DETERMINISTIC AND PROBABILISTIC TEST GENERATION FOR BINARY AND TERNARY QUANTUM CIRCUITS

Sowmya Aligala, Sreecharani Ratakonda, Kiran Narayan, Kanagalakshmi Nagarajan, Martin Lukac, Jacob Biamonte and Marek Perkowski

Portland Quantum Logic Group

Electrical & Computer Engineering Department, Portland State University, Portland, OR 97207-0751, USA, mperkows@ee.pdx.edu

Abstract

It is believed that quantum computing will begin to have an impact around year 2010. Much work is done on physical realization and synthesis of quantum circuits, but nothing so far on the problem of generating tests and localization of faults for such circuits. Even fault models for quantum circuits have been not formulated yet. We propose an approach to test generation for a wide category of fault models of single and multiple faults. It uses deterministic and probabilistic tests to detect faults. A Fault Table is created that includes probabilistic information. If possible, deterministic tests are first selected, while covering faults with tests, in order to shorten the total length of the test sequence. The method is applicable to both binary and ternary quantum circuits. The system generates test sequences and adaptive trees for fault localization for small binary and ternary quantum circuits.

1. Introduction

Quantum computers will reach a lower limit as to how much heat the computing device can generate. They will be super-fast and super-small in size. Quantum logic gate [1,3,4,5,6,7,9,10,11,13,17,18] is a device which performs an operation described by a unitary matrix on selected qubits (quantum bits, [11]). Binary quantum gates process qubits which can have either a |0> or |1> or both |0> and |1> at the same time to varying extents and hence exhibit a superposition state $\alpha|0>+|\beta>$ where $|\alpha|^2 + |\beta|^2 = 1$, α and β are complex numbers such that measurement probability of |0> is $|\alpha|^2$ and measurement probability of |1> is $|\beta|^2$. $|X|^2$ is a result of multiplication of complex number X and its conjugate. When the qubit state is observed or measured, it becomes invariably either |0> or |1>. Ternary quantum gates [2,3,7,10] process qutrits which can be pure state |0>, |1> or |2> or any combination of |0>, [1> and |2>, a superposition state $\alpha|0> + |\beta> + |\gamma>$ where $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$, α, β and γ are complex numbers such that measurement probability of |0> is $|\alpha|^2$, measurement probability of |1> is $|\beta|^2$ and measurement probability of |2> is $|\gamma|^2$, [10,11]. When the qubit state is observed or measured, it becomes either |0>, |1> or |2>. Quantum gates and circuits exhibit the additional property of reversibility as their mechanism of action is through Schrödinger's evolution (which is reversible by virtue of being unitary). Thus, methods developed for permutative (reversible) circuits [12,15] are helpful for quantum circuits as well. Matrices of all quantum operations are unitary (and usually have complex numbers as entries). Matrix X is unitary when $X * X^+ = I$, where I is an identity matrix and X^+ is a hermitian matrix of X. Hermitian matrix of X is conjugate transpose matrix of X. Permutative circuits have only pure states and their matrices are permutative [11].

The Test Generation Problem is to find a sequence of input vectors (tests) that applied to the circuit will either confirm that the circuit is correct or will determine that it has one or more faults from certain library of faults. When only a single fault in a circuit is assumed, we use a single-fault model [8]. Here we are concerned with a multiple-fault model, where there may be more than one fault in the circuit. The reversibility property notably

simplifies the problem of testing reversible circuits compared to irreversible circuits [12]. The basic quantum gates that are used in quantum circuits in this paper are Toffoli, Feynman, CV (controlled square root of NOT) and CV^+ (controlled square root of NOT Hermitian gate). These gates are selected for explanation only, since they are truly quantum and allow to create all permutative binary quantum gates. However, the test generation and fault localization methods [8,15] outlined here are for arbitrary (binary or ternary) quantum gates and for broad fault models (including stuck-at pure states, stuck-at superposition states, bridging-AND, bridging-OR, shift of value, phase shift, gate change and many others).

The paper is organized as follows. Section 2 presents the quantum gates that will be used in circuits for which we will generate tests. Section 3 discusses fault models. Section 4 presents testing of reversible and quantum circuits. Section 5 discusses generalization to ternary reversible and quantum circuits. Section 6 discusses generating complete test sets and localizing faults and section 7 concludes the paper. Although we tried to make the paper self-contained, the reader interested in more details may need to consult basic textbooks about test generation and fault localization (at least [8]) and quantum textbooks like [11], as well as paper [15].

2. Quantum Gates and Their Unitary Matrices.

Toffoli Gate.

Toffoli gate (C^2NOT) [11,16] is a 3-qubit universal permutative gate that has the most central role in quantum computing. As the alternative name implies, this gate only negates the third qubit if the first two qubits are in the |1> state. One use of this gate is as a reversible AND gate: when the third qubit is set to |0>, it is only flipped to |1> if both the first and the second inputs are true. Likewise, by setting the third qubit to |1>, we get NOT-AND (NAND) functionality (Figure 1a).

The unitary operation matrix of this gate is shown in Figure 2a. As we see, it is a permutation matrix; in every row and column there is only one "1" and all other entries are "0", meaning a permutation of states.

Feynman Gate.

