A Quantum Algorithm for Computing All Diagnoses of a Switching Circuit

Alexander Feldman, Johan de Kleer, and Ion Matei

Xerox PARC, Intelligent Systems Lab
3333 Coyote Hill Road, Palo Alto, CA 94304
{afeldman,dekleer,imatei}@parc.com

Abstract

Faults are stochastic by nature while most man-made systems, and especially computers, work deterministically. This necessitates the linking of probability theory with mathematical logics, automata, and switching circuit theory. This paper provides such a connecting via quantum information theory which is an intuitive approach as quantum physics obeys probability laws.

In this paper we provide a novel approach for computing diagnosis of switching circuits with gate-based quantum computers. The approach is based on the idea of putting the qubits representing faults in superposition and compute all, often exponentially many, diagnoses simultaneously.

We empirically compare the quantum algorithm for diagnostics to an approach based on SAT and model-counting. For a benchmark of combinational circuits we establish an error of less than one percent in estimating the true probability of faults.

1 Introduction

Diagnostics of complex systems with many components is a challenging and important topic [1]. Problems in Model-Based Diagnosis (MBD) are often NP-hard or worse [2]. Quantum computing is a modern and promising approach to solving a range of algorithmic challenges [3]. Researchers have already studied the problem of finding a single minimal/cardinality diagnosis with an adiabatic quantum computer [4]. In this paper we propose a novel algorithm for the simultaneous computation of all diagnoses of a system by using a traditional gate-based quantum computer.

The diagnostic problem we solve in this paper is framed as computing a probability distribution function for each fault that can occur in a switching circuit. This is driven by the fact that both in physics and in complex systems, faults are stochastic. Classical computers and switching circuits are deterministic. The result of a classical quantum computation is always a probability distribution function. Therefore, the mathematical framework defined by a quantum computer can be used for combining switching circuits and probability.

The diagnostic problem we solve computes a probability distribution function of each fault. The complexity of this problem is, to the best of our knowledge, unknown. We hypothesize that the problem is #P-hard, the same as counting all solutions of a Disjunctive Normal Form (DNF) formula. To analyze the correctness of our quantum algorithm we compare it to a classical one based on enumerating all solutions via satisfiability [5] and blocking clauses.

We have empirically evaluated the efficiency of the quantum diagnostic algorithm on a benchmark of circuit families. Even in simulation, the algorithm can handle surprisingly large circuits. We have had success with circuits consisting of up to 30 gates. All experiments resulted in a small error, typically within less than 1% compared to the golden standard computed by deterministic model counting.

The algorithm we present does not add to complexity theory. Although, theoretically, one can use a quantum algorithm for model-counting and 3-SAT, in practice, the proposed algorithm is only for approximation. The approximation error is arbitrarily large, especially, when the underling SAT problem is in the phase-transition region [6]. We discuss this in more detail in Sec. 6.

2 Preliminaries

This text explains letters and symbols and helps with the intuitive understanding of Boolean circuits. For precise definitions of Boolean circuits, see the mathematical introduction to circuit complexity by H. Vollmer [7]. For a thorough treaty on the subject of quantum circuits, the reader is referred to the book on quantum computing by M. Nielsen and I. Chuang [8].

2.1 Switching Circuits

A switching circuit C implements a multi-output Boolean function $f_C : \{0,1\}^m \to \{0,1\}^n$. The m arguments of f_C are primary inputs in C and are denoted as $X = \{x_1, x_2, \ldots, x_n\}$. Similarly, the n results of f_C are primary outputs in C and are specified as $Y = \{y_1, y_2, \ldots, y_m\}$.

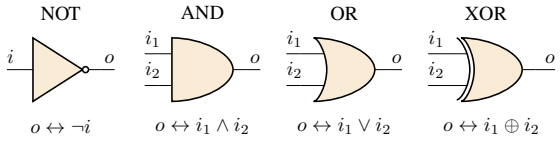


Figure 1: The standard basis

A switching circuit is a graphical network that connects gates with wires. Each gate implement some (typically small) Boolean function. This set of elementary Boolean functions is called the basis of the circuit. Figure 1 shows a common set of gates and their corresponding Boolean formulas that are used often in computer science and VLSI design. It contains of two-input AND, OR, and XOR gates and an inverter. With this minimalistic basis, multi-input AND and OR gates are implemented by chaining; and NAND, and NOR gates are implemented by appending invertors to AND and OR gates, respectively.

Some common Boolean functions are negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), exclusive or ($\oplus$), implication[1] ($\to$), and equivalence ($\leftrightarrow$). This paper uses everywhere infix, as opposed to prefix, notation. For example, $p \vee q$ is used instead of $\vee\,(p, q)$.

