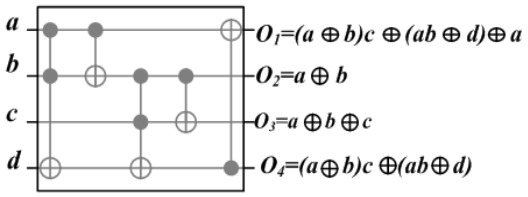

Fig. 1: UTG gate and its function

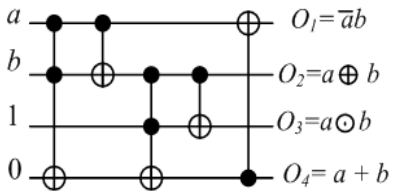

Fig. 2: Basic implemented functions by UTG

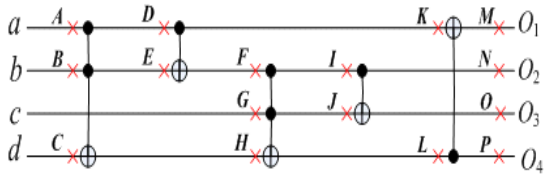

Fig. 3: Single stuck-at faults in a UTG gate

the number of the elementary quantum operations required to realize the function given by the gate.

The quantum cost is an important indicator for evaluating the performance of a reversible circuit. For example, for Toffoli gate family, the quantum cost of NOT and CNOT gates (q_N, q_{CN}) is one, while the quantum cost of the TOFFOLI gate (q_{TO}) is five (Maslov, 2005).

The Universal Toffoli Gate (UTG) is shown in Fig. 1 which consists of two Toffoli gates and three CNOT gates adding up to the quantum cost of 5*2+1*3 = 13. The reversible gate with four inputs and outputs can realize all basic Boolean functions.

During its normal operation the input "*c*" is set to 0 and "*d*" is set to 1. The AND, XOR, EXOR and OR functions can be simultaneously implemented in a UTG, Fig. 2. In consequence, any basic and complex function can be easily realized by cascading UTGs.

According to the lemma (Patel *et al*., 2003), each test vector can exactly cover half of the possible faults and any $2^{n-1}+1$ (n is the input wires) distinct test vectors construct a complete test set, so detecting all stuck-at faults at least requires two vectors and at most $2^{4-1}+1 = 9$ vectors.

Algorithm for test vector:

Generation: According to the internal structure of the UTG gate, we infer the presence of $2l = 32$ single stuck-at faults, Fig. 3. Note, that l is the line number in the UTG. Since reversible circuits have a cascaded structure, stuck-at faults (s-a-v) are analyzed in each layer. Table 1 describes correct outputs (blue color) and faulty outputs for each input vector.

In order to efficiently test UTG for all single stuck-at faults, we need to find the minimum number of input vectors. The necessary conditions for a complete test for single stuck-at faults is the capability of test vectors to assign both "0" and "1" for all wires in each level. Obviously, if the set cannot assign a value "0" in a wire, the stuck-at 1 (s-a-1) fault in this wire is undetectable because the outputs do not change. Hence, as long as all wires can be set to "0" and "1", a test vector will change the faulty point from "0" to "1" or vice versa which must affect output values. Based on this fundamental definition of the test vectors for s-a-v faults, the following proposition can be derived.

Proposition 1: A complete test set must include values of at least one "0" and one "1" for each primary input variable in a UTG.

Proof: If test vectors do not include one "0" or "1" for an input variable, the faults on the lines connected to the primary inputs cannot be detected.

For example, for the UTG gate in Fig. 3 choose (0001, 0111, 0100, 0011) as test vectors. Since the primary variable "*a*" is only set to value "0" in four vectors, the fault *A: s-a-0* cannot be detected because the faulty truth table is the same as the correct one. Therefore, the Proposition 1 can be used to check
whether a test set is possibly complete.

By analyzing Table 1, we find that three vectors (0000, 0101, 1111) labeled by orange color in Table 1 can test all faults. Clearly, the proposed UTG is easy to test as 3/16 = 18.75% input vectors satisfy the testing requirement. Note, that in order for the UTG to generate some specific functions, some inputs have to be pre-set to "0" or "1" (Example 1). Therefore, not all 32 faults can be detected. For example s-a-0 on the input line set to "0" will be undetectable.

Example 1: A UTG with two determined inputs, which are set as "01ab" is illustrated in Fig. 4.