Feynman gate (CNOT) gate takes two qubits as input, |x> and |y>. The result is the |x> qubit and an XOR of |x> and |y>. If the value of |x> is 0, the value of |y> remains the same, else the value of |y> is flipped to it's opposite (see Figure 1b). The unitary operation matrix of Feynman is also permutative (Figure 2b).

Square Root of NOT Gate

Connecting two "square root of NOT" gates in series acts as a NOT gate inverting a qubit (that is, the probabilities that the qubit will collapse to pure state |1> is changed to the probability that that the qubit will collapse to |0>, [1]). If one measures the qubit after only one "square root of NOT" gate, the result of the measurement is unknown.

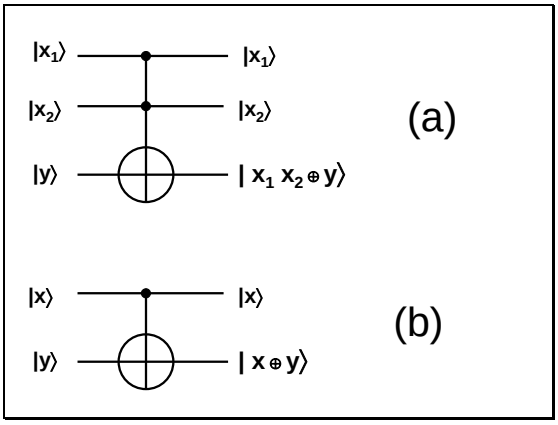


Figure 1 (a) Toffoli gate, (b) Feynman gate

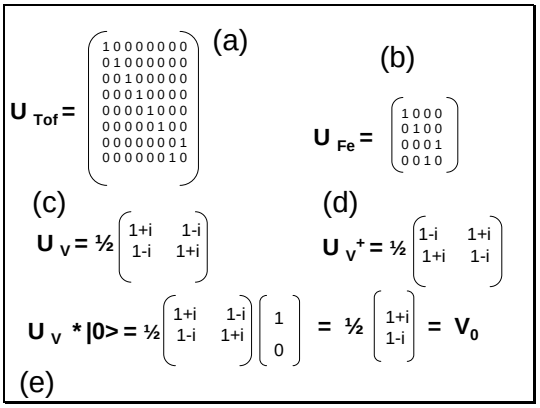


Figure 2. (a) Unitary matrix of Toffoli gate, (b) unitary matrix of Feynman gate, (c) unitary matrix of Square-root-of-Not gate, (d) unitary matrix of Square-root-of-Not-Hermitian gate, (e) calculation of result on output of Square-root-of-Not gate when its input is in pure state |0>

There is however an equal probability of measuring a |0> and a |1>. By V we denote the "square root of NOT" gate. The unitary operation matrix U_V of V

gate is shown in Figure 2c. As we see, this is not a permutative gate, but a truly quantum gate. It means, applied to pure states it creates superposition states on its output (Figure 2e).

Square Root of Not Hermitian Gate

The conjugated transpose of a unitary matrix X is called the hermitian of matrix X and denoted by X^+. By V^+ we denote a gate that has a unitary matrix which is a hermitian of V. Therefore, the hermitian of V is called "square root of NOT hermitian" and has the unitary matrix U_{V+} of gate V^+ from Figure 2d.

Operation of V and V^+ Gates.

Design of many permutative gates is based on (controlled) cascading of V and V^+ gates. Cascading two square root of NOT gates acts as a basic inverted gate (see Figures 3 and 4a).

$$\begin{pmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{pmatrix}\begin{pmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{pmatrix} =$$

$$= 0.5\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} 0.5\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} =$$

$$= 0.25\begin{pmatrix} (1+i)(1+i)+(1-i)(1-i) & (1+i)(1-i)+(1-i)(1+i) \\ (1-i)(1+i)+(1+i)(1-i) & (1-i)(1-i)+(1+i)(1+i) \end{pmatrix} =$$

$$= 0.25\begin{pmatrix} (1+2i+i^2)+(1-2i+i^2) & 2(1-i^2) \\ 2(1-i^2) & (1+2i+i^2)+(1-2i+i^2) \end{pmatrix} =$$

$$= 0.25\begin{pmatrix} 1+i^2+1+i^2 & 2-2i^2 \\ 2-2i^2 & 1+i^2+1+i^2 \end{pmatrix} = 0.5\begin{pmatrix} 1+i^2 & 1-i^2 \\ 1-i^2 & 1+i^2 \end{pmatrix} =$$

$$= 0.5\begin{pmatrix} 1+i^2 & 1-i^2 \\ 1-i^2 & 1+i^2 \end{pmatrix} = 0.5\begin{pmatrix} 1+(-1) & 1-(-1) \\ 1-(-1) & 1+(-1) \end{pmatrix} = 0.5\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Figure 3 (from [1]). Step-by-step calculation of unitary matrix of an inverter gate created by multiplication of unitary matrices of Square-root-of-Not gates connected in series (standard matrix multiplication).

The operation of the circuit from Figure 4a can be explained by the matrix equations from Figure 3. Multiplying the unitary matrix U_V by the input state we obtain the vector ½ $[1+i \;\; 1-i]^T$ = V_0, Figure 2e. By multiplying V by this vector we obtain vector $[0\;1]^T$ = |1>.

