Quantum Circuit Design and Analysis for Database Search Applications

Yi-Lin Ju, I-Ming Tsai, and Sy-Yen Kuo, *Fellow, IEEE*

***Abstract*—In this paper, we show how quantum Boolean circuits can be used to implement the oracle and the inversion-about-average function in Grover's search algorithm. Before illustrating how this can be done, we present the circuit design principle using the satisfiability (SAT) problem as an example. Then, based on this principle, we show the quantum circuits for two different kinds of applications. The first one is searching a phone book. Although this is a typical example of Grover's algorithm, we show that it is impractical as a real-world application. As the second application, we give the quantum circuits for a more practical application—breaking a symmetric cryptosystem. Although these two applications have quite different types of search criteria, they are both one-way functions and the proposed circuits can actually be generalized to any such problems. In this perspective, we conclude this paper by proposing a template of quantum circuits that is capable of searching the solution of a certain class of one-way functions.**

***Index Terms*—Grover's algorithm, nanoscale circuits, one-way function, quantum circuits.**

I. Introduction

OVER THE PAST years, the density of transistors on conventional integrated-circuit chips has been increased dramatically. However, with the components continue to shrink, the conventional transistor technology will eventually reach its physical limit. Nanoelectronics is regarded as the most likely direction of developing computer technologies. It is believed that nanocomputer can be smaller, more densely integrated and more powerful. A promising way of achieving this is to take the advantage of quantum mechanics. Recent discoveries on secure key distribution [1], polynomial time prime factorization [2], and fast database search [3] are good examples. They take advantage of quantum mechanics to improve the efficiency of a computation process. As a result, quantum computing has become the most rapidly expanding fields of research.

As an example, unordered database search is an important and widely discussed problem. Unordered database search is important because, from an engineering point of view, many problems can be formulated as a database searching process. For instance, cracking a 1024-digit secret key in a 2^{1024} key space is essentially an unordered database searching process. Classically, the only way to search such a database is to test the elements sequentially against the condition until the target is found. For a database of size N, this brute force search requires an average of $O(N/2)$ comparisons. However, Grover's algorithm can identify the target in $O(\sqrt{N})$ steps, which is more efficient.

The main idea of Grover's algorithm is to amplify the probability amplitude of the target state so it can be found with a high probability when the final measurement is performed. More specifically, given a Boolean expression f, Grover's algorithm allows the target state to be marked by an "oracle", so the target x with $f(x) = 1$ can be found with a $O(\sqrt{N})$ complexity. It has also been shown that, when there are multiple (t) targets, the problem can still be solved in $O(\sqrt{N/t})$ even if t is not known in advance [4]. The optimality of Grover's algorithm [5] and several generalizations including searching multiple objects [6], analyzing the initial state and optimal unitary operator [7], [8], and studying the stability and robustness of Grover's algorithms [9]–[12] have also been studied. In addition, variations of this algorithm have been applied to many other problems such as state preparation [13], finding the minimum [14], element distinctness [15], and many others [16]–[18].

Implementation of Grover's algorithm is also the focus of many researches. For example, Diao *et al.* showed how to use 1-bit unitary gates and 2-bit quantum phase gates to realize Grover's algorithm in cavity QED systems [19]. Roland *et al.* analysed the differences between continuous-time analog Hamiltonian search algorithms and original discrete circuit based Grover's algorithm [20]. Xiao *et al.* described how to robust single and two qubit logic gates based on nuclear magnetic resource (NMR) technologies and use the gates to implement counting algorithm [21], which is based on Grover's algorithm. Most of the studies described above emphasized on either theoretical study or small-scale physical implementation. However, for large quantum circuits designed for a real-world application, some modifications should be made. For example, in a database query application, the demand should not only be finding the target (answering if there is a x satisfying $f(x)$) but also getting the related information (given y, answering if there is a x fulfilling $f(x) = y$). Therefore, in this paper we aim to discuss the quantum circuit design issues in a realistic way. We discuss, from the circuit design perspective, what kind of applications are suitable for Grover's algorithm and propose a circuit template for such applications. We begin by showing the main idea about how to design quantum Boolean

Manuscript received December 11, 2006; revised June 12, 2007. This work was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC 95-2221-E-002-068. This paper was recommended by Guest Editor C. Lau.

Y.-L. Ju is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C.

I.-M. Tsai is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C., and also with Chunghwa Telecommunication Laboratories, Yaoyuan 32601, Taiwan, R.O.C.,

S.-Y. Kuo is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, TaiwanR.O.C., and also with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 10617, Taiwan, R.O.C. (e-mail: sykuo@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCSI.2007.907845

circuits for the oracle and the inversion-about-average function in Grover's algorithm. We use a well-known problem of satisfiability (SAT) as an example to emphasize the generality of our circuit model [22]. Based on this circuit model, we then discuss two real-world applications that have different types of search criteria. The first application is to perform a reverse database search in a phone book. We show the quantum circuits for establishing an empty database, inserting data, and querying the database. In this process, we found that one has to traverse the entire phone book to build the database. As a result, we point out that it is not practical in a real-world scenario. In the second application, we give a detailed circuit design for breaking a symmetric cryptosystem. Our discussion is based on known plaintext attack. We show how the known plaintext, after encrypted by all possible keys, can be compared with its ciphertext.

Although the search criteria of the two applications we discussed above are quite different, both of them are actually one-way functions. In this perspective, we conclude this paper by proposing a template of quantum circuits to solve well-formulated one-way functions. As a result, we believe that this paper makes the following contributions. First, we give a detailed quantum circuit design for implementing large scale Grover's algorithm. Second, we extend our circuits to a template for solving any one-way functions.

