Testing for Missing-Gate Faults in Reversible Circuits

John P. Hayes[1,2] Ilia Polian[2] Bernd Becker[2]

[1]Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI 48109-2122, USA
jhayes@eecs.umich.edu

[2]Albert-Ludwigs-University
Georges-Köhler-Allee 51
79110 Freiburg i. Br., Germany
{polian|becker}@informatik.uni-freiburg.de

Abstract

Logical reversibility occurs in low-power applications and is an essential feature of quantum circuits. Of special interest are reversible circuits constructed from a class of reversible elements called k-CNOT (controllable NOT) gates. We review the characteristics of k-CNOT circuits and observe that traditional fault models like the stuck-at model may not accurately represent their faulty behavior or test requirements. A new fault model, the missing gate fault (MGF) model, is proposed to better represent the physical failure modes of quantum technologies. It is shown that MGFs are highly testable, and that all MGFs in an N-gate k-CNOT circuit can be detected with from one to $\lceil N/2 \rceil$ test vectors. A design-for-test (DFT) method to make an arbitrary circuit fully testable for MGFs using a single test vector is described. Finally, we present simulation results to determine (near) optimal test sets and DFT configurations for some benchmark circuits.

Keywords: Reversible circuits, quantum circuits, fault models, missing gate faults, design for test.

1 Introduction

Reversible circuits are n-input, n-output circuits in which every input pattern maps to a unique output pattern, thus enabling inputs to be determined from outputs. They are of interest because of their applications in extremely low-power circuit design [1, 2] and quantum computation [1]. In the latter case, reversibility is a necessary requirement of all operations, and non-quantum or "classical" reversible circuits form an important subclass of the quantum circuits. Reversible circuits can be based on many different physical phenomena, which generally fall under the nanotechnology heading. The testing of classical reversible circuits has been considered previously with respect to conventional fault models, in particular, the stuck-at model [3]. Patel et al. [4] have shown that such circuits are generally much easier to test than irreversible ones. For example, very few test vectors are needed to cover all single stuck-at faults, and detection of all single faults guarantees detection of all the corresponding multiple faults. However, as we explain, the stuck-at and other classical fault models are difficult to justify physically in the quantum domain. Here we propose and investigate a very different fault model called the missing-gate (MGF) model, which is better suited to quantum technologies, and may have non-quantum applications as well.

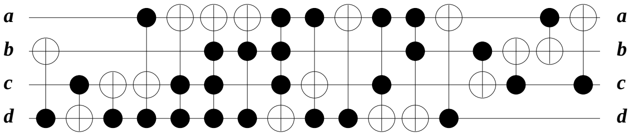

Figure 1: A reversible benchmark circuit `hwb4` composed of 17 k-CNOT gates [5]

As in [4], we only consider circuits composed of reversible elements called k-CNOT gates. Figure 1 shows an example using standard notation. As usual, the input signals are assumed to enter at the left side of the circuit. This figure consists of four horizontal lines denoting signal-carrying wires, and 17 vertical lines denoting gate operations. A general k-CNOT has $k+1$ inputs and outputs, k of which are control nodes $c_1, c_2, \ldots, c_k$ denoted by black dots, while the remaining input-output pair t defines the target node denoted by a ring-sum. With classical binary signal values, a k-CNOT implements the Boolean function

$$c_1, c_2, \ldots, c_k, t \mapsto c_1, c_2, \ldots, c_k, (c_1 \cdot c_2 \cdots c_k) \oplus t \quad (1)$$

implying that the output t is inverted iff all the control inputs are 1. In quantum computations, a k-CNOT can perform the same operation on superimposed quantum states. The leftmost gate in Figure 1 is a 1-CNOT gate, or simply a CNOT, with control d and target b; it realizes the function $d, b \mapsto d, b \oplus d$. The sixth gate from the left is a 3-CNOT with control nodes b, c, d and target a. A 0-CNOT (not shown in the figure) is just an ordinary NOT gate or inverter. In an obvious way, we can associate a stuck-at-0 or a stuck-at-1 fault with each wire segment connected to the input and output sides of any node in a k-CNOT. With these assumptions, such traditional testing tasks as finding a complete, and possibly minimal, test set for stuck-at faults can be addressed.