Let us now try to find, by matrix/vector multiplication, all possible states that can be created by applying all possible serial combinations of gates V and V^+ to states |0> , |1> and all states created from these pure states (Figure 5). A qubit |0> given to a "square root of NOT" gate (Figures 2e and 5a) gives a state denoted by $|V_0\rangle$. After measurement this state gives |0> and |1> with equal probabilities ½. Similarly all other possible cases are calculated in Figure 5b – h.

As we see, after obtaining states |0>, |1> $|V_0\rangle$ and $|V_1\rangle$ the system is closed and no more states are generated. Therefore the subset of (complex, continuous) quantum space of states is restricted with these gates to a set of states that can be described by a four-valued algebra with states { |0>, |1>, $|V_0\rangle$, $|V_1\rangle$ }. We assume here for simplification of explanation that only faults s-a-

0, s-a-1, s-a- V_0, and s-a-V_1 are possible, but many other fault models can be defined.

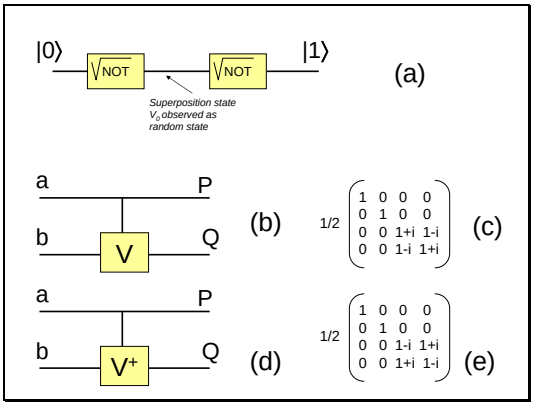


Figure 4. (a) Cascading V gates creates an inverter. Measurement of intermediate state would give |0> and |1> with equal probabilities, (b) Controlled-V gate, (c) its unitary matrix, (d) Controlled-V+ gate, (e) its unitary matrix.

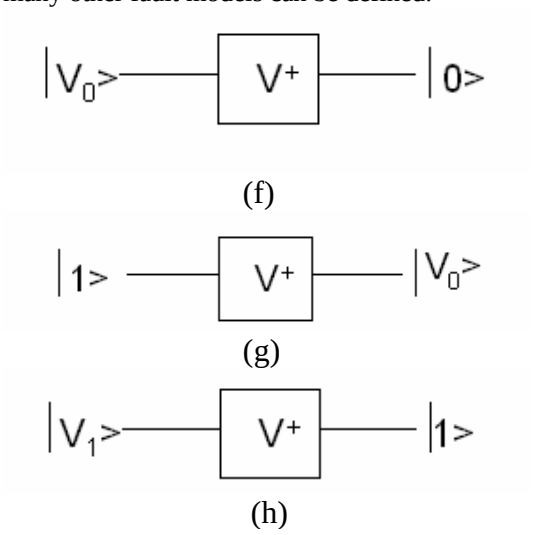
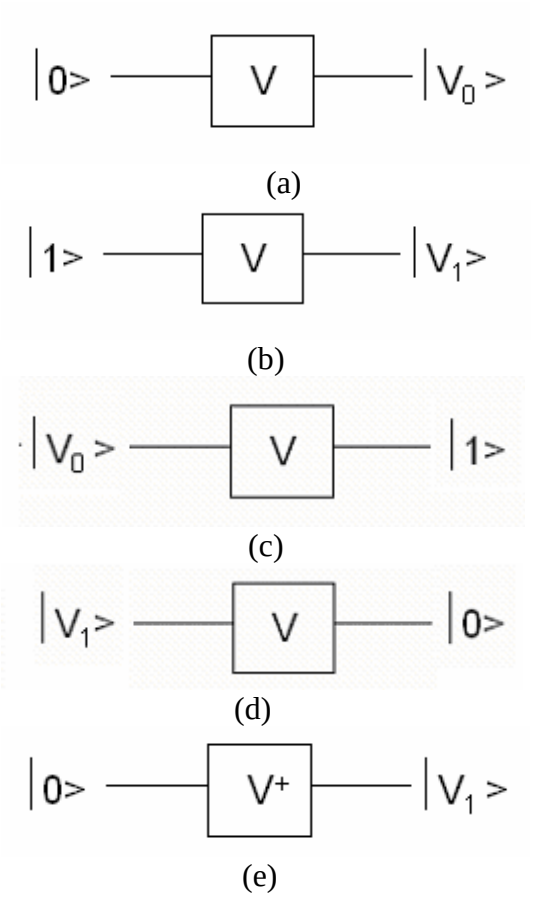


Figure 5. Calculating all possible superposition states that can be obtained from pure states |0> and |1> using V and V+ gates.



Controlled gates

Figure 4b shows a Controlled-V gate (Controlled-Square-Root-of-Not) and Figure 4c its unitary matrix. The gate operates as follows. Control signal *a* goes through the gate unaffected, i.e. *P* = *a*. If the control signal has value 0 then the qubit *b* goes through the controlled part unaffected, i.e. *Q* = *b*. If *a* = 1 then the unitary operation that is inside the box is applied to the input signal *b*, it means *Q* = V*(b)* in our case. This operation is general for all binary controlled gates, for instance the Controlled-V-hermitian (Controlled-Square-root-of-Not-hermitian). This gate is shown in Figure 4d and its unitary matrix in Figure 4e. Ternary controlled gates are described in [3,7,10] and their fault models in [14].