II. Background

A. States and Gates

In quantum mechanics, the state of a single two-level quantum bit (*qubit*) can be written as a linear combination in a two-dimensional complex vector space as

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle \tag{1}$$

where $c_0, c_1 \in \mathcal{C}$ and $\sum |c_n|^2 = 1$. The state shown above exhibits a unique phenomenon in quantum mechanics called *superposition*. When you measure such a particle, the system is projected to one of its eigenstates (i.e., either $|0\rangle$ or $|1\rangle$). The probability for projecting to each state is given by the absolute square of its probability amplitude, i.e., $|c_0|^2$ and $|c_1|^2$. Obviously, the sum of $|c_0|^2$ and $|c_1|^2$ shall be 1 to satisfy the probability rule. Multiple qubits can also form a quantum system. A multi-qubit system is spanned by the basis of the tensor product of each space. For example, the joint state of qubit a and qubit b is spanned by $\{|00\rangle_{ab}, |01\rangle_{ab}, |10\rangle_{ab}, |11\rangle_{ab}\}$, i.e.

$$|\phi\rangle_{ab} = c_0|00\rangle_{ab} + c_1|01\rangle_{ab} + c_2|10\rangle_{ab} + c_3|11\rangle_{ab} \tag{2}$$

where $c_0, c_1, c_2, c_3 \in \mathcal{C}$ and $\sum |c_n|^2 = 1$.

A quantum system can be manipulated in many different ways, called *quantum gates*. An example is the quantum *Not* $(\mathbf{N})$ gate, which functions as

$$\begin{cases} \mathbf{N}(|0\rangle) = |1\rangle \\ \mathbf{N}(|1\rangle) = |0\rangle \end{cases} \tag{3}$$

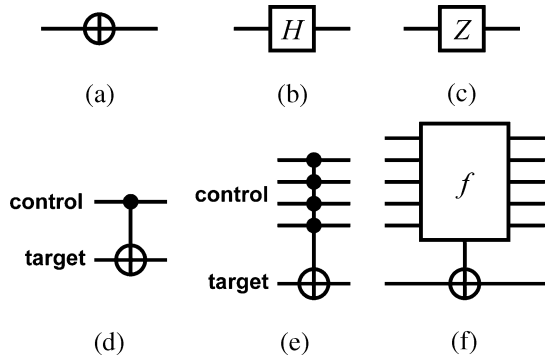


Fig. 1. Symbols for basic quantum gates.

As a result, when a qubit $\alpha|0\rangle + \beta|1\rangle$ goes through a quantum $\mathbf{N}$ gate, the state transforms into $\beta|0\rangle + \alpha|1\rangle$. The symbol of an $\mathbf{N}$ gate is shown in Fig. 1(a). Note that the horizontal line connecting the input and the output is not a physical wire, instead it represents a qubit under time evolution.

The *Hadamard* $(\mathbf{H})$ gate is an important gate. It changes the state as

$$\begin{cases} \mathbf{H}(|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \mathbf{H}(|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \tag{4}$$

and its symbol is depicted in Fig. 1(b). Another useful quantum gate which shifts the relative phase by π is called the $\mathbf{Z}$ gate. The operation can be described as

$$\begin{cases} \mathbf{Z}(|0\rangle) = |0\rangle \\ \mathbf{Z}(|1\rangle) = -|1\rangle \end{cases} \tag{5}$$

and its symbol is depicted in Fig. 1(c).

Similarly, a two-bit quantum system can be manipulated using a two-bit gate. For example, a *Control-Not* $(\mathbf{CN})$ gate performs the following transformation:

$$\begin{cases} \mathbf{CN}(|00\rangle) = |00\rangle \\ \mathbf{CN}(|01\rangle) = |01\rangle \\ \mathbf{CN}(|10\rangle) = |11\rangle \\ \mathbf{CN}(|11\rangle) = |10\rangle. \end{cases} \tag{6}$$

More specifically, a $\mathbf{CN}$ gate consists of one *control* qubit, which does not change its value, and a *target* qubit, which changes its value only if the control qubit is $|1\rangle$. The operation can be written as $\mathrm{CN}(|\mathrm{control}, \mathrm{target}\rangle) = |\mathrm{control}, \mathrm{control} \oplus \mathrm{target}\rangle$, where "$\oplus$" denotes exclusive-or (XOR). The symbol of a $\mathbf{CN}$ gate is shown in Fig. 1(d). A generalization of the $\mathbf{CN}$ gate involves multiple control qubits. For example, a $m+1$ bit $\mathrm{C}^m\mathbf{N}$ gate is defined as

$$\mathbf{C}^m\mathbf{N}(|x_1, \cdots, x_m, y\rangle) = |x_1, \cdots, x_m, (\wedge_{i=1}^m x_i) \oplus y\rangle. \tag{7}$$

This gate inverts the target qubit if all the control qubits are $|1\rangle$'s. When $m = 1$ this gate reduces to the $\mathbf{CN}$ gate described before. The symbol of a generalized $\mathbf{C}^m\mathbf{N}$ gate is shown in Fig. 1(e).