As mentioned already, a variety of nanotechnologies are being actively investigated for implementing quantum circuits [1]. Several of these use quantum states of (sub)atomic particles—spin-up and spin-down, for instance—to represent information in the form of qubits (quantum bits). These essentially static states are modified by dynamic electromagnetic (EM) pulses that implement gate functions like CNOT. For example, in trapped-ion technology, qubits are individual atoms whose electric charge states are altered by directing laser pulses of precise frequency and duration at them under control of a (classical) computer. In contrast, conventional IC technologies employ static gates and dynamic information-carrying signals. Thus in the quantum case, the "gates" appearing in circuit diagrams like Figure 1 often represent EM pulses, while the "wires" indicate the order in which the gate operations are applied. This casts serious doubt on the applicability of wire-oriented classical faults, like stuck-at or bridging faults [3] to quantum or quantum-like circuits. Furthermore, a qubit's value takes the form conventionally written in the vector notation

$$\psi = a_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad (2)$$

which denotes a superposition of logical 0 and 1. The coefficients a_0 and a_1 in Equation (2) are complex numbers (probability amplitudes), and so are continuous or analog quantities. Quantum gate operations are represented mathematically by unitary matrices.

The question then arises: How should faults in quantum circuits be represented? In view of the large number of physical implementation technologies now under consideration, good fault models should be largely technology-independent and computationally tractable. Previously suggested models include unitary error matrices [6] and various physically motivated digital and analog models [7], most of which are substantially less tractable and scalable than the stuck-at model.

We can analyze quantum faults by making a few, very general physical observations:

- Gate operations are pulse-like, localized and microscopic in scale.
- Errors are caused by faults affecting the length, energy, or direction (spatial alignment) of the pulses.

Dependence on short-range local interactions among qubits tends to limit gate size, so k-CNOTs may be restricted to small values of k, such as k = 0, 1 and 2. Pulse energy is proportional to frequency in the quantum domain, and current flow squared in the classical domain. Experience with a prototype quantum processor based on NMR (nuclear magnetic resonance) technology suggests that the specification and implementation of the gate operations, in this case, hundreds of RF pulses of precise length and frequency, was a major design challenge requiring extensive computer simulation as well as physical trial-and-error [8]. Table 1 lists some fault types suggested by the foregoing observations. Here G_k denotes a k-(qu)bit gate operation, and I_k is the k-(qu)bit identity operation or "no-op". While fault models of this kind are necessarily speculative, they are technology-independent in that they capture common features of known technologies. Moreover, they can readily be augmented with temporal and probabilistic attributes to capture sequential and nondeterministic effects, if desired.

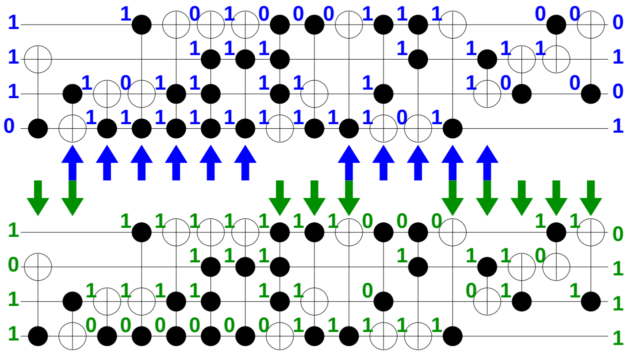

Figure 2: Two tests that detect all MGFs in hwb4

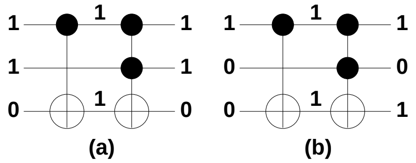

Figure 3: Single vs. multiple MGFs

Of particular interest, is the *missing-gate fault* (MGF) model, defined as the complete removal of a gate operation, or equivalently, replacement of the gate by a set of wires. This implies that a k-(qu)bit gate's matrix G_k is replaced by the corresponding identity matrix I_k. For example, an MGF affecting a CNOT causes the following functional change:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

Note that an n-wire system implies a 2^n-dimensional vector space. Hence, the CNOT matrix in (3) transforms 4-dimensional (column) vectors of the form $[a_{00}\ a_{01}\ a_{10}\ a_{11}]^T$ to $[a_{00}\ a_{01}\ a_{11}\ a_{10}]^T$. We will show in Section 2 that the other fault models defined in Table 1 are covered by the MGF model, so we will focus entirely on the latter.