3. Fault Models used in the Algorithm.

Figure 6 shows some selected examples of fault matrices (these matrices are square and in general complex non-unitary). They are usually real matrices in this paper, but in general they are complex matrices, depending on the assumed faults' nature. The fault model in our system is that a fault matrix is inserted in place of the fault, more formally, the fault matrix is inserted in the netlist of parallel-serial connections of unitary matrices of gates, as will be shown in Fig. 12, Section 6 below. Observe in Figure 6 that these are not only stuck-at fault matrices. The matrices of stuck-at faults are not unitary, but some other

fault matrices may be unitary, like for instance changing phase or inserting a Pauli rotation.

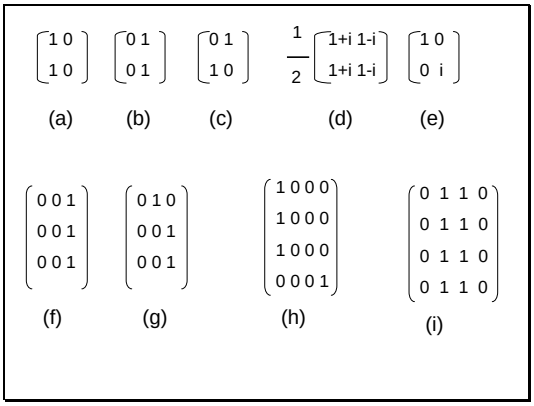


Figure 6. Matrices of faults. (a) stuck-at-0 in binary quantum circuit, (b) stuck-at-1 in binary quantum circuit, (c) inverter in binary quantum circuit, (d) stuck-at-V_0 in binary quantum circuit, (e) phase fault in binary quantum circuit, (f) stuck-at-2 in ternary quantum circuit, (g) truncated plus 1 in ternary quantum circuit, (h) AND-bridging fault in binary quantum circuit, (i)stuck-at-entangled-state-01-10 of two quantum wires.

4. Principles of Test Generation for Binary Quantum Circuts

At first, we consider a quantum equivalent of stuck-at faults in this paper. It means, the gate is stuck to a pure state or a superposition state that can be created from initial pure states using the gates used in the system. It means, if the circuit contains only permutative gates and (controlled by pure signals) gates V and V^+, the four values listed above; { |0>, |1>, $|V_0>$, $|V_1>$ } can occur as the correct and the stuck-at values. In case of the circuit from Figure 7, every quantum wire, denoted by a letter *x, y, z, p, r, q, s* can be stuck to one of the values: { |0>, |1>, $|V_0>$, $|V_1>$ }. We call it "four-valued algebra" keeping in mind that V_0 and V_1 are symbolic representations of pairs of real numbers, as shown earlier. In general, symbols correspond to groups of complex numbers. The output measurements of a faulty circuit is in general probabilistic, because of the use of the non-permutative quantum gates such as Controlled-V, resulting in measured random outputs. To illustrate the method, the fault-free output is calculated and is represented by a K-map of function *f* from Figure 7a, as shown in Figure 7b. Similarly, the K-map of the faulty output is calculated in four-valued algebra for outputs when there is a stuck-at-fault at the point *p* in quantum wire *c* of the circuit from Fig. 7a. By comparing the faulty output (in general, a vector of complex numbers) with the faultless output (a binary vector), it can be observed whether the test for this fault is ***deterministic*** or ***probabilistic***. As seen from Kmaps in Figure 7b-e the tests *a'b'c'* and *ab'c'* are deterministic and tests *a'bc', a'bc, abc'* and *abc* are probabilistic (by *b'* we denote a negation of *b*). The Kmap can be drawn for each output separately (Figure 7b,c) or for all outputs together (Figure 7d,e). All outputs are measured (observed) together (Figure 7e). The internal signals (complex numbers) of the quantum world cannot be observed by the testing program. Only the externally measured classical signals |0>, |1> (for binary) or |0>, |1>, |2> (for ternary) in the classical world are observed and measured at the input to the Tester Program (see Figure 14). We know, however, from the symbolic output what binary vector values can occur and with what probabilities on the output. So all possible measurement vectors can be known in advance with their probabilities. If these values can collapse to the correct output vector (of a non-faulty circuit) they are called probabilistic tests, otherwise they are called the deterministic tests. If they are the same in the faulty and non-faulty circuit then they cannot be used as tests for this fault which is represented by a zero or a blank cell in the Fault Table (Figure 13).

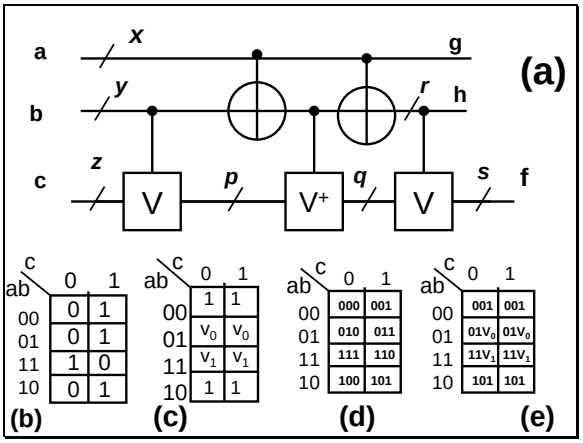


Figure 7. (a) Realization of a Toffoli gate from Controlled V, Controlled V^+ and Feynman gates.Places of faults are denoted by small italic letters.(b) Kmap of the fault free output f for circuit from Figure 6, and the Kmap of the same output f when there is a stuck-at-1 fault in quantum wire p, (c) symbolic output for function f for stuck-at-1 fault in wire p, (d) binary output signals of a correct circuit, (e) symbolic faulty outputs with probabilistic tests for the same fault.