Further generalization can be made if the inversion of the target is controlled by evaluating the control qubits using a given Boolean function. In other words, the target is inverted if the

control qubits satisfy a given Boolean expression, otherwise it is unchanged. The Boolean function can be as simple as evaluating whether the control qubits are all $|1\rangle$'s (or $|0\rangle$'s), or a complicated Boolean expression which is satisfied by more than one Boolean variable assignment. The symbol of such an *f-Control-Not* gate ($\mathbf{f-CN}$) is shown in Fig. 1(f). Note that, a special case of $\mathbf{f-CN}$ gates is to invert the target qubit when some of the control bits are $|1\rangle$'s (indicated by black dots) and some are $|0\rangle$'s (indicated by white circles).

It is interesting that if the target is initially set as $(1/\sqrt{2})(|0\rangle - |1\rangle)$, applying an $\mathbf{f-CN}$ gate results in an *eigenvalue kickback*, which causes a π phase shift on some components of the input state. This phenomenon is described as follows.

Without loss of generality, we assume the quantum state of the control qubits is

$$|\psi_0\rangle = \sum_{i\in A} c_i|i\rangle + \sum_{i\in S-A} c_i|i\rangle \tag{8}$$

where A is the set of satisfied assignments and S-A is the set of un-satisfied assignments, i.e., $S = \{0, 1, \ldots, 2^{n-1}\}$ and $A = \{i | f(i) = 1, i \in S\}$.

Due to the linearity of quantum mechanics, we have

$$\begin{aligned}
&\mathbf{f-CN}\left(\left(\sum_{i\in A} c_i|i\rangle + \sum_{i\in S-A} c_i|i\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)\right) \\
&= \left(\sum_{i\in A} c_i|i\rangle\right) \otimes \mathbf{N}\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) \\
&\quad + \left(\sum_{i\in S-A} c_i|i\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= \left(\sum_{i\in A} c_i|i\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}(|1\rangle - |0\rangle)\right) \\
&\quad + \left(\sum_{i\in S-A} c_i|i\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= \left(-\sum_{i\in A} c_i|i\rangle + \sum_{i\in S-A} c_i|i\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)
\end{aligned} \tag{9}$$

which effectively inverts the phase of those satisfied assignments, i.e., $i \in A$. This mechanism is used as the basic function in our circuits.

B. Grover's Algorithm

The main idea of Grover's algorithm is to first prepare a register in a superposition state with the probability amplitudes uniformly distributed. Then, *selectively invert* the target and perform an *inversion-about-average* operation. The selective-inversion followed by an inversion-about-average will cause amplification on the probability amplitude of the target. After $O(\sqrt{N})$ such iterations, we have a high probability of getting the target upon a final measurement.

As we can see, the only request in Grover's algorithm is the information regarding whether an item is the target. This is also known as an *oracle*. If we label the items in a database with the integers $0, 1, 2, \ldots, 2^n - 1$ and denote the label of the unknown marked record by x_0, the oracle is an n-bit binary function

$$f : \{0,1\}^n \longrightarrow \{0,1\} \tag{10}$$

defined by

$$f(x) = \begin{cases} 1, & \text{if } x = x_0 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Note that, as a standard oracle, we have no access to the internal structure of the function f. It operates transparently in Grover's algorithm as a black-box function, which we can query as many times as we like.

Using the oracle, the selective-inversion $I_{|x_0\rangle}$ can be defined as the following unitary function:

$$I_{|x_0\rangle}(|x\rangle) = (-1)^{f(x)}|x\rangle = \begin{cases} -|x_0\rangle, & \text{if } x = x_0 \\ |x\rangle, & \text{otherwise.} \end{cases} \tag{12}$$

This function does an inversion on the input if it is the target, while leaving all other cases unchanged. Due to the linearity of quantum mechanics, all items in the database can be processed simultaneously by applying $I_{|x_0\rangle}$ to the superposition state of all items.

Obviously, the operation described above can be written as

$$I_{|x_0\rangle} = I - 2|x_0\rangle\langle x_0| \tag{13}$$

and it is actually an inversion in $\mathcal{H}$ about the hyperplane perpendicular to $|x_0\rangle$. Following the same notation, if we define

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \tag{14}$$

the inversion-about-average operation can be regarded as an inversion in $\mathcal{H}$ about the hyperplane perpendicular to $|\psi_0^{\perp}\rangle$ [23], and

$$I_{|\psi_0^{\perp}\rangle} = -I_{|\psi_0\rangle}. \tag{15}$$

As a result, the operation of a selective-inversion followed by an inversion-about-average can be written as

$$G = -I_{|\psi_0\rangle} I_{|x_0\rangle}. \tag{16}$$

Based on this operator, Grover's algorithm can find the target with a high probability (approaching 1) by applying the operator G to the superposition state

$$|\psi_0\rangle = \mathbf{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \tag{17}$$

for $O(\sqrt{N})$ times before the final measurement.

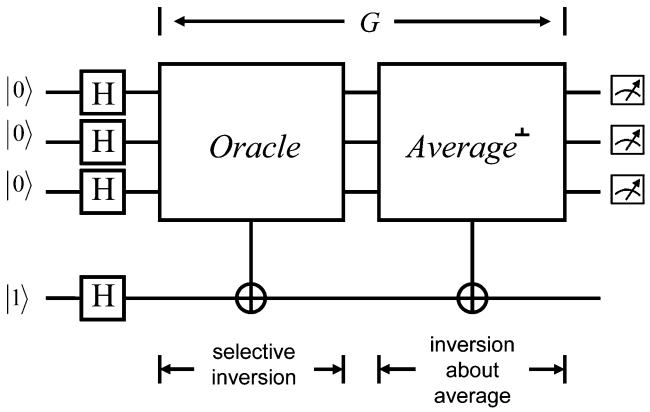


Fig. 2. Circuit block diagram of Grover's search algorithm.