The remainder of the paper is organized as follows. Section 2 examines the basic testing properties of MGFs in k-CNOT circuits. Bounds on the size of complete MGF test sets are considered in Section 3, while Section 4 describes a low-overhead DFT method to achieve the lower bound of one test vector. Section 5 presents simulation experiments to determine optimal or near-optimal test sets and DFT configurations for benchmark circuits.

2 Testability

We begin by examining the basic testing requirements and properties of MGFs in k-CNOT circuits containing $n > k$ wires and N gates. We assume that at most one gate can be

Fault type	**Abstract model**	**Possible corresponding physical defect**
Missing gate	$G_k \rightarrow I_k$	Short, missing, misaligned or mistuned gate pulses
Repeated gate	$G_k \rightarrow G_k \times G_k$	Long or duplicated gate pulses
Reduced gate	$G_k \rightarrow G_{k-1}$	Partially misaligned or mistuned gate pulses

Table 1: Some possible fault types in quantum-style circuits

faulty at a time, and that faulty signals are detected (measured) only at the circuit's primary (rightmost) outputs. We make no distinction between different types of input/output lines such as non-functional "garbage" lines [1]. Unless otherwise stated, a test for a k-CNOT will be assumed to have the structure $c_1c_2 \ldots c_kt$ with the t-node value on the right.

It follows immediately from the definition (1) of a k-CNOT that input pattern P produces the identical output pattern P in all but the two cases when $c_1c_2 \ldots c_k = 11 \ldots 1$, which perform the non-identity mapping

$$11 \ldots 1t \mapsto 11 \ldots 1t'$$

Hence, to distinguish a k-CNOT from a k-wire identity function requires application of an input pattern of the form 111. . . 10 or 111. . . 11. To test a circuit for MGFs, it is necessary and sufficient to apply one pattern of the form 111. . . $1t$ to every gate. Note that the reversibility property ensures that a signal change (error) occurring at any node must propagate to the circuit's primary outputs. Reversibility also guarantees that all MGFs can be detected, so no redundant faults can occur. The number of tests required to detect all MGFs is obviously between one and N, the number of gates. For example, hwb4 (Figure 1) has two tests {1110, 1011}, which detect all MGFs; see Figure 2. It is easily verified that every gate in this circuit has one of its required MGF tests applied to it.

It is instructive to compare the testing requirements of MGFs with those of other fault models. Stuck-at faults require two patterns to be applied to every gate; it is necessary and sufficient for detection of all such faults that 0 and 1 be applied at least once to every line. For example, a k-CNOT can be tested for all stuck-at faults by applying a complementary pair of tests such as 010. . . 10 and 101. . . 01 to it. However, unlike the MGF test 111. . . $1t$, these stuck-at tests do not exercise the gate's key inversion operation. In fact, 010. . . 10, 101. . . 01 is a "passive" test set, which could equally well serve to test a set of $k + 1$ wires for stuck-at faults.

Consider the other fault models appearing in Table 1. Every k-CNOT operation G_k is its own inverse, i.e., $G_k \times G_k = I_k$, so the missing and repeated gate faults are essentially equivalent. In the case of the reduced gate fault $G_k \rightarrow G_{k-1}$, one c-node, say the topmost one, changes to a wire, so that (1) becomes

$$c_1, c_2, \ldots, c_k, t \mapsto c_1, c_2, \ldots, c_k, (c_2 \cdot c_3 \cdots c_k) \oplus t$$

Although an MGF test for the original gate G_k is not always a test for the reduced gate G_{k-1}, it is worth noting that G_{k-1} dominates G_k in the usual testing sense [3]. Hence for many test generation purposes, the MGF model covers the other models of Table 1, as asserted earlier.

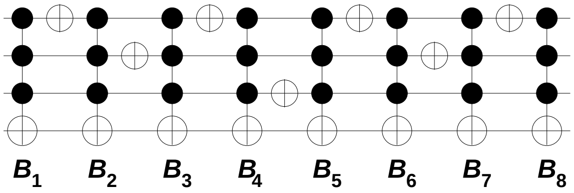

Figure 4: Circuit requiring a linear number of tests

Dominance with respect to MGFs can be helpful during test generation. Another useful property is independence, where two gates are said to be independent if they cannot be tested simultaneously. To illustrate, consider again the circuit hwb4 in Figures 1 and 2 with the gates labeled 1 through 17 from left to right. Gate 6 is clearly dominated by its neighbors, gates 5 and 7, since the c-nodes of gate 6 contain those of the other two gates. On the other hand, gates 6 and 9 are independent because an MGF test for both would force gate 8, which lies between them, to invert the value of wire *d* (lowermost in the figure), producing complementary values on the *d* control nodes of gates 6 and 9. Hence these gates have no test in common. Gates 6 and 9 are independent implying that at least two tests are required by hwb4. Consequently, the test set {1110, 1011} shown in Figure 2 is optimal in size.