If the outputs are "deterministic", we apply test vectors and detect the fault using standard deterministic approach to testing. If the outputs are "probabilistic" (complex numbers denoted by symbolic values like V_0 and V_1 in our case), we calculate the probability of occurrence of the observed output as explained in the following sections. We give priority to deterministic tests. Observe that a deterministic fault in quantum circuit can be observed on the output probabilistically. There are also probabilistic faults that are observed probabilistically.

4.1. Probability calculation of the random output

By iteratively applying the same input test vector (a probabilistic test) we are calculating the probability of getting the observed output. The input vectors are always vectors of pure states. Each successive iteration reduces the probability of obtaining a correct measurement for a faulty circuit. For five iterations the probability of obtaining a correct measurement in presence of a given fault is reduced to 1/32. Similarly for six iterations it is 1/64 and for eight iterations it is 1/256 (See Figure 8). Hence, the greater the number of iterations the lesser is the probability of getting the correct measurement for faulty circuit. Suppose that we use test 110 for fault S-a-1 in location *p*. According to Fig. 7d the correct output should be 111 and according to Figure 7e the faulty output is $11V_1$, which means 110 or 111 with equal probabilities of ½ are measured. If we tested eight times and we get |1> in bit *f* each time, we have the probability 255/256 that there no fault S-a-1 in location *p*. Of course, in this particular case we should apply deterministic tests *a'b'c'* or *ab'c'* since they are possible. In general, probabilistic tests may not exist. The number of iterations should depend on the expected accuracy and cost of testing.

5. Generalization to ternary logic

For a given reversible circuit and a fault set, a test set is said to be the complete test set if and only if the test vectors can detect all possible faults [8]. A complete test set with minimum number of vectors is called the minimal test set. The two properties of reversibility namely, *Reversible Controllability* and *Reversible Observability* simplify the test set generation problem [12]. The Reversible Controllability states that there are test vectors that will generate any given desired state on the wires at any given level. The Reversible Observability states that any single fault that changes an intermediate state in the circuit will also change the output.

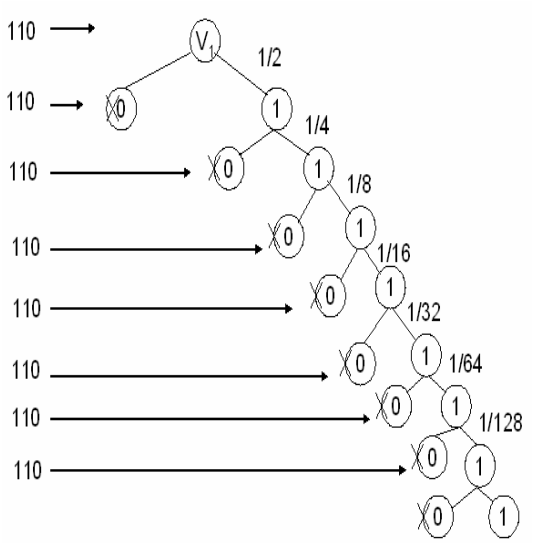


Figure 8. Probabilistic Supernode – a tree for calculating the probability of obtaining a sequence of n signals "1" from a gate which has probability ½ of output "1".

5.1. Generalization of propositions from binary to ternary logic

PROPOSITION 1: Under the single stuck-at-fault model (i.e. stuck-at-0 and stuck-at-1 in binary), a test set is complete if and only if each of the wires at every level can be set to |0>, |1> and |2> by the test set.

PROPOSITION 2: Any test set that is complete for the single stuck-at-fault model is also complete for the multiple stuck-at-fault model.

LEMMA 1: Each test vector covers exactly two-third of the faults, and every fault is covered by exactly two-thirds of the possible test vectors.

These propositions can be observed from the circuit in Figure 9. It is straightforward to further generalize these results to a multiple-valued logic with arbitrary radix. Thus similar propositions and lemma can be used in quantum or reversible logic of any radix.

5.2. Testing the ternary reversible circuit

The ternary circuit in Figure 9 uses 3 quantum wires and has the depth of 3. The gates are ternary Feynman gates, they use modulo 3 addition instead of EXOR (which is a modulo 2 addition). The analogy to binary case is straightforward.