III. Circuit Block Diagram

As described before, the implementation of the operator G consists of two parts: selective-inversion $I_{|x_0\rangle}$, and inversion-about-average $I_{|\psi_0^{\perp}\rangle}$. The eigenvalue kickback is used in the construction of these two parts, as shown in Fig. 2.

The initial qubits in Fig. 2 include n qubits prepared in the ground state $|\psi\rangle = |0\rangle^{\otimes n}$ and one auxiliary qubit in the excited state $|\phi\rangle = |1\rangle$. This can be written as

$$|\psi\rangle \otimes |\phi\rangle = |0\rangle^{\otimes n} \otimes |1\rangle. \tag{18}$$

After the Hadamard transform $\mathbf{H}^{\otimes n}$ and $\mathbf{H}$, all possible states are superposed as $|\psi_0\rangle \otimes |\phi_0\rangle$, where

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \tag{19}$$

$$|\phi_0\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \tag{20}$$

A. Selective-Inversion

The first part of the circuits is the *oracle-Control-Not* ($\mathbf{o-CN}$), as shown in Fig. 2. Theoretically, the oracle is only a function that checks whether a specific item is the target. However, due to the linearity of quantum mechanics, when the oracle is applied to the superposition state $|\psi_0\rangle$, all possible items are examined against the criteria. It follows from the fact that the eigenvalue of $(1/\sqrt{2})(|0\rangle - |1\rangle)$, i.e., -1, is kicked back to the target state, so the $\mathbf{o-CN}$ effectively inverts the target, as

$$\mathbf{o-CN}\,(|\psi_0\rangle \otimes |\phi_0\rangle) = \left(I_{|x_0\rangle}\,(|\psi_0\rangle)\right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{21}$$

In other words, the $\mathbf{o-CN}$ circuits are computationally equivalent to the operator $I_{|x_0\rangle}$, which inverts the unknown target in one single operation. Since most practical engineering problems can be binary encoded, searching the target in a database is the same thing as finding a satisfiable solution for a binary Boolean expression. This means the oracle function can be represented by a $n \times 1$ Boolean logic.

There are two ways to implement such a Boolean logic. The straightforward method uses elementary gates to simulate classical gates including **AND**, **OR**, and **NOT**, as shown in Fig. 3. Note that, unlike classical circuits, the **FANOUT** function must be done explicitly by a quantum **CN** gate, instead of a metallic contact.

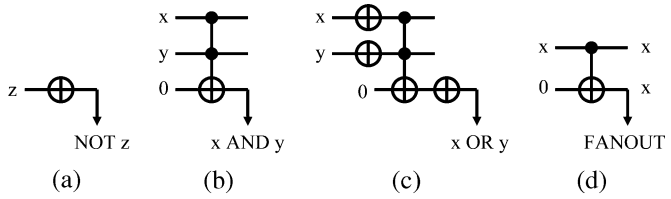


Fig. 3. The straightforward implementation of classical (a) NOT, (b) AND, (c) OR, (d) FANOUT gates.

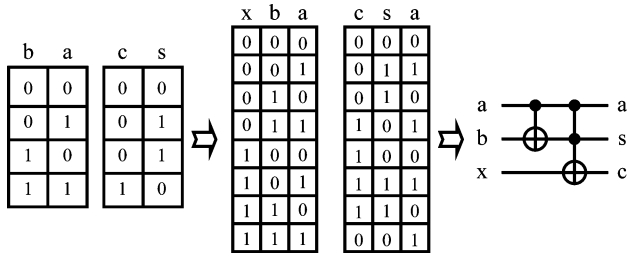


Fig. 4. Minimum space implementation of a half adder.

An alternative way that achieves the minimum space consumption is based on the concept of permutation. Since an n-bit quantum Boolean operation is a permutation on the set of all n-bit binary patterns, the desired Boolean function can be achieved on a subset of the qubits [24]. For example, the circuits for a half adder are shown in Fig. 4. However, the construction of the permutation requires a full listing of the truth table, which is as hard as finding the target. As a result, the first method has to be used to construct the oracle circuits.

B. Inversion-About-Average

The second part of the operation G is the circuit implementing the function of inversion-about-average. When the operation is applied to a superposition state, it actually keeps the component in the $|\psi_0\rangle$ direction unchanged, while inverting the components in dimensions that are perpendicular to $|\psi_0\rangle$. This can be represented as

$$I_{|\psi_0^{\perp}\rangle} = -I_{|\psi_0\rangle} \tag{22}$$

where

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \tag{23}$$

represents the *average*. Using the same concept of eigenvalue kickback, the component in the $|\psi_0^{\perp}\rangle$ dimension can be selectively inverted, as shown in Fig. 2. Note that this part of the circuits does not depend on the search criteria.