By investigating the UTG internal structure, the following faults: A:s-a-0, B:s-a-1, D:s-a-0, E:s-a-1, F:s-a-1, I:s-a-1, K:s-a-0, N:s-a-1 are undetectable because output values of UTG are still the same whether the above faults exist or not. Since they do not affect the UTG output functions. Table 2 describes two undetectable faults of A:s-a-0 and F:s-a-1. Note, that all untestable fault s are due to the fact that some input lines of the UTG gates are pre-set to "0"or "1" in order for the UTGs to realize required Boolean functions. The test set for the given UTG is (0100, 0101, 0110, 0111). n

Table 1: Truth table of UTG and the all stuck-at faults

Input abcd	Output $O_1O_2O_3O_4$	A: s-a-0 $O_1O_2O_3O_4$	A:s-a-1 $O_1O_2O_3O_4$	B: s-a-0 $O_1O_2O_3O_4$	B- sa1 $O_1O_2O_3O_4$	C:s-a-0 $O_1O_2O_3O_4$	C: s-a-1 $O_1O_2O_3O_4$	D: s-a-0 $O_1O_2O_3O_4$	D: s-a-1 $O_1O_2O_3O_4$
0000	0000	0000	1110	0000	0110	0000	1001	0000	1110
0001	1001	1001	0111	1001	1111	0000	1001	1001	0111
0010	0010	0010	0101	0010	1101	0010	1011	0010	0101
0011	1011	1011	1100	1011	0100	0010	1011	1011	1101
0100	0110	0110	0001	0000	0110	0110	1111	0110	1000
0101	1111	1111	1000	1001	1111	0110	1111	1111	0001
0110	1101	1101	0011	0010	1101	1101	0100	1101	1010
0111	0100	0100	1010	1011	0100	1101	0100	0100	0011
1000	1110	0000	1110	1110	0001	1110	0111	0000	1110
1001	0111	1001	0111	0111	1000	1110	0111	1001	0111
1010	0101	0010	0101	0101	0011	0101	1100	0010	0101
1011	1100	1011	1100	1100	1010	0101	1100	1011	1100
1100	0001	0110	0001	1110	0001	0001	1000	1111	0001
1101	1000	1111	1000	0111	1000	0001	1000	0110	1000
1110	0011	1101	0011	0101	0011	0011	1010	0100	0011
1111	1010	0100	1010	1100	1010	0011	1010	1101	1010
Input abcd	E:s-a-0 $O_1O_2O_3O_4$	E: s-a-1 $O_1O_2O_3O_4$	F: s-a-0 $O_1O_2O_3O_4$	F: s-a-1 $O_1O_2O_3O_4$	G: s-a-0 $O_1O_2O_3O_4$	G: s-a-1 $O_1O_2O_3O_4$	H: s-a-0 $O_1O_2O_3O_4$	H: s-a-1 $O_1O_2O_3O_4$	I: s-a-0 $O_1O_2O_3O_4$
0000	0000	0110	0000	0110	0000	0000	0000	1001	0000
0001	1001	1111	1001	1111	1001	0000	0000	1001	1001
0010	0010	1101	0010	1101	0000	0010	0010	1011	0010
0011	1011	0100	1011	0100	1001	0010	0010	1011	1011
0100	0000	0110	0000	0110	0110	1101	0110	1111	0000
0101	1001	1111	1001	1111	1111	0100	0110	1111	1001
0110	0010	1101	0010	1101	0110	1101	1101	0100	1011
0111	1011	0100	1011	0100	1111	0100	1101	0100	0010
1000	1110	1000	1000	1110	1110	0101	1110	0111	1000
1001	0111	0001	0001	0111	0111	1100	1110	0111	0001
1010	0101	1010	1010	0101	1110	0101	0101	1100	0011
1011	1100	0011	0011	1100	0111	1100	0101	1100	1010
1100	0111	0001	0001	0111	0001	0011	1000	0001	0001
1101	1110	1000	1000	1110	1000	1010	1000	0001	1000
1110	1100	0011	0011	1100	0001	0011	1010	0011	0011
1111	0101	1010	1010	0101	1000	1010	1010	0011	1010
Input abcd	I:s-a-1 $O_1O_2O_3O_4$	J: s-a-0 $O_1O_2O_3O_4$	J: s-a-1 $O_1O_2O_3O_4$	K: s-a-0 $O_1O_2O_3O_4$	K: s-a-1 $O_1O_2O_3O_4O_1$	L: s-a-0 $O_2O_3O_4O_1$	L: s-a-1 $O_2O_3O_4$	M: s-a-0 $O_1O_2O_3O_4$	M: s-a-1 $O_1O_2O_3O_4$
0000	0110	0000	0010	0000	1000	0000	1001	0000	1000
0001	1111	1001	1011	1001	0001	0000	1001	0001	1001
0010	0100	0000	0010	0010	1010	0010	1011	0010	1010
0011	1101	1001	1011	1011	1010	0010	1011	0011	1011
0100	0110	0110	0100	0110	0110	0110	1111	0110	1110
0101	1111	1111	1101	1111	0111	0110	1111	0111	1111
0110	1100	1111	1101	1101	0101	0100	1101	0101	1101
0111	0100	0110	0100	0100	1100	0100	1101	0100	1100
1000	1110	1110	1100	0110	1110	1110	0111	0010	1010
1001	0111	0111	0101	1111	0111	1110	0111	0011	1011
1010	0101	0111	0101	1101	0101	1100	0101	0001	1001
1011	1100	1110	1100	0100	1100	1010	0011	0110	1110
1100	0111	0001	0011	1001	0001	1000	0001	0101	1101
1101	1110	1000	1010	0000	1000	1000	0001	0100	1100
1110	0101	0001	0011	1011	0011	1010	0011	0111	1111
1111	1100	1000	1010	0010	1010	1010	0011	0110	1110
Input abcd	N:s-a-0 $O_1O_2O_3O_4$	N: s-a-1 $O_1O_2O_3O_4$	O: s-a-0 $O_1O_2O_3O_4$	O: s-a-1 $O_1O_2O_3O_4$	P: s-a-0 $O_1O_2O_3O_4$	P: s-a-1 $O_1O_2O_3O_4$			
0000	0000	0100	0000	0010	0000	0001			
0001	1001	1101	1001	1011	1000	1001			
0010	0010	0110	0000	0010	0010	0011			
0011	1011	1111	1001	1011	1010	1011			
0100	0010	0110	0100	0110	0110	0111			
0101	1011	1111	1101	1111	1110	1111			
0110	1001	1101	1101	1111	1100	1101			
0111	0000	0100	0100	0110	0100	0101			
1000	1010	1110	1000	1010	1010	1011			
1001	0011	0111	0001	0011	0010	0011			