In [4], it is proven that a test set that detects all single stuck-at faults in a k-CNOT circuit also detects all multiple stuck-at faults. The example in Figure 3 demonstrates that this property does not hold for MGFs. The test vector 110 (Figure 3(a)) detects the two possible single MGFs in this circuit, but if both gates are missing, the values at the circuit's outputs are identical to the fault-free case. On the other hand, the vector 100 (Figure 3b)) detects the multiple MGF (both gates missing), but only one of the single MGFs.

3 Test Set Size

An arbitrary test set consisting of at least $2^{n-1} + 1$ vectors is shown in [4] to detect all the stuck-at faults in a k-CNOT circuit with n wires. Using similar reasoning, any test set of size $2^n - 2^{n-k_{\max}} + 1$ can be proven to detect all the MGF faults, where $k_{\max}$ is the maximum number of control nodes of any CNOT gate in the circuit. Consider, for instance, a k-CNOT gate, with $k \leq k_{\max}$. The MGF for this gate is detected iff at least one of its vectors applies 1 to all k of its control inputs. Due to reversibility, different vectors at the circuit's inputs will result in different values on the n wires preceding the gate under consideration. Consequently, 2^{n-k} out of 2^n possible vectors detect the fault. At least one of these 2^{n-k} vectors must be contained in a set of cardinality $2^n - 2^{n-k_{\max}} + 1 > 2^n - 2^{n-k_m}$.

Patel et al. [4] also give two upper bounds for the size of a stuck-at test set. In the MGF case, we have the trivial upper bound of N test vectors for an N-gate circuit. For two consecutive gates in the circuit there is always a test vector that detects the MGFs in both these gates, namely, the vector that justifies 1 values on all n wires between the gates, ensuring that 1 is applied to all control inputs of both gates. Taking this into consideration, the upper bound on test set size becomes $\lceil N/2 \rceil$.

Many reversible benchmark circuits, including an adder design of an arbitrary width [9], turn out to be testable by just two vectors. This suggests the question of whether there are test sets of constant or logarithmic size for an arbitrary circuit. Next we construct a circuit that requires a number of tests that is linear in its number of gates, suggesting that the above upper bound is tight.

Let l be an arbitrary number greater than 1. The circuit in question has $n = \lceil \log_2 l \rceil + 1$ wires and l $(n-1)$-CNOT gates B_1 through B_l, with the target nodes on the lowermost wire, and $l-1$ inverters (0-CNOT gates) placed between the gates B_i. Figure 4 shows the circuit for $l = 8$. In general, an inverter is placed on the first (uppermost) wire after every second B_i starting with B_1; it is placed on wire 2 after every fourth B_i, starting with B_2; on wire 3 after every eighth B_i, starting with B_4 and, in general, on wire j on every 2^jth B_i starting with $B_{2^{j-1}}$.

The circuit constructed in this way requires l tests, because all B_i's are *pairwise independent*, meaning that no test vector can detect the MGFs for both B_i and B_j simultaneously, $i \neq j$. This is because to detect the MGF wrt B_i (B_j), the uppermost $n-1$ wires preceding B_i (B_j) must assume the value 1. However, it can be seen from the construction that the number of inverters between any two of B_i's is odd for at least one wire. Consequently, the all-1 vector cannot be justified for more than one gate simultaneously. Overall, the circuit has $N = 2l - 1$ gates and requires $l = \lceil N/2 \rceil$ tests, which is exactly the upper bound derived above. So, this upper bound is sharp for general reversible circuits.

While the lower bound for the number of tests required to detect all stuck-at faults in a k-CNOT circuit is two, there are circuits that can be tested for all MGFs with only one test vector. In the next section, we show a way to transform an arbitrary circuit into such a circuit.

4 Design for Testability

Next we demonstrate how, by adding one wire and several 1-CNOT gates, an arbitrary circuit can be made testable for all MGFs with a single vector. The method starts with any vector applied to the original circuit, and systematically constructs DFT logic which enables this test vector to detect all MGFs. By conditionally inverting the values at gates that correspond to undetected faults, all detection conditions can be met simultaneously.