We also use equivalence ($\leftrightarrow$) instead of the equal sign ($=$) to specify Boolean functions. The function output is on the left while the inputs are on the right. For example, the Boolean function $r = p \vee q$ is written as $r \leftrightarrow p \vee q$. When there are multiple outputs, we give a formula for each one of them.

Figure 2 shows a simple and frequently used switching circuit that is used for adding the two binary numbers i_1 and i_2 and a carry input bit c_i. The output is found in the sum bit σ and in the carry output c_o. Notice that there are two identical subcircuits in Figure 2. These are the two half-adders.

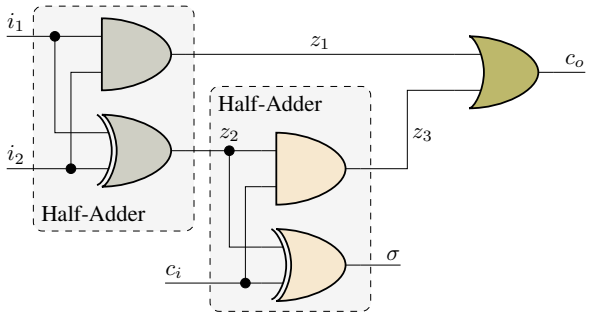


Figure 2: A full-adder

An assignment to the primary inputs or outputs of a switching circuit is a set of variable/value pairs. The values are the Boolean constants 0 or 1. An assignment can be written as a propositional conjunction. For example assigning

[1]This paper, similar to many others, shares the same symbol ($\to$) for implication and for function mapping. The use is clear from the context.

$i_1 = 1$, $i_2 = 0$, and $c_i = 1$ is written as $\neg i_1 \wedge \neg i_2 \wedge c_i$.

2.2 Quantum Circuits

Although, on the surface, quantum circuits can be reminiscent of the switching ones described above, there are multiple important differences. The biggest one is, that while the state of a Boolean circuit is a Boolean vector over all wires, the state of a quantum circuit is a superposition over all quantum bits. The quantum state is of size, exponential to the number of quantum bits.

A quantum circuit is a set of quantum wires $\psi_1, \psi_2, \ldots, \psi_n$ that go through quantum gates. In reality a quantum wire does not have to look anything like a piece of copper metal but can be, for example laser light, or even some passage of time.

The simplest quantum circuit consists of one quantum wire ψ_1 and one or more quantum gates that act on this single wire. The state of the quantum circuit can be illustrated as a position on a fictional Bloch sphere (see Figure 3). Using P. Dirac's notation, the state of the circuit is $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$. In this bootstrapping example all kets are vectors of size two, $|0\rangle$ is $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, and $|1\rangle$ is $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$.

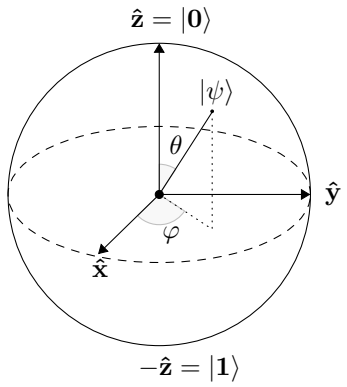


Figure 3: Bloch sphere

The state of a quantum circuit with two qubits is a superposition with four terms: $|\psi_1\psi_2\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$ where $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11} \in \mathbb{C}$.

Three types of quantum gates are sufficient for illustrating the quantum algorithms of this paper: (1) Pauli-X, (2) Hadamard, and (3) CNOT gate. The symbols and matrices of these gates are shown in Figure 4.

3 Algorithms

After a brief introduction to fault-modeling and conditional fault probability, we present a SAT-based algorithm for diagnostic model-counting and then we proceed with the main contribution of this paper: the quantum algorithm for circuit diagnostics.

3.1 Fault-Modeling

Our approach to diagnosis is to automatically modify the input circuit C and to use a generic reasoning algorithm which does not use the notion of faults. The method we employ is based on sub-circuit rewriting. Depending on the exact type of rewriting, it is possible to model stuck-at faults, gates behaving like other gates (for example an AND-gate, behaving like an OR-gate), short-circuits between two or more wires, and others.