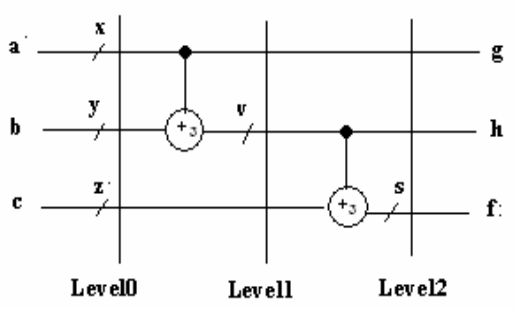


Figure 9. Ternary reversible circuit

Greedy approach is adopted to detect faults. It has been analyzed that a minimum of two test vectors can cover all the faults. Hence the minimal test set has two vectors (i.e., 012 and 121), as illustrated in the table in Figure 11. In this table we denote by X0 the stuck-at |0> fault in place X, by X1 the stuck-at |1> fault in this place and by X2 the stuck-at |2> fault in point X. Symbol * at the intersection of row Ri and column Cj denotes that test Ri detects fault Cj. Because in reversible logic there are relatively many tests for each fault, the test generation is easier than in irreversible logic. Creating fault tables and adaptive trees [8,15] for fault localization for binary and ternary reversible logic and quantum computing is very similar [15].

6. Test Generation and Fault Localization System.

The system for test generation and fault localization in binary and ternary quantum circuits is shown in Figure 10. We generate tests and use them to localize the fault in quantum circuits (details about fault localization are in [15]). Localization is done using an adaptive tree (decision tree, diagnostic tree, fault tree). In adaptive tree there are two types of nodes, deterministic nodes which are the same as in standard decision (adaptive) tree, and probabilistic supernodes (which represent small trees of probabilistic tests like one from Figure 8). In contrast to reversible circuits [15] where forward and backward fault simulation is applied for the test generation efficiency, in case of quantum circuits only forward simulation is possible since using backward simulation from the arbitrary fault point back to inputs may result in mixed (superposition) states on inputs, and we can create only pure states on the circuit inputs. The algorithm for quantum fault simulation is similar to standard fault simulation in a irreversible circuit, only the fault model is more general now.

The sequence of steps is as follows.

1. *Preprocessing the netlist.*

First the circuit is preprocessed to a form for which it is possible to apply operators of Kronecker multiplication (for parallel connection of sub-circuits (gates, blocks, columns). Standard matrix multiplication are applied for serial connection of gates (columns). As seen in Fig.12, swap gates are introduced in such a way that the circuit is build from parallel and serial connections of blocks. This allows to calculate a resultant matrix of each column using Kronecker multiplications. This was not possible if a wire was crossing a gate description as in Figure 12a in the first gate from the left.

2. *Creating a resultant unitary matrix of a correct circuit.*

A non-faulty circuit is simulated for all pure state input combinations to create a fault-free table like that from Figure 7d. It is done as follows. The program calculates the resultant transition matrix of the correct circuit from component matrices of gates (unitary) and faults (usually non-unitary). Operations of Kronecker matrix multiplication and standard matrix multiplication are used to calculate the resultant matrices of non-faulty and faulty circuits [10].

3. Calculation of all correct output vectors corresponding to input vectors.

The resultant matrix of the correct circuit is multiplied by the vector of all possible (pure state) input vectors to obtain the vector of all corresponding output vectors.

4. *Generating tests for every possible fault*

Every single fault is inserted and all tests for it are generated as in Figure 7e (currently we use exhaustive methods as in points 2,3 above). Assume a fault F1 is inserted in place X. It means, a value different than the stuck-at value should be in this place. (If a correct value is V_0 then 0, 1 and V_1 are faulty values in our case). This way, for certain input vector, a vector of faulty values is created in the place and level of the fault. Every fault is simulated by inserting a unitary or non-unitary transition matrix in the location of this fault. For stuck-at faults the

matrices are non-unitary. This way we can simulate not only the multiple stuck-at faults in the same level but many other fault types as well. The circuit is continued to be simulated forward from the level of the fault to create finally the output vector of complex numbers (see Figure 7e).

As an example, in Figure 12 the matrix of block B1 is S ⊗ I, where I is 1-qubit identity matrix, S is swap gate matrix and ⊗ is a symbol of Kronecker multiplication. Similarly matrix B2 = I ⊗ CV, where CV is a matrix of Controlled-V with control up. B3 = B1. B4 = I ⊗ I ⊗ S-a-0, where S-a-0 is non-unitary matrix of the stuck-at-0 fault. B5 = B2. B6 = Fe ⊗ I. B7 = I ⊗ CV^+ where CV^+ is a matrix of Controlled-V^+ with control up. B8 = B6. The final matrix with the inserted fault is B8 * B7 * B6 * B5 * B4 * B3 * B2 * B1. For every fault the set of all corresponding output vectors is calculated using the respective resultant transition matrix of the faulty circuit.

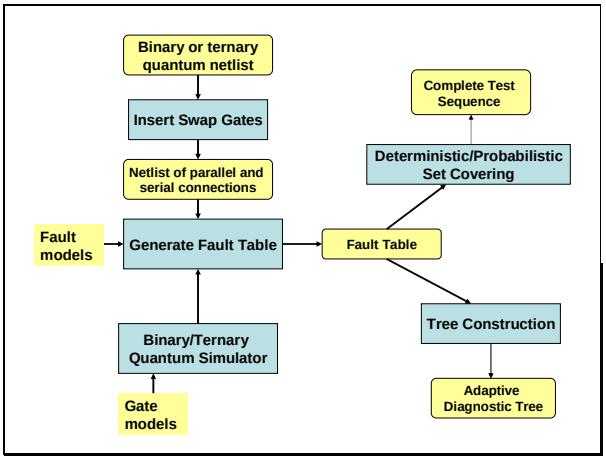


Figure 10. The block diagram of the system for test generation and fault localization in binary and ternary quantum circuits.