Figure 5 shows the circuit `hwb4` made testable for the pattern 1011. The extra wire is denoted DFT. When 0 is applied to the DFT input, the circuit's function is unchanged (normal operation mode). When 1 is applied to DFT, some of the circuit's signals are inverted such that all the control inputs of all the gates are set to 1 (test mode). Consider the third gate from the left in the original circuit. Simulation of the vector 1011 (which is shown in the lower part of Figure 2) results in a 0 being applied to its control input, thus violating the detection condition for the corresponding MGF. Inserting a 1-CNOT gate inverts this value (while in test mode) and ensures the testability of the MGF corresponding to this gate; the same holds for four subsequent gates. However, doing so leads to a violation of the detection condition for the 3-CNOT gate that is eighth from the left in the original circuit, requiring insertion of a 1-CNOT gate before that gate. This process continues until the outputs of the circuit are reached.

For an arbitrary circuit, an arbitrary vector is simulated and a 1-CNOT gate is added before the first control node from the left with a 0 value. Then the simulation is continued starting from the modified location; the leftmost control input with 0 value is identified, and another 1-CNOT gate is inserted. This is iterated until all the control nodes of all the gates have 1 values applied to them. Note that inserting the 1-CNOT gates may have impact on other gates in the circuit as well; simulating `hwb4` once and inserting the 1-CNOT gates at all control inputs with logic-0 values would not result in a correct solution. The values that differ from those in the circuit without DFT logic are encircled in Figure 5. Note that the DFT gates themselves have 1's on their control inputs, hence all MGFs associated by them are detected.

While the foregoing method works with any test vector, it can be used to obtain a (unique) test vector that requires the minimum number of extra 1-CNOT gates, and so is optimal with respect to DFT. This vector can be determined as follows. The value applied to a wire is 1 when the number of target inputs to the left of the leftmost control input on that wire is zero or odd, and is 1 otherwise. Note that since all the control inputs assume the 1 value, all the target nodes invert the signals applied to them. For instance, 1011 is the optimal vector for `hwb4`. If the vector were, say, 1010, then a 1-CNOT gate would be required to ensure the detection of the MGF corresponding to the leftmost gate. The solution for the rest of the circuit would remain the same, so the insertion of the additional gate would have no benefit. The same argument holds for all other wires, so 1011 (or, for an arbitrary circuit, the vector calculated as outlined above) is optimal.

5 Experimental Results

We have implemented fault simulation, automatic test pattern generation (ATPG) and DFT synthesis for missing gate faults. Table 2 summarizes the results obtained for the

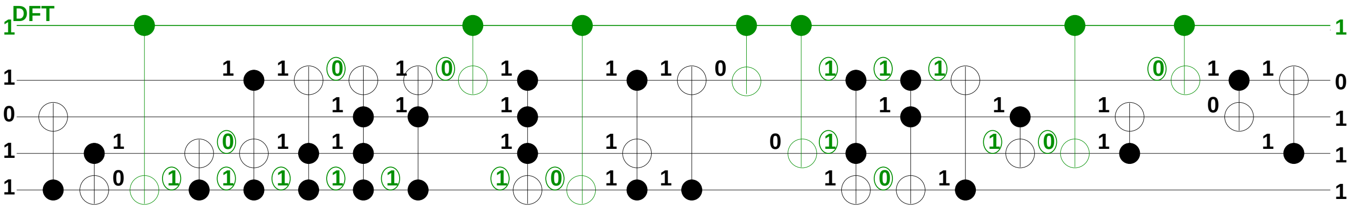

Figure 5: DFT logic for `hwb4` and vector 1011

benchmark circuits given in [5], and a few reversible adders from [9]. The first three columns of the table give the circuit's name, number of gates N (which under the MGF model is also the number of faults) and number of wires n.

We applied 1,000 random patterns to each circuit. 100% of faults are detected for all circuits (recall that MGFs cannot be redundant). The column "Vecs" contains the index of the last vector that detect a fault, and in the column "Req", the number of vectors that actually detect new faults is given. For instance, if the first vector detects some faults, the second vector detects no faults not detected by the first vector, and the third vector detects all the remaining faults, then the value of "Vecs" would be 3 and the value of "Req" would be 2.