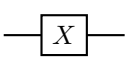
In this paper, similar to Automatic Test-Pattern Generation (ATPG) [9], we are interested in stuck-at-faults. Unlike in ATPG where the fault location can be both at the stem and at the branches of a fan-out we place stuck-at faults only

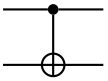
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathrm{CCNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$





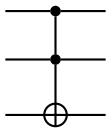


Figure 4: Quantum gates and their corresponding matrices

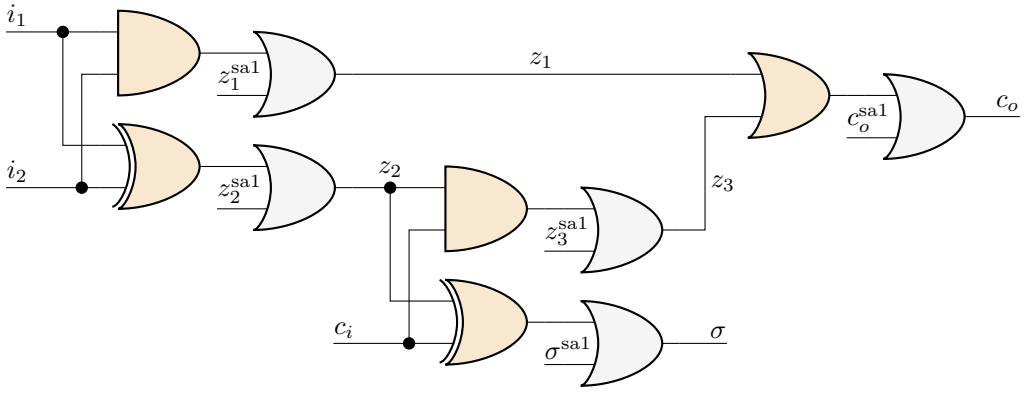


Figure 5: A fault-augmented full-adder

at gate outputs. To model for a stuck-at-one gate, it is enough to insert an OR-gate at the fault location. The second input of the OR-gate becomes a special type of a primary input, called *assumable*.

Figure 5 shows the full-adder example from Figure 2 with all outputs of the original five gates allowed to be stuck-at-1. The five new assumable inputs are z_1^{sa1}, z_2^{sa1}, z_3^{sa1}, σ^{sa1}, and c_o^{sa1}.

3.2 Conditional Probability of Faults

In the preceding section, we have shown, how computation of a diagnosis can be reduced to finding compatible values for a special subset of inputs, the fault inputs. In reality, however, faults show stochastic behavior, so it is worth to establish a computational framework for probabilities.

Connecting probabilities to switching circuits has been done by John von Neumann [10], while Parker and McCluskey show how, from a circuit and the probability of each input one can compute the probability of each output [11]. Parker and McCluskey show two algorithms for computing the probabilities of the circuit outputs: one that first computes all prime implicants of the Boolean function modeling the switching circuit, and the other by starting from the inputs and performing real-value arithmetics over the probability values. It is of no surprise that the second algorithm has polynomial complexity as forward propagation from inputs to outputs can be done easily.

We extend the framework of K. Parker and E. McCluskey by allowing one to compute the probability of any wire from any set of probabilities we know. In particular, we are interested in computing the probabilities of a subset of inputs, the fault inputs given the primary inputs and the primary outputs of a circuit. For that we use an NP-complete approach, i.e., we implicitly create a table with all consistent values. Each row of the table is a diagnosis and each column is a fault variable. To compute the probability of a fault we divide the number of diagnoses in which the specified fault shows-up by the total number of diagnoses.

Consider the circuit C_f from Fig. 5, the primary inputs assignment $\alpha = \neg i_1 \wedge \neg i_2 \wedge c_i$, and a double-fault injection $\gamma = \sigma^{\mathrm{sa1}} \wedge c_o^{\mathrm{sa1}}$. The primary outputs are actually determined only by the fault γ and are $\beta = \sigma \wedge c_o$. Table 1 contains all diagnoses of C_f, α, and β; as computed by Algorithm 1. There is a total of 22 diagnoses and their distribution resembles binomial as there is relatively little masking in this example circuit.

z_1^{sa1}	z_2^{sa1}	z_3^{sa1}	σ^{sa1}	c_o^{sa1}
0	0	0	0	1
0	0	1	0	0
1	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
0	0	1	1	1
⋮	⋮	⋮	⋮	⋮

z_1^{sa1}	z_2^{sa1}	z_3^{sa1}	σ^{sa1}	c_o^{sa1}
⋮	⋮	⋮	⋮	⋮
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0
1	1	1	1	1
$\frac{12}{22}$	$\frac{8}{22}$	$\frac{12}{22}$	$\frac{15}{22}$	$\frac{12}{22}$

Table 1: Truth-table, diagnoses and conditional fault-probabilities for the full-adder running example and an observation

3.3 An Algorithm for Computing Conditional Fault Probabilities Based on Satisfiability

Algorithm 1 is a classical reference algorithm for calculating the conditional probability $P(f_i|C, \alpha)$ for each $f_i \in F$. It converts the circuit C to a Boolean formula in Conjunctive Normal Form and repetitively calls a SAT solver to compute satisfiable solutions that also happen to be diagnoses. After each call to the SAT subroutine, a row is added to a table from which the conditional probability can be computed and a blocking clause is added to prevent the satisfiable solution ω from showing-up again.