For each test, the output vectors of the faulty and non-faulty circuits (Figure 7d,e) are compared to create a column of the Fault Table. This is iterated for all fault models in all locations. A part of Fault Table for a correct circuit and a column for fault S-a-1 in location *p* is shown in Figure 13. It was created from Figure 7. The Fault Table is a starting point to both the minimal test sequence generation and adaptive tree generation. The entries in the table for a quantum circuit are more general than for a irreversible [8] or reversible [15] circuit. An empty cell (equivalent to entries "no" "no" in both subcolumns) at the intersection of fault column Fi and row (test vector) Tj means that fault Fi is not detected by vector Tj. A symbol 1 at the intersection of column Fi (at the right of the left subcolumn) and row Tj mean that test vector Tj is a deterministic (probability =1) test able to detect fault Fi. A number other than 1 is a probability that test Tj detects fault Fi. For instance, ½ means that test Tj detects fault Fi with probability one half. However, by repeating this test successfully two time the probability is increased to ¾. In some variants (shown in Figure 13) the Fault Table stores also the correct output, output for the detected fault in the column of the fault, and other possible output vectors for this test in case of this fault. These data are used in probabilistic test sequence generation, adaptive tree generation and adaptive testing.

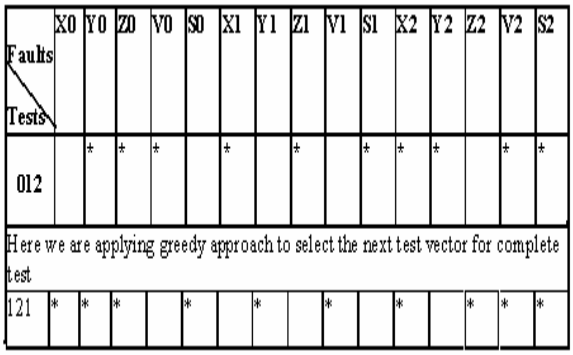


Figure 11. Two rows of Fault Table for the Ternary reversible circuit. These rows are enough to solve the fault covering problem in which all columns should be covered by a minimum subset of rows.

A probabilistic approach is used to calculate the occurrence of faulty output, by applying the test vectors when the outputs are random in nature. Observe that even if there are probabilistic outputs in output test vectors, a fault can be sometimes found with 100% certainty. For instance, if the correct circuit gives output reading 100 and the symbolic output complex vector is $00V_0$, we know that the circuit is faulty because the first bit differs. Thus, whatever the random number assigned to V_0, the measured output vector will be different from the expected output vector 100. This example points to the importance of knowing during measurements which output bits are probabilistic and which are deterministic. This information is stored in the

Fault Table (Fig. 13) that is used by the programs (Fig. 14).

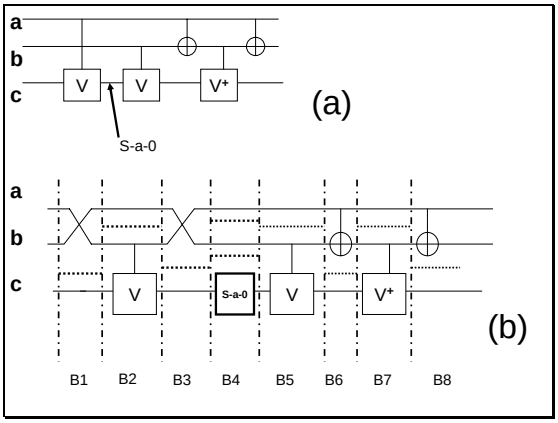


Figure12. (a) Netlist of another variant of Toffoli gate represented as a cascade of simple quantum gates with S-a-0 fault inserted,, (b) cascade from (a) preprocessed by inserting swap gates to allow Kronecker and matrix multiplications. Observe the inserted matrix of S-a-0 fault in column B4

Input test vector	Correct output vector	...	Fault S-a-1 in p		.
		..	Faulty output	Other outputs	.
000	000		001 1	no	
001	001	.	no	no	
010	010	...	011 1/2	010	.
011	011		010 1/2	011	
100	100	...	101 1	no	
101	101	..	no	no	.
110	111		110 1/2	111	.
111	110	.	111 1/2	110	

Figure 13. A segment of a Fault Table with correct outputs for each input vector (test) and a column for s-a-1 fault in location p. Note the subcolumn for faulty output with its probability and the subcolumn for other output values measured.

Using the system for testing quantum circuit models and quantum circuits.

Figure 14 shows the system for testing the test generation program using the approach outlined above. To verify the operation of the system without a quantum computer the tests are given to the software model of the correct and faulty quantum circuit. The Tester Program returns "good/bad" decision about the circuit and also (optionally) the localization of faults and other information. The switch at the bottom center of the Figure takes the outputs from either the model of a quantum circuit or from the measurement of the real quantum circuit in the quantum computer.