Generally speaking, the gap between these two numbers is quite large for most circuits. This suggest that, similarly to classical circuits and stuck-at faults, some test patterns are effective in detecting large numbers of faults, and others are not. Consequently, deterministic ATPG can be expected to reduce test set size significantly. Given that the circuits are rather small, we were able to deploy ATPG algorithms that are more powerful than the standard methods for stuck-at faults that target one fault at a time.

The first approach we used is a greedy heuristic. It fault-simulates all 2^n possible test vectors and picks one that detects most faults. Then it determines for the remaining vectors how many yet-undetected faults they detect, and the best vector is selected. This process is iterated until no undetected faults are left. The second approach is an exact branch-and-bound algorithm. The sizes of test sets calculated by the greedy and the branch-and-bound algorithms are given in the columns "Gr." and "B&B", respectively. While the run times of the greedy algorithm were reasonable for all the tested circuits, this was not the case for the exact algorithm. For a few circuits, we could only prove that there is no complete test set consisting of V or fewer vectors; the highest known value of V is quoted in the table.

It can be seen that deterministic ATPG indeed leads to much more compact test sets. On the other hand, the quality of the results from the greedy algorithm is quite high: for most circuits, it determined optimal solutions. Overall, most benchmark circuits can be tested by a few vectors if deterministic ATPG is performed. Applying random vectors also results in detection of all faults, but the number of needed tests is much higher.

We also implemented the DFT synthesis procedure from Section 4 that makes a circuit testable by one vector. The number of additional 1-CNOT gates needed when the optimal vector is used is given in the penultimate column "Gates" of Table 2. (The algorithm to compute the optimal vector was introduced in Section 4). The last column of the table marked "%QC", is the overhead imposed by the DFT logic in terms of the *quantum cost* metric defined for k-CNOTs in [10, 11]. The quantum cost of a 0-CNOT (an inverter) and a 1-CNOT is 1; it is 5 for a 2-CNOT; 13 for a 3-CNOT; 29 for a 4-CNOT; 61 for a 5-CNOT; 125 for a 6-CNOT; and 253 for a 7-CNOT, which is the largest gate that shows up in the benchmark circuits [5].

The fact that 1-CNOT gates are relatively inexpensive is advantageous for our DFT design. For instance, the sheer number of gates required for the circuit `hwb7tc` is quite high (more than 60% of gates in the circuit), but the actual overhead is only 3.34%. Overall, the overhead ranges between 0% (for those circuits that can be tested by one test vector without modification) and 16.67%. Such an overhead appears to us to be reasonable. The results for the adders from [9] are shown in Table 3. As in [9], `add`i stands for an adder that adds two $(i+1)$-bit numbers. It has $i+2$ auxiliary wires that are necessary in order to make the circuit reversible, making a total of $3i+4$ wires. It can be shown that these adders can be completely tested by two vectors for an arbitrary value of i. Hence, we omit deterministic ATPG results for the adder case. Similarly to the other benchmark circuits, random pattern simulation fails to produce test sets of size close to the optimum (which is 2). The DFT overhead is around 8% for the adders.

6 Conclusions

We have examined the testing requirements of reversible circuits composed of k-CNOT gates with respect to a new fault model, the MGF model. MGFs are motivated by certain technologies used in quantum computing. We have shown that MGFs are highly testable with relatively few test vectors. The number of tests can be reduced to one with a low-cost DFT method. In the context of reversible circuits, MGFs are comparable to stuck-at faults in computational complexity. MGFs appear to be less useful for testing irreversible circuits where a gate may have fewer outputs than inputs, and so cannot be replaced by a set of wires. However, MGFs may have applications to design verification of both reversible and irreversible circuits, since omitting a needed gate is a not infrequent design error [12].