The truth table does not need to be created explicitly and it is sufficient to maintain a set of counters for each fault variable. These counters are denoted as $n[f], f \in F$ where F is the set of all fault variables. The total number of satisfiable assignments (diagnoses) is kept in d. To receive the probability of a fault it is enought to divide the respective $n[f]$ by the total number of diagnoses d.

3.4 Quantum-Based Conditional Probability

A Hilbert space representing a quantum system must satisfy the normalization condition. The inner product of a vector with itself is equal to one: $\langle\psi|\psi\rangle = 1$. The length of a vector in a particular direction represents the "probability amplitude" of the quantum system. The idea is to set all assumable inputs to a superposition of $|0\rangle$ and $|1\rangle$ and to apply a quantum oracle that is analogous to the Boolean circuit being diagnosed. The collapsed probability amplitude readout is the a posteriori probability distribution function of each fault given the observation α and the circuit φ.

Algorithm 1: CIRCUITHEALTHSAT

Input : C_f, the fault-augmented input circuit with fault inputs $F = \{f_1, f_2, \ldots, f_n\}$
Input : α, primary inputs assignment
Input : β, primary outputs assignment
Output : R, conditional probabilities of faults

$B \leftarrow \top$
for $f \in F$ **do**
  $n[f] \leftarrow 0$
end
$d \leftarrow 0$
while $\omega \leftarrow \text{SAT}(\text{CNF}(C_f) \wedge \alpha \wedge \beta \wedge B)$ **do**
  for $f \in F$ **do**
    if ISFAULT(f) **then**
      $n[f] \leftarrow n[f] + 1$
    end
    $B \leftarrow B \wedge$ BLOCKINGCLAUSE(ω)
    $d \leftarrow d + 1$
  end
end
$R \leftarrow \left\{ \frac{n[f]}{d} : f \in F \right\}$
return R

3.4.1 Making the Quantum Oracle Circuit

The first step is to create a quantum oracle by converting the classical circuit to a quantum one. Algorithm 2, which is classical, shows this transformation. It starts by adding a quantum wire for each primary input of the circuit. Then, for each classical gate, a quantum sub-circuit is added to the quantum one. The quantum circuits implementing the standard classical gates are shown in Figure 6.

Algorithm 2 first topologically sorts all gates in the switching circuit. This can be done with a simple graph traversal starting from the primary inputs and proceeding toward the primary outputs. The map M keeps the correspondences between the switching circuit wires and the qubits in the quantum oracle. During the initialization phase, Algorithm 2 creates a qubit for each primary input and for each fault input and places a one-to-one entries for them in M.

During the main phase of Algorithm 2, each gate in the input circuit C is replaced with its corresponding quantum sub-circuit (see Figure 6). The quantum circuits are constructed such that the information on the input wires is preserved, so it can be reused by downstream gates. Algorithm 2 also adds a new ancillary qubit for each gate.

Figure 7 shows the result of applying Algorithm 2 on the full-adder running example from Figure 5.

3.4.2 Quantum Algorithm for Circuit Diagnosis

Figure 8 shows a quantum circuit that can be used for directly computing the conditional probabilities for each fault. Similar to many existing quantum algorithms, the idea is to put each of the n unknown fault inputs $f \in F$ in a superposition by applying Hadamard gates. The primary inputs are initialized as $|0\rangle$ or $|1\rangle$ corresponding to the Boolean values of the input assignment α.

The quantum oracle U is constructed by Algorithm 2. Before the oracle conversion the input multi-output Boolean circuit is converted to a single-output one. This single output circuit contains also the values of the observed primary outputs β. Each primary outout of the multi-output circuit is fed to the input of a multi-input AND-gate. The outputs go directly to the inputs of the AND-gate or through inverters, depending on the observed values for β (primary outputs corresponding to zero bits are inverted).