IV. Searching a SAT Solution

The SAT problem is a well-known problem in theoretical computer science. Given a Boolean expression in conjunctive normal form, the problem of deciding whether this expression has an assignment that satisfies the formula has been shown to be an NP-complete problem [25]. In this section, we will show

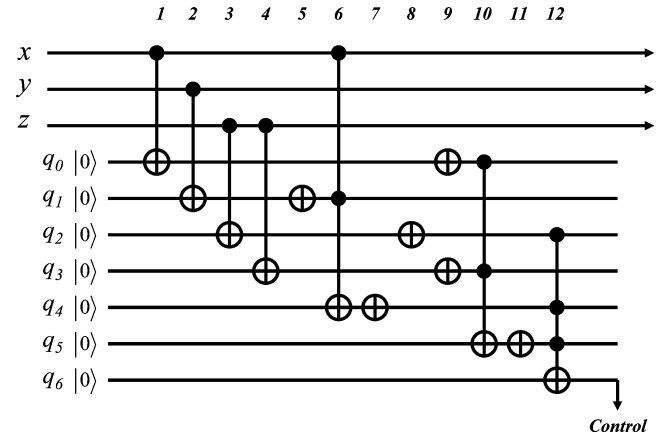


Fig. 5. Oracle circuits for a single-target search.

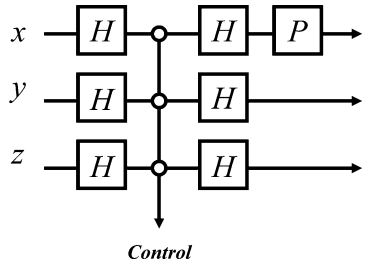


Fig. 6. The circuits of inversion-about-average.

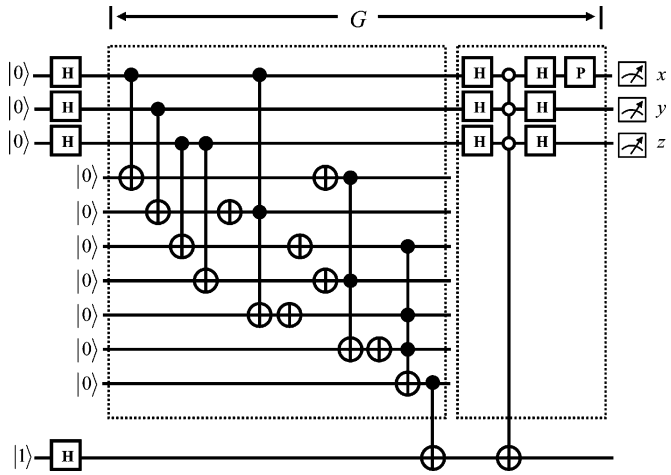


Fig. 7. Circuits G in the quantum search algorithm.

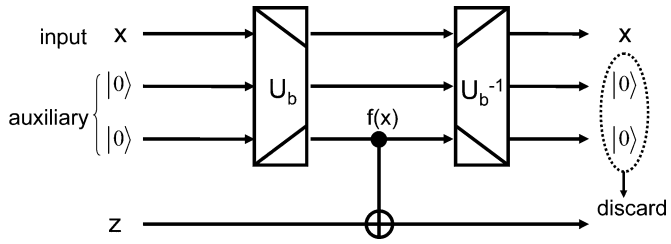


Fig. 8. Template for recovering the auxiliary qubits.

how Grover's algorithm can be used to find a satisfiable solution for the SAT problem. Without loss of generality, we use the binary expression

$$(\bar{x} \vee y) \wedge \bar{z} \wedge (x \vee z) \tag{24}$$

as an example. As described before, the circuits of the oracle can be implemented according to the Boolean expression itself, which is shown in Fig. 5. In Fig. 5, gate 1, 2, 3, and 4 copy the input x, y, and z to q_0, q_1, q_2, and q_3, so subsequent operations do not affect the original input. Gate 5, 6, and 7 implement the function $(\bar{x} \vee y)$, gate 8 inverts z, gate 9, 10, and 11 implement the function $(x \vee z)$, and finally gate 12 performs the **AND** function for these three clauses. The final result is in qubit q_6, which is the control qubit for the selective inversion $(\mathbf{o-CN})$ in Fig. 2. Note that although the three-input **AND** function can be implemented by cascading two two-input **AND** gates, it is shown as one $\mathbf{C^3N}$ gate for simplicity.

The second part of the circuits performs the function of inversion-about-average. To invert the component in the $|\psi_0\rangle$ dimension, Hadamard gates are used to transform $|\psi_0\rangle$ to $|0\rangle$, then the state $|0\rangle$ is selectively inverted. After the inversion, the state is transformed back by another set of Hadamard gates. The circuits can be represented by

$$-I_{|\psi_0\rangle} = -HI_{|0\rangle}H \tag{25}$$

as shown in Fig. 6.

Note that the function of $I_{|0\rangle}$ is implemented by an eigenvalue kickback. It is shown in Fig. 6 as a generalized $\mathbf{C}^m\mathbf{N}$ gate which inverts the target when all the control qubits are $|0\rangle$'s. This is shown as white circles, instead of black dots, in Fig. 6. Although it does not affect the measurement, the minus sign in $-I_{|\psi_0\rangle}$ can be implemented by applying a phase of π to any one of the qubits. This is shown as the P transform in Fig. 6. Note that since this part of the circuits is completely independent of the search criterion, it can be used to do any single- or multi-target search. The complete circuit design for G is shown in Fig. 7. After applying $O(\sqrt{N})$ times of G, a final measurement on qubit x, y, and z will reveal the target with high probability.

Although the circuits can be used to find the target, the auxiliary qubits must be restored and discarded to avoid the accumulation of qubits and to reduce the extra efforts needed for preserving the auxiliary qubits from noise. A general circuit block diagram for removing auxiliary qubits is shown in Fig. 8.