Circuit	N	n	Random		ATPG		DFT cost	
			Vecs	Req	Gr.	B&B	Gates	%QC
2of5d1	30	6	30	8	4	4	10	6.33
2of5d2	26	7	12	6	2	2	5	12.50
3_17tc	12	3	6	3	2	2	1	7.14
4_49tc1	24	4	3	3	3	3	7	11.67
5mod5tc	29	6	55	6	1	1	0	0.00
6symd2	40	10	16	5	2	2	10	13.89
9symd2	52	12	18	8	3	3	18	16.67
ham15tc1	162	15	188	17	7	> 2	47	1.84
ham3tc	11	3	2	2	2	2	1	11.11
ham7tc	38	7	12	6	4	4	5	5.95
hwb4tc	25	4	6	4	2	2	7	10.77
hwb5tc	66	5	23	12	5	5	38	10.80
hwb6tc	138	6	59	19	9	8	83	5.42
hwb7tc	305	7	115	32	15	> 4	190	3.34
mod5adders	33	6	18	5	3	3	8	6.40
mod5d1	18	5	5	3	1	1	0	0.00
mod5d2	19	5	4	3	1	1	0	0.00
rd32	12	4	2	2	2	2	1	8.33
rd53d1	26	7	9	5	2	2	3	2.14
rd53d2	28	8	8	6	2	2	5	11.36
rd53rcmg	44	7	83	8	4	3	19	7.60
rd73d2	40	10	16	5	3	3	11	14.47
rd84d1	58	15	8	5	3	3	14	12.50
xor5d1	14	5	5	3	1	1	0	0.00
add1	28	7	13	5	2	2	3	7.89
add2	42	10	7	5	2	2	5	8.06
add3	56	13	12	6	2	2	7	8.14
add4	70	16	7	7	2	2	9	8.18
add5	84	19	8	6	2	2	11	8.21

Table 2: Results for fault simulation, ATPG and DFT

Acknowledgment

John P. Hayes contributed to this work while visiting the University of Freiburg under an award from the Alexander von Humboldt Foundation.

7 References

[1] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[2] C. Bennett. Logical reversibility of computation. *IBM J. Res. and Develop.*, 17:525–532, 1973.

[3] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[4] K.N. Patel, J.P. Hayes, and I.L. Markov. Fault testing for reversible circuits. In *VLSI Test Symp.*, pages 410–416, 2003.

[5] D. Maslov, G. Dueck, and N. Scott. *Reversible Logic Synthesis Benchmarks Page*. http://www.cs.uvic.ca/˜dmaslov/, 2004.

[6] E. Knill, R. Laflamme, A. Ashikhmin, H. Barnum, L. Viola, and W. H. Zurek. Introduction to quantum error correction. *Los Alamos Science*, 27:188–221, 2002.

Circuit	N	n	Random		DFT cost	
			Vecs	Req	Gates	%QC
add1	28	7	13	5	3	7.89
add2	42	10	7	5	5	8.06
add3	56	13	12	6	7	8.14
add4	70	16	7	7	9	8.18
add5	84	19	8	6	11	8.21
add6	98	22	13	7	13	8.23
add7	112	25	11	7	15	8.24
add8	126	28	9	8	17	8.25
add9	140	31	8	7	19	8.26
add10	154	34	7	7	21	8.27
add11	168	37	19	9	23	8.27
add12	182	40	18	8	25	8.28
add13	196	43	16	10	27	8.28
add14	210	46	8	8	29	8.29
add15	224	49	17	11	31	8.29
add16	238	52	17	10	33	8.29
add17	252	55	16	10	35	8.29
add18	266	58	14	9	37	8.30
add19	280	61	12	12	39	8.30
add20	294	64	28	12	41	8.30
add21	308	67	12	12	43	8.30
add22	322	70	16	11	45	8.30
add23	336	73	14	12	47	8.30
add24	350	76	14	13	49	8.31
add25	364	79	20	14	51	8.31
add26	378	82	20	12	53	8.31
add27	392	85	17	12	55	8.31
add28	406	88	26	14	57	8.31
add29	420	91	14	12	59	8.31
add30	434	94	11	11	61	8.31
add31	448	97	13	12	63	8.31
add32	462	100	25	16	65	8.31
add48	686	148	19	15	97	8.32
add64	910	196	13	13	129	8.32

Table 3: Results for fault simulation and DFT, adder designs from [9]

[7] K.M. Obenland and A.M. Despain. Impact of errors on a quantum computer architecture. Technical report, Univ. of Southern California, 1996.

[8] L. Steffen, L.M.K. Vandersypen, and I.L. Chuang. Toward quantum computation: a five-qubit quantum processor. *IEEE Micro*, 21(2):24–34, 3 2001.

[9] V. Vedral, A. Barenco, and A. Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(147), 1996.

[10] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A.1 Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 11 1995.

[11] D. Maslov and G. Dueck. Improved quantum cost for n-bit Toffoli gates. *Electronic Letters*, 39(25):1790–1791, 12 2003.

[12] H. Al-Asaad and J.P. Hayes. Logic design validation via simulation and automatic test pattern generation. *Jour. of Electronic Testing: Theory and Applications*, 16:575–589, 2000.