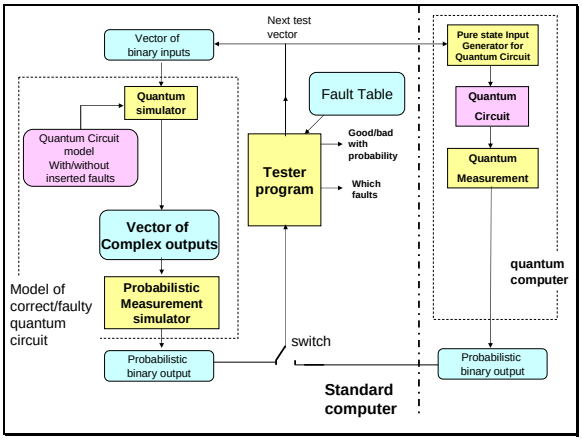


Figure 14. The system to test the test generating and tester programs and to use the tester for quantum circuits.

7. Conclusion

The presented paper presents the first attempt at the test generation and fault localization for both permutative and non-permutative quantum circuts. With powerful enough fault model the approach can be used to arbitrary quantum circuits. In software, we used standard realization of matrix operations to implement matrix and Kronecker multiplications on unitary and non-unitary matrices [9]. Because these operations are repeated many times, the speed of the program suffers and we cannot handle larger functions. Currently we can work only with 3-qubit and 4-qubit binary and ternary circuits. A better approach, that we are working on, is to use a new technique for gate-level simulation of quantum circuits that was recently proposed by Viamontes et al [17,18]. It is based on a new data structure called Quantum Information Decision Diagrams (QuIDDs), which are generalizations of Binary Decision Diagrams, well-known for their ability to efficiently represent many problems. We are going to use these new diagrams in synthesis problems as well [3,4,7,9].

In the proposed method we repeat the same test several times in supernodes, used both for test sequence generation and for fault localization in

adaptive trees. It can be observed, however, that statistical information can be obtained also from various different tests [14], thus shorter test sequences can be perhaps build for the same circuits and with the same error probability than the sequences generated according to the presented algorithm.

Future works include developing more efficient test generation and fault localization algorithms for binary, multi-valued and mixed (binary-ternary) quantum circuits composed of arbitrary permutative and non-permutative gates [14]. We will also use all fault models that are necessary in real quantum computing, such as NMR.

References

1. A. Buller, Quantrix: Toward Automated Synthesis of Quantum Cascades, Technical Report TR-HIS-00-*, ATR Human Information Processing Laboratories, Kyoto, 2003. 03.15.
2. H.F. Chau, Correcting quantum errors in higher spin systems. *Physical Review A*, Vol. 55, R839-R841, 1997.
3. E. Curtis, and M. Perkowski, Minimization of Ternary Reversible Logic Cascades using a Universal Subset of Generalized Ternary Gates, submitted to *International Journal on Multiple-Valued Logic and Soft Computing, Svetlana Yanushkevich, editor*
4. W. Hung, X. Song, G. Yang, and M. Perkowski. Reachability Analysis for reversible minimization. *Proceedings of DAC 2004.*
5. http://www.dhushara.com/book/quantcos/qcompu/qc2/qc2.htm
6. http://www.themilkyway.com/quantum/FinalReport/QuantumGates.html
7. M. H. A. Khan and M. Perkowski, Genetic Algorithms Based Synthesis of Multi-Output Ternary Functions Using Quantum Cascade of Generalized Ternary Gates, submitted to special issue of *International Journal on Multiple-Valued Logic and Soft Computing, Tatjana Kalganova, editor.*
8. Z. Kohavi, Switching and Finite Automata Theory, *Mc Graw-Hill*, 1978.
9. M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, Ch-H. Yu, K. Chung, H. Jee, B.G. Kim, and Y-D. Kim, Evolutionary approach to Quantum and Reversible Circuits synthesis, *Artificial Intelligence Review Journal*, Special Issue on Artificial Intelligence in Logic Design, S. Yanushkevich guest editor, 2003.
10. A. Muthukrishnan and C. R. Stroud Jr., "Multivalued Logic Gates for Quantum Computation," *Physical Review A*, Vol. 62, pp. 052309.1-8, 2000.
11. M. Nielsen, and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
12. K.N. Patel, J.P. Hayes and I. Markov, "Fault testing for reversible circuits," Proc. *VLSI Test Symp.* (*VTS 03*), Napa, CA, pp. 410–416, April 2003.
13. B. Patterson, http:www.cs.iastate.edu/~patterbi/cs/quantum.fp/FinalPaper.pdf
14. M. Perkowski, and J. Biamonte, Probabilistic Testing and Fault Localization of Binary and Ternary Quantum Circuits, *to be submitted*, 2004.
15. K. Ramasamy, R. Tagare, E. Perkins, and M. Perkowski, Greedy Algorithm for Fault Localization in Binary Reversible Circuits, *submitted to IWLS 2004.*
16. T. Toffoli, „Reversible Computing", in *Automata, Languages and Programming* (edited by de J. W. Bakker and J. van Leeuwen), Springer Verlag, pp. 632-644, 1980.
17. G.F. Viamontes, M. Rajagopalan, I.L. Markov and J.P. Hayes, Gate-Level Simulation of Quantum Circuits, quant-ph/0208003.
18. G. F. Viamontes, I. L. Markov and J. P. Hayes, ``Improving Gate-Level Simulation of Quantum Circuits" (quant-ph/0309060), to appear in *Quantum Information Processing*, 2004.