The operation U_b in the figure calculates the result $f(x)$ with the help of some auxiliary qubits. Then the qubit representing the result $f(x)$ is used as the control qubit of a $\mathbf{CN}$ gate to save the state in z. Since quantum operations are reversible by nature, the auxiliary qubits can be restored back to their initial states by simply applying the reverse operation U_b^{-1}, then they can be discarded safely. In Fig. 9, we give a simple example showing how to implement $\bar{x}y$ with the help of auxiliary qubits and how to restore and discard the auxiliary qubits safely.

With the principle described above, auxiliary qubits in Fig. 7 can be removed by applying the gates shown in Fig. 10. Since this part does not affect the result, we will skip the recovering circuits for the rest of this paper in order to make our discussion clear.

The circuit design for a single-object search can be easily generalized to search multiple objects. The only difference is

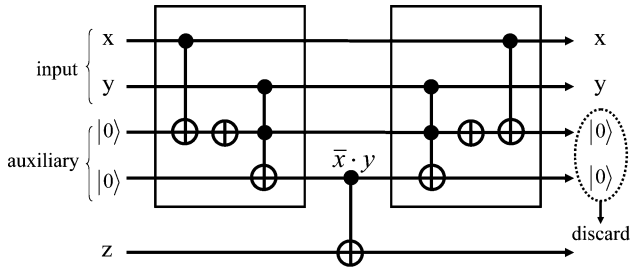


Fig. 9. Example circuits for recovering the auxiliary qubits.

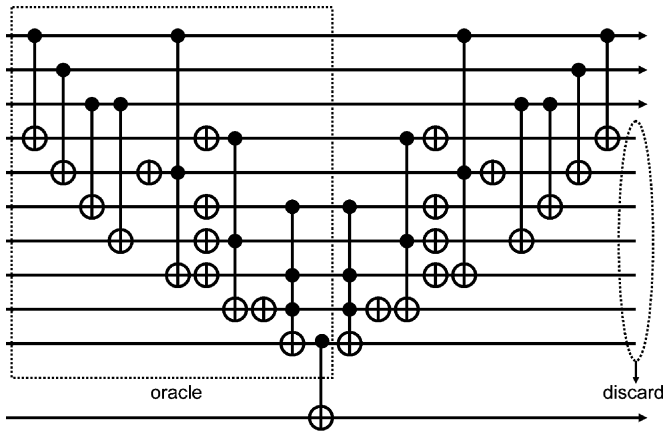


Fig. 10. Circuits in the reverse order to discard the auxiliary qubits.

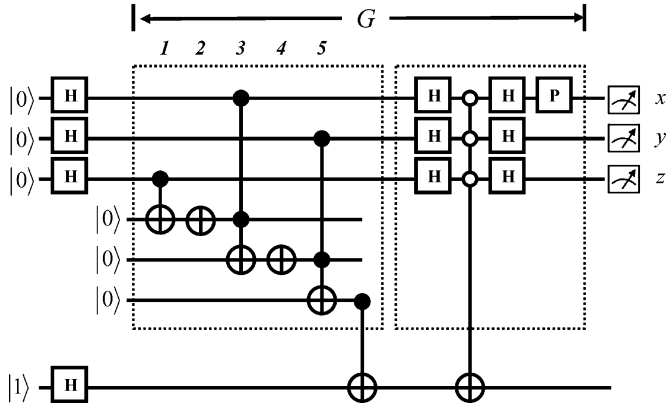


Fig. 11. Circuits for searching the first target.

that, after each target is found, the oracle has to be modified to exclude this target. A typical scenario is shown in the following example.

Assuming the problem is binary encoded as

$$(\bar{x} \vee z) \wedge y \tag{26}$$

the oracle for searching the first target is shown in Fig. 11. In these circuits, gate 1 copies the variable z, gate 2, 3, and 4 implement the function $(\bar{x} \vee z)$, and finally gate 5 performs the **AND** function for these two clauses. Then, the $\mathbf{o-CN}$ circuits followed by the same circuits in Fig. 6 implement the operation G for the first target. A final measurement after $O(\sqrt{N})$ times of G gives one of the targets with equal probability.

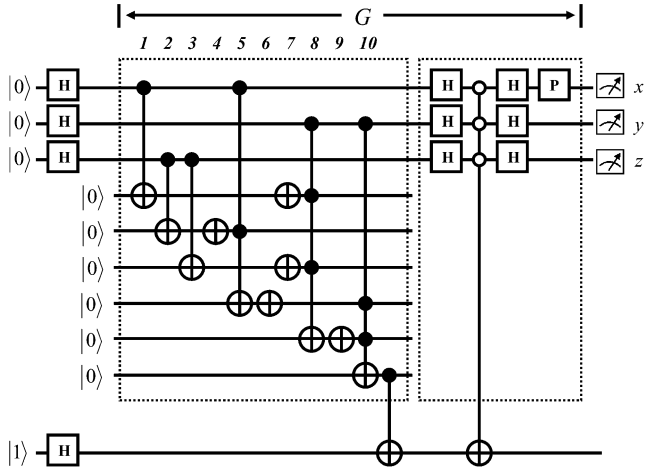


Fig. 12. Circuits for searching the second target.