As is customary for many classes of quantum algorithms, the experiments is repeated multiple times. Each result is the collapsed state of the system shown in Algorithm 2. It is a probability distribution function over all possible

Algorithm 2: MAKEORACLE(C)

Input : C, the input circuit
Output : U, the quantum circuit oracle
Local Variables : M, dictionary, Boolean variables to quantum wires map

$U \leftarrow \emptyset$
$C = \text{SORTGATES}(C)$
for $q \in \{1, 2, \ldots, m\}$ **do**
  $M[q] \leftarrow q$
end
for $g \in \{1, 2, \ldots, n\}$ **do**
  switch GATETYPE(G) **do**
    $\psi_1 = M[i_1], \psi_2 = M[i_2], \ldots$
    case *Inverter* **do**
      $U(i) \leftarrow \text{CCNOT}(I \otimes X) \left|\psi_1 a_i\right\rangle$
    end
    case *AND-gate* **do**
      $U(i) \leftarrow \text{CCNOT} \left|\psi_1 \psi_2 a_i\right\rangle$
    end
    case *OR-gate* **do**
      $U(i) \leftarrow \text{CCNOT X}^3 \ \left|\psi_1 \psi_2 a_i\right\rangle$
    end
    case *XOR-gate* **do**
      $U(i) \leftarrow \text{CNOT}(I \otimes \text{CNOT}) \left|\psi_2 \psi_1 a_i\right\rangle$
    end
  end
  $M[o] = a_i$
  $i \leftarrow i + 1$
end
return U

combinations of values for the fault inputs and the single output of the system. The answer of the diagnostic problem is taken only from the experiments that result in positive values for the combined output o. Alternative if there is not enough probability mass in the positive values of o, the result can be marginalized from $\neg o$. In this case one obtains first the probability of each gate of being healthy as opposed to being faulty.

Let us work out a circuit that has a single inverter with an input i and an output o that is known to be true. The full quantum circuit that is used for computing $\Pr(o = 1)$ is shown in Figure 9.

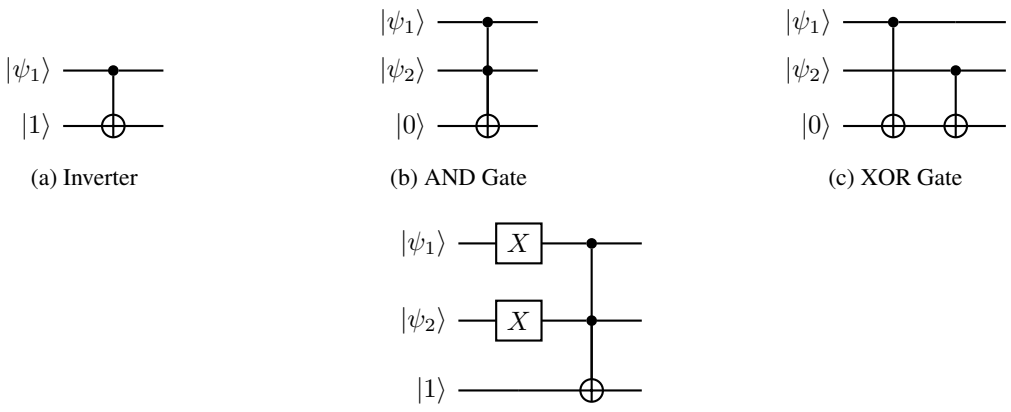
As the circuit in Figure 9 two qubits only, its state can be fully described by a complex vector of size four. Notice that in this whole paper the assumptions on the initial state and on the unitary transformations are such that the imaginary parts of all complex numbers are always zero.

The quantum circuit shown in Figure 9 works by first multiplying the unitary matrix of the quantum circuit

$$(X \otimes H)\,\text{CNOT} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

with the initial state $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\intercal$. This result in the final state of the quantum circuit $1/\sqrt{2} \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}^\intercal$. In Dirac's



(a) Inverter (b) AND Gate (c) XOR Gate

(d) OR Gate

Figure 6: Quantum implementation of classical gates from the standard basis

notation the state is $1/\sqrt{2}\,|01\rangle + 1/\sqrt{2}\,|10\rangle$. After collapsing the state, the resulting histogram is $\Pr(|01\rangle) = \Pr(|10\rangle) = 1/2$. Recall that the first qubit corresponds to the switching circuit's input i and the second qubit to the value of the global output a_o. This means that when computing the probability mass function of i we have to take only measurement probabilities in which $a_o = 1$. This leaves us only with $\Pr(|01\rangle) = 1/2$ from which it follows that $\Pr(i = 1) = 0$ which, given that the output is one, is consistent with the truth table of the inverter gate.

4 Experimental Results

Next, we perform an empirical comparison of Algorithm 1 and Algorithm 8. For the implementation of Algorithm 1 we have implemented a naïve model counter based on CADICAL [12], while for the quantum algorithm we have used QISKIT [13] and its simulation engine. All experiments were performed on a 2-CPU (4 cores per CPU) Intel Xeon 3.3 GHz Linux computer with 1.5 TiB of RAM.