Table. 1: Continue

Input abcd	N:s-a-0 $O_1O_2O_3O_4$	N: s-a-1 $O_1O_2O_3O_4$	O: s-a-0 $O_1O_2O_3O_4$	O: s-a-1 $O_1O_2O_3O_4$	P: s-a-0 $O_1O_2O_3O_4$	P: s-a-1 $O_1O_2O_3O_4$
1010	0001	0101	0001	0011	0000	0001
1011	1000	1100	1000	1010	1000	1001
1100	0001	0101	0101	0111	0100	0101
1101	1000	1100	1100	1110	1100	1101
1110	0011	0111	0101	0111	0110	0111
1111	1010	1110	1100	1110	1110	1111

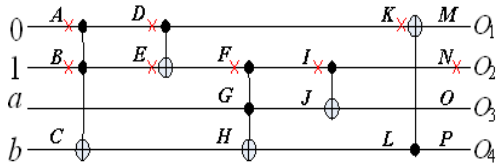

Fig. 4: An example of UTG with determined inputs

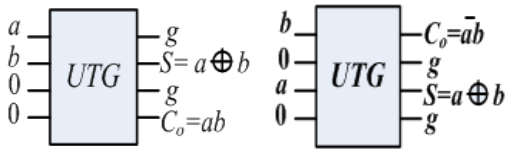

Fig. 5: A half adder implemented by UTG

Example 2: A half adder and half subtractor are the elementary devices in a complex reversible arithmetic circuit. Figure 5 describes how to use a UTG to construct them.

The input "c" and "d" are set to "0" to implement a half adder. O_2 and O_4 represent the sum and carry-out respectively. Since two inputs have fixed values, three faults (C:s-a-0, G:s-a-0 and J:s-a-0) are undetectable. The test vectors (0000, 0100, 1000, 1100) are capable of detecting all the remaining s-a-v faults.