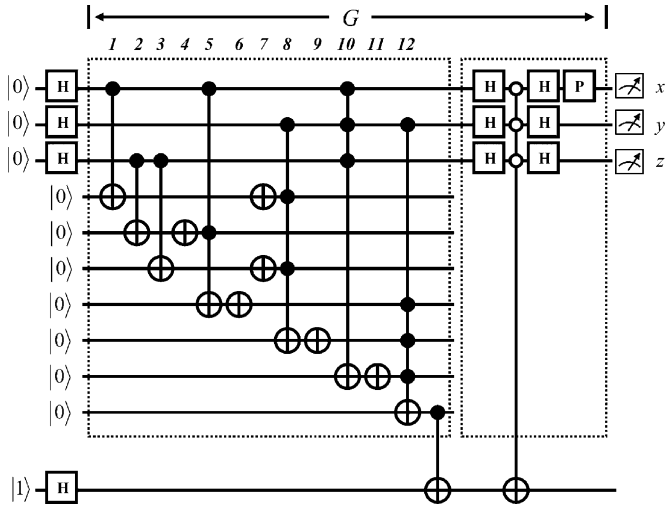


Fig. 13. Circuits for searching the third target.

After the first measurement, the same process is repeated again, the only difference is that the oracle is modified to exclude the target that has been found. Assuming the first result is $x = 0$, $y = 1$, $z = 0$, the oracle is now modified as

$$(\bar{x} \vee z) \wedge y \wedge \overline{\bar{x} \cdot y \cdot \bar{z}}. \tag{27}$$

The clause $\overline{\bar{x} \cdot y \cdot \bar{z}}$ is used to exclude the target $x = 0$, $y = 1$, and $z = 0$. The new oracle is shown in Fig. 12. In these circuits, gate 7, 8, and 9 are used to implement $\overline{\bar{x} \cdot y \cdot \bar{z}}$. Then, gate 10 gives the oracle's result. This new oracle and the same circuits of inversion-about-average are used to find the second target.

Similarly, assuming the result of the second measurement is $x = 1$, $y = 1$, and $z = 1$, the third oracle is then changed to be

$$(\bar{x} \vee z) \wedge y \wedge \overline{\bar{x} \cdot y \cdot \bar{z}} \wedge \overline{x \cdot y \cdot z}. \tag{28}$$

The new clause $\overline{x \cdot y \cdot z}$ is added to exclude the second target. The oracle is shown in Fig. 13. Again, gates 10 and 11 are used to implement the new criterion $\overline{x \cdot y \cdot z}$, gate 12 gives the result

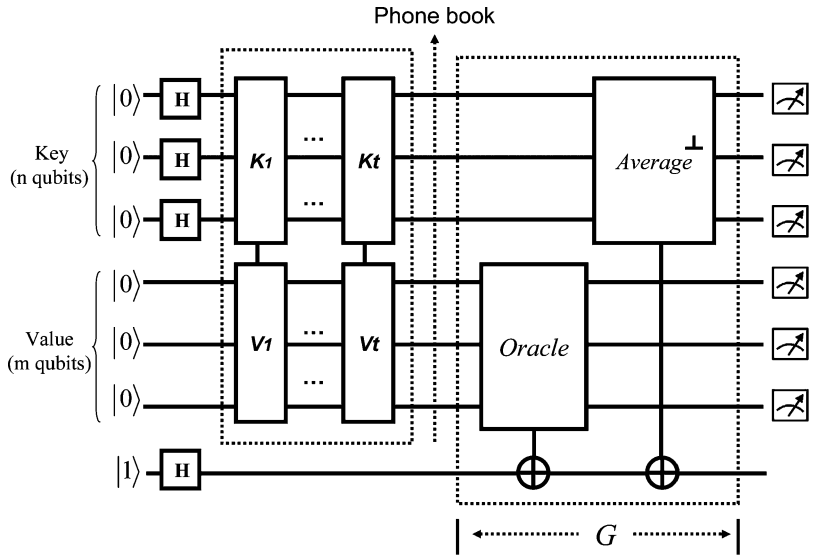


Fig. 14. Quantum circuits for searching a phone book.

of the new oracle. This new oracle, together with the circuits of inversion-about-average, is used to find the third target.

V. Searching in a Phone Book

In this section, we give the circuits for applying the Grover's algorithm to a phone-book-like database searching problem. Although this is a well-known example of Grover's algorithm, it is not so practical from a circuit point of view. We will give more discussions about the drawbacks at the end of this section. The scenario of the "phone book searching" problem is described as follows. Imagine that one day you miss a phone call, but fortunately you have caller ID and the phone number is recorded in the phone. Based on the number and a phone book, you decide to find who was calling you. Since a classical phone book is sorted by name, there is no easy way to identify the caller except to browse through all the entries until you find the number. Assuming the size of the phone book is N, the number of entries you have to search is on average $O(N/2)$, while in the worst case you would have to search all the entries (which could be thousands or millions of entries). In addition to finding the phone number, we also want to retrieve other associated information about the number (i.e., name and address). Although it is often cited that Grover's algorithm can help identify the phone number with only $O(\sqrt{(N)})$ queries, they are usually mentioned without further explanation or implementation. In fact, with a little more consideration, the problem inherently has some drawbacks in realizing the circuits and is actually not a good example. In the following we will show the detailed circuits about how a phone book is encoded and why such application is not practical.

The structure of each entry in a phone book is naturally a pair of (key, value). Classically, the key is the name while the value includes other associated information such as the number and/or address. Without loss of generality, we assume that there are no duplicate names in the phone book, and each name (the key) is encoded as an n-bit binary string so they can be represented as $0, 1, \ldots .2^{n-1}$. This means the number of entries in the phone book is $N = 2^n$. Similarly, the numbers (the value) are encoded as m-bit binary strings. As a result, in a *quantum phone book*, all the entries are superposed like

$$\sum_{\text{key}} |\text{key}\rangle|\text{value}\rangle \tag{29}$$

which is equivalent to

$$\sum_{\text{name}} |\text{name}\rangle|\text{number}\rangle. \tag{30}$$

The circuits for encoding a phone book are shown in Fig. 14. It consists of two parts. The first part is for establishing an empty phone book while the second part is for inserting data into the phone book.