Several families of arithmetic circuit are diagnoses: ripple-carry adders, ripple-borrow subtractors, barrel shifters, multipliers, multi-operand adders, multiplexers, demultiplexers, and comparators. The fault augmentation is stuck-at-one at gate outputs only. For each circuit 10 random fault injections have been generated.

To asses the performance of the quantum approach we compare it to the SAT-based approach, which is precise, complete, and sound. The error in assessing the fault state of the system is defined as:

$$\mathrm{Err} = \sum_{f \in F} \left[\Pr_s\left(f = 1\right) - \Pr_q\left(f = 1\right)\right]^2,$$

where $\Pr_s\left(f = 1\right)$ is the probability of a fault input f, computed by Algorithm 1 and $\Pr_q\left(f = 1\right)$ is the corresponding probability computed by Algorithm 8.

Figure 11 plots Err for a various set of circuits from the PARC ALU benchmark. The error is small and does not grow as a function of the circuit size. The maximum error size is ≈ 0.08 for a circuit with 20 gates which indicates that the quantum approach is suitable for approximating circuit diagnostics.

Gate-based quantum computers build the resulting probability mass function by repeating the computation multiple times. Let us denote the number of runs as N. Figure 12 shows the error as a function of N for a two-bit full adder with five inputs and three outputs. It is visible that the error decreases sharply when N becomes larger, and a small number of experiments is sufficient for a good approximation.

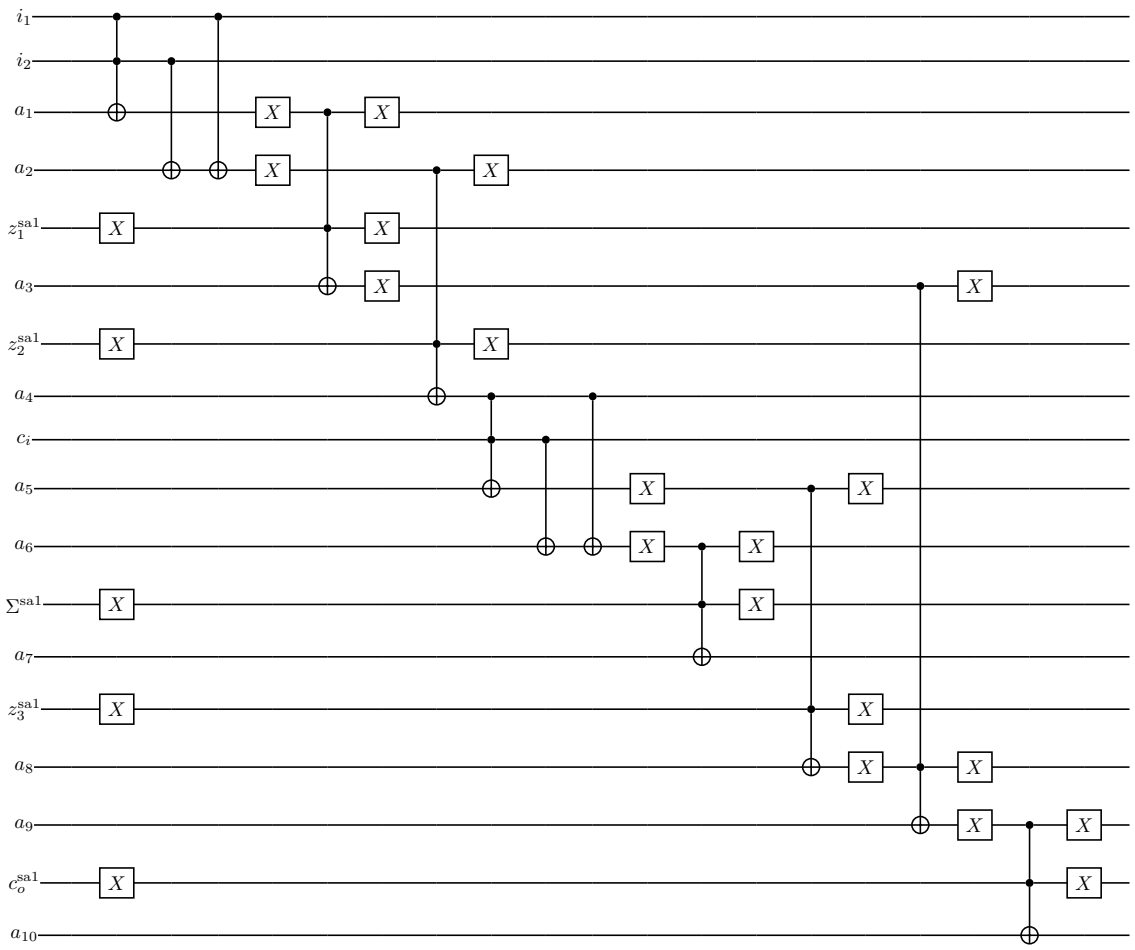


Figure 7: Quantum circuit oracle from the fault-augmented switching circuit shown in Figure 5