Similar to the half adder, a half subtractor also consists of one UTG with "b" and "d" in UTG mapped to "0". In this configuration only the faults (B:s-a-0, C:s-a-0, E:s-a-0, H:s-a-0) cannot be detected. The test set to detect all the remaining faults is (0000, 0010, 1000, 1010). n

Based on Table 1 describing UTG faults, it is easy to test a complex reversible network like the one in Fig. 6 comprising a number of UTGs. Figure 7 presents the algorithm to find test vectors for a reversible circuit created by cascading UTGs.

The first step analyzes the given reversible circuit to obtain its UTG structure and the number of UTGs. A loop starts to traverse each UTG in Step 2. According to each UTG inputs, Step 3 finds test vectors for this UTG by investigating Table 1, while Step 4 uses found vectors to calculate the UTG outputs, which is helpful for combining the circuit structure to obtain the inputs of next stage UTG (Step 5). While this loop terminates, test vectors for all

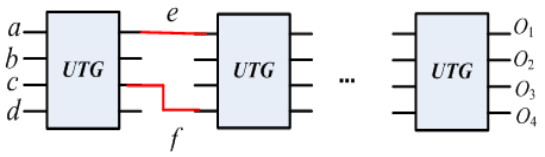

Fig. 6: An example of a complex reversible network

```
Find_Test_Vectors (rev_cir)
1. { (UTG_struc, UTG_num) = Analyze (rev_cir);
2.    for (i=0; i< UTG_num; i++)
3.    { UTG_vec[i] = Find (UTG_in[i]);
4.      UTG_out[i] = Cal (UTG_vec[i]);
5.      UTG_in[i+1]=Map(UTG_struc, UTG_out[i]); }
6.    test_vectors =Retrieve (UTG_struc, UTG_vec[i]);
   return test_vectors; }
```

Fig. 7: An algorithm to find test vectors

UTGs are obtained. The algorithm deletes values of interconnected variables between UTGs (for instance, red color lines in Fig. 7). Further, it only reserves values of primary inputs, so the minimum test vectors of the reversible circuits are found.

EXPERIMENTAL RESULTS

In this section we use the algorithm to find the test vectors for several circuits such as a full adder, full subtractor and a multiplexer on an Intel 2.8 GHz Pentium 4 and 2G bytes of main memory running on Linux.

A full adder: As a basic arithmetic unit, full adders are present in arithmetic datapaths, hence testing of a full adder is very important. Figure 8 describes the use of a UTG to implement a full adder by setting "*d*" to zero.

We use algorithms in (Patel *et al*., 2003; Farazmand and Mehdi, 2010) as comparison. Table 2 gives results for the adder implemented by these algorithms as well as ours. The gate cost and quantum cost of an implemented full adder are described in the column "GC" and "QC" Table 3 and 4.

Table 5 shows the cost comparison of a full adder. Since only one UTG is used, the full adder has 1 reversible gate and quantum cost is 13, while another two adders realized by (Patel *et al*., 2003; Farazmand and

Table 2: Cost comparison of a full adder

Input 01ab	Output $O_1O_2O_3O_4$	A-sa0 O_1O_2, O_3O_4	F-sa1 O_1O_2, O_3O_4
0100	0110	0110	0110
0101	1111	1111	1111
0110	1101	1101	1101
0111	0100	0100	0100

Table 3: Cost comparison of past methods and ours

	GC	QC
Full ubtsractor [5]	8	30
Full subtractor [8]	6	38
Proposed design	2	26
Improvement in %	66.7	13.3

Table 4: Comparison of testable reversible multiplexer

	GC	QC
Multiplexer [5]	8	30
Multiplexer [8]	6	38
Proposed design	2	26
Improvement in %	66.7	13.3

Table 5: Two undetectable faults in the UTG

	GC	QC
Full adder [5]	8	30
Full adder [8]	4	20
Proposed design	1	13
Improvement in %	75	35

Mehdi, 2010) have "30" and "20". Clearly, using UTG to realize a full adder can significantly save quantum cost (around 35 and 60% compared to the other two methods). Only three test vectors are enough to find all faults and there are seven test sets as (0000, 1110, 0100), (0100, 1010, 1100), (0010, 0110, 1100), (0000, 1010, 1100), (0010, 0100, 1110), (1000, 1010, 1110) and (0100, 1010, 1110).

The testing method in (Farazmand and Mehdi, 2010) adopts an additional reversible circuit with inversed function to map the outputs into the inputs, which helps check whether the original reversible circuit has faults. Hence, all test vectors must be used to compare whether the output values are the same as the input ones. As a result, the full adder can be tested by eight vectors. However, our proposed design only needs three vectors, so the improving testing efficiency is distinct.

A full subtractor: Another important arithmetic circuit, i.e., a reversible full subtractor requires two UTGs, Fig. 9. Since the first UTG sets two determined values "0" and the three inputs of the second UTG come from the first one, the number of the total detectable fault is 52 and not 32*2 = 64. The test vectors are (0000, 1000, 0001, 1001).

A multiplexer: Another complex gate implemented using UTG gates is a multiplexer shown in Fig. 10. The faults (*B:s-a-0*, *E:s-a-0*, *J:s-a-1*) in the first UTG and the faults (*G:s-a-0*, *J:s-a-0*) in the second UTG cannot be detected, so the circuit has 59 detectable stuck-at faults and its test vectors are (001000, 001110, 101000).

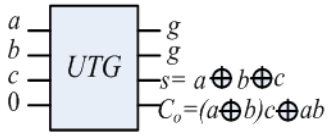

Fig. 8: A full adder implemented by a UTG

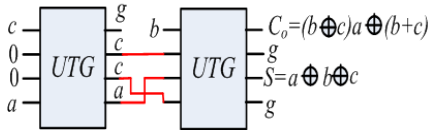

Fig. 9: A full subtractor by UTGs

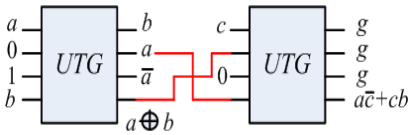

Fig. 10: A multiplexer by UTGs

CONCLUSION

Reversible logic grows in interest due, among others, to its low power characteristic. Most explorations focus on synthesis of reversible circuits but seldom for fault testing. This study presents a universal Toffoli gate, which can realize all basic Boolean functions. The single stuck-at fault model is given and the method to find the minimum number of test vectors is introduced. The analysis shows that only four vectors can test all faults, which proves high testing efficiency of the gate. An algorithm is proposed to find test vectors for a complex reversible circuit constructed by the gate. The experiments compare another two testing methods with ours, which shows that the reversible circuit generated by the gate has less quantum cost and is easy to test.

ACKNOWLEDGMENT

The research reported herein was sponsored largely by the Ministry of Industry and Information Technology of the People's Republic of China under the grant of special projects for internet of things and by the National Natural Science Foundation of China under the grant No. 61102075 and by the Natural Science Foundation of Chongqing under the grant No. CSTC 2011BB2142.

REFERENCES

Bubna, M., N. Goyal and I. Sengupta, 2007. A DFT methodology for detecting bridging faults in reversible logic circuits, Proceeding of IEEE Tencon, pp: 1-4.

Farazmand, N.M.Z. and B.T. Mehdi, 2010. Online Fault Testing of Reversible Logic Using Dual Rail Coding, IEEE 16th International On-Line Testing Symposium, pp: 204-205.

Hayes, J.P., I. Polian and B. Becker, 2004. Testing for missing-gate faults in reversible circuits, Proceeding of. Asian Test Symposium, pp: 100-105.

Ketan, N.P., P.H. John and L.M. Igor, 2004. Fault testing for reversible circuits. IEEE T. Comput. Aid. D. Integr. Circ. Syst., 23(8): 1220-1230.

Landauer, R., 1961. Irreversibility and heat generation in the computing process. IBM J. R.D., 5: 183-191.

Maslov, D., 2005. Reversible Logic Synthesis Benchmarks Page. Retrieved from: http://www.cs.uvic.ca/~dmaslov/.

Patel, K.N., J.P. Hayes and I.L. Matkov, 2003. Fault testing for reversible logic circuits, proceeding. of VTS, pp: 410-416.

Vasudevan, D.P., P.K. Lala and J.P. Parkerson, 2004a. A novel scheme for on-line testable reversible logic circuit design. Proceeding of the 13th Asian Test Symposium, pp: 325-330.

Vasudevan, D.P., P.K. Lala and J.P. Parkerson, 2004b. online testable reversible logic circuit design using NAND blocks. 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp: 324-331.