5 Related Work

Our approach builds upon Deutsch–Jozsa's quantum algorithm [14] for determining if a Boolean function is balanced or constant. The main difference between Deutsch–Jozsa's algorithm and ours is that while Deutsch–Jozsa's algorithm is deterministic, i.e., solves a decision problem with arbitrarily high confidence, ours computes a probability mass function. What makes our algorithm of real practical value is that most circuit diagnostic problems tend to be underconstrained due to long implication chains. While this makes classical model counting and compilation difficult, the relatively large split between diagnoses and inconsistent assignments results in probability mass functions with values sufficiently different from zero or one.

The potential of quantum computing to improve the computational performance of diagnostics has been extensively studied in the context of quantum annealers [4]. An earlier and slightly more general approach [15] reports also on experiments with random sampling of the solution space of a diagnostic problem. The biggest difference between these two papers and the algorithm presented here is that quantum annealing diagnostic algorithms compute one diagnosis at a time while in our approach we compute a superposition of all diagnoses.

Automated Test Pattern Generation [16] is related to diagnostics of switching circuit. Singh, Bharadwaj, and Harpreet have studied non-convential algorithms for ATPG based on DNA and quantum computing [17]. Their approach resembles the one presented here by the fact that they put the primary inputs of a circuit in a super position to search for a test-vector. In their case, however, a test-vector can be classical computed with a single call to a SAT

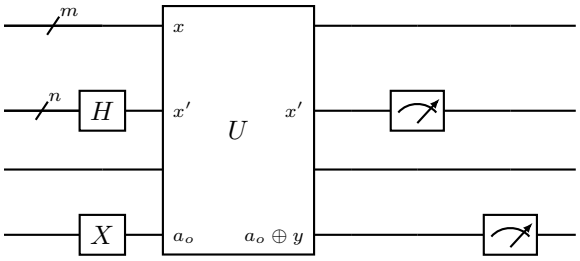


Figure 8: Quantum algorithm for circuit diagnosis

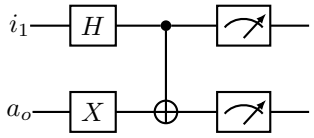


Figure 9: Diagnostic quantum circuit from a single inverter oracle

oracle, while in our case we need some kind of an enumeration of multiple solutions.

6 Discussion

On the surface, it may seem that Algorithm 8 can be used for solving hard SAT or factorization problems. Indeed, we have experimented with putting a digital multiplier circuit as an oracle, assigning some binary number to the primary outputs and computing the probability mass function of all inputs to the multiplier. After running the algorithm, the probability of the single oracle output can be used for calculating the number of satisfiable assignments. The problem, however, is that hard SAT and factorization problem have either one or a very small number of SAT assignments. This non-even split of SAT/UNSAT will be essentially lost in the error of any physical implementation of a quantum computer. Notice that the foundational Shor [18] and Deutch-Josza [14] algorithms are robust to noise, however they are closer to decision problems while we target counting.

Quantum algorithms typically follow classical ones but that does not have to be always the case. In this paper, for example, we have solved large diagnostic problems, simulating a quantum computer on a classical one. The largest circuit that we could solve, a 3-bit subtractor, has 42 gates and ? variables. This was challenging for the

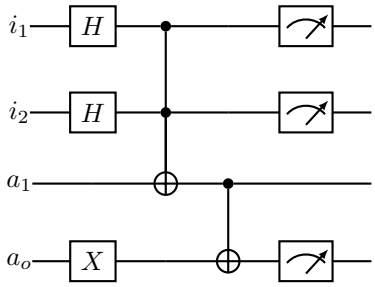


Figure 10: Diagnostic quantum circuit from a single AND-gate oracle

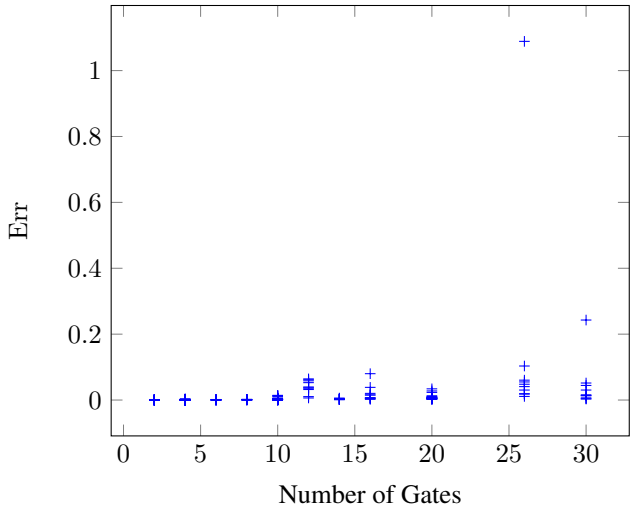


Figure 11: Quantum algorithm error as a function of the switching circuit size

SAT-based algorithm from Algorithm 1 and it took on average ?s to compute all diagnoses. The good performance of the simulated quantum algorithm is due to the fact that the overall approach resembles compilation techniques such as Ordered Binary Decision Diagrams (OBDDs) [19]. We have also noticed that all probabilities are rational numbers. This means that compilation-based approach combined with Satisfiability Modulo Theories (SMT) [20] has a lot of potential.

7 Conclusions

We have presented a quantum algorithm for computing all diagnoses in a switching circuit. The main idea is to put the bits modeling the faults in superposition and to read out the probability of each fault. The quantum algorithm is compared to an exact approach based on model counting and achieves small error.

In the future we plan to improve the quantum algorithm by reducing the number of ancillary bits in the oracle. This can be achieved by various methods for preprocessing.

References

[1] Johan de Kleer and Brian Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[2] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *Journal of the ACM (JACM)*, 42(1):3–42, 1995.

[3] Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1):1–8, 2016.

[4] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V. Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas. Readiness of quantum optimization machines for industrial applications. *Phys. Rev. Applied*, 12:014004, July 2019.

[5] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of Satisfiability*, volume 185. IOS press, 2009.

[6] Ian P Gent and Toby Walsh. The SAT phase transition. In *ECAI*, volume 94, pages 105–109, 1994.

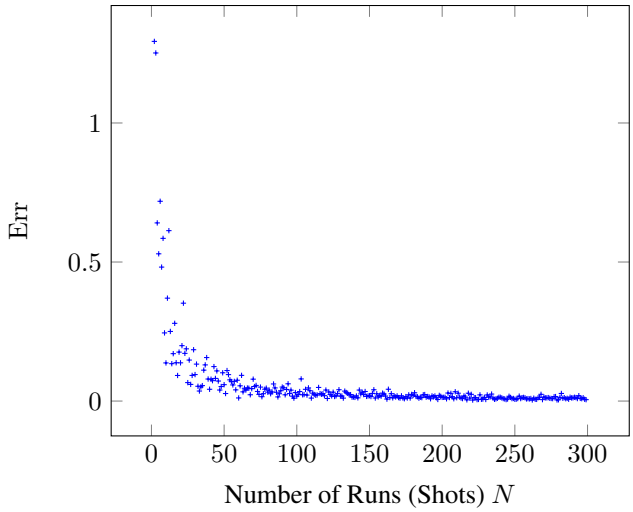


Figure 12: Quantum algorithm error as a function of the number of repetitions of the experiment

[7] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer Science & Business Media, 2013.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[9] Alexander Feldman, Mikoláš Janota, Alexandre Perez, and Johan de Kleer. A SAT-based encoding for finding a minimal automated test pattern generation test-suite. In *Proceedings of the Thirty-Second International Workshop on Principles of Diagnosis (DX'21), Hamburg, Germany*, pages 1–6, 2021.

[10] John von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34(34):43–98, 1956.

[11] Kenneth P. Parker and Edward J. McCluskey. Probabilistic treatment of general combinational networks. *IEEE Transactions on Computers*, 100(6):668–670, 1975.

[12] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[13] M.D. Sajid Anis et al. Qiskit: An open-source framework for quantum computing, 2021.

[14] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[15] Zhengbing Bian, Fabian Chudak, Robert Brian Israel, Brad Lackey, William G Macready, and Aidan Roy. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT*, page 14, 2016.

[16] Michael L. Bushnell and Vishwani D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, Boston, 2000.

[17] Amardeep Singh, Lalit M Bharadwaj, and Singh Harpreet. Dna and quantum based algorithms for VLSI circuits testing. *Natural Computing*, 4(1):53–72, 2005.

[18] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.

[19] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

[20] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.