The qubits in a quantum phone book are separated into two groups (n key-qubits and m value-qubits). Initially all the key qubits and value qubits are in the $|0\rangle$ state. This can be written as

$$|\Psi\rangle_{\text{init}} = |0\rangle^{\otimes n}|0\rangle^{\otimes m}. \tag{31}$$

By applying the Hadamard gates, the state of the key-qubits are transformed into a *uniform* superposition of all possible states $(|0\rangle, |1\rangle, \ldots, |2^{n-1}\rangle)$. The resulting state is

$$H^{\otimes n} I^{\otimes m} \left(|0\rangle^{\otimes n}|0\rangle^{\otimes m}\right) = \frac{1}{\sqrt{2^n}} \sum_{key=0}^{2^n-1} |\text{key}\rangle|0\rangle^{\otimes \text{m}}. \tag{32}$$

Note that an entry with its $|\text{number}\rangle$ as $|0\rangle$ indicates an invalid entry. As a result, (32) is an empty phone book with keys (names) ranging from $|0\rangle$ to $|2^{n-1}\rangle$. With an empty quantum phone book, the data can be inserted with a *key-control-value* gate, as shown in Fig. 15(a). The key-control-value gate includes a key gate and a value gate, each with input qubits $|x\rangle$ and $|y\rangle$. If the key gate with input state $|x\rangle$ evaluates true, the value gate will be applied on the $|y\rangle$ qubits. Otherwise, it does nothing. A sample key-control-value gate is shown in Fig. 15(b). In this example, if the input state $|x_1x_2x_3\rangle$ satisfies $\bar{x_1}x_2\bar{x_3}$ (which

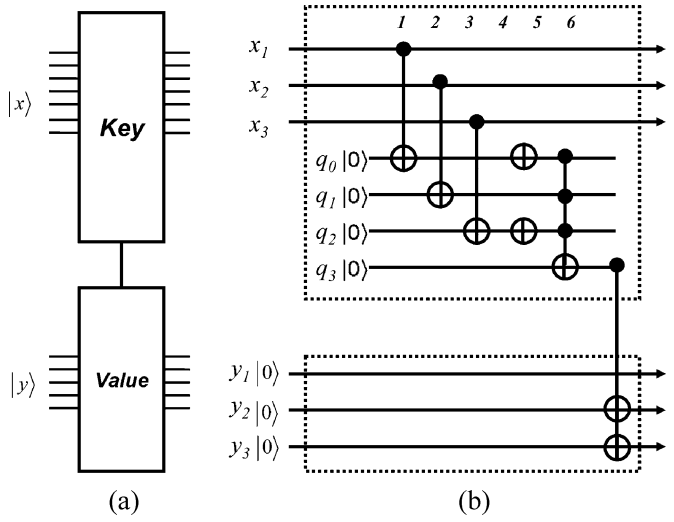


Fig. 15. (a) Key-control-value gate. (b) Implementation example.

means $x_1 = 1$, $x_2 = 1$, $x_3 = 0$), then it changes $|y_1y_2y_3\rangle$ from $|000\rangle$ to $|011\rangle$. Otherwise it does nothing. In other words, the key gate is a simple comparison gate, which outputs true if the input state equals to a specific value and otherwise it outputs false. And if the key gate outputs true, the value gate is applied on the values qubits, which functions as storing the value into the value qubits. Therefore, if we pass the empty quantum phone book $(1/\sqrt{2^n})\sum_{key=0}^{2^n-1}|key\rangle|0\rangle^m$ through key-control-value gates, each gate will record one key(name) with the corresponding value(number) into the phone book.

After inserting all the data, the quantum phone book becomes

$$\frac{1}{\sqrt{2^n}}\sum_{key=0}^{2^n-1}|\text{key}\rangle|\text{value}\rangle \tag{33}$$

with all entries in a uniform distribution. The description above only shows how a quantum phone book is encoded. It has nothing to do with the searching process. The searching process is described in the following paragraphs.

As illustrated, the effect of the $\mathbf{o-CN}$ gate is to apply a phase of π if the input state is the target (i.e., the state satisfies the oracle). It is noteworthy that, the oracle can be applied to selected groups of qubits according to a specific query condition. For example, assuming the query is to find a given phone number (which is much harder for a classical computer), the oracle circuits can be applied on the "value-qubits" only, as illustrated in Fig. 14. For each entry state $|key\rangle|value\rangle$, it only checks if the "value-qubits" are equal to the query string and flips the sign of its amplitude accordingly. Note that the same mechanism can also be used to query the "key-qubits" with equal cost in complexity. As a matter of fact, the database can be designed as $\sum|name\rangle|number\rangle|address\rangle$, in which case it can be queried either by name, number, or address (or any combination) and get all other information after applying $O(\sqrt{N})$ times of G. The only trick is to apply an appropriate oracle on appropriate qubits. This means, unlike in a classical phone book, searching a phone book by either name or number is equal in terms of computation complexity.

Following that, the *inversion-about-average* operation is applied on the key-qubits as usual. The inversion-about-average operation must be applied on the key-qubits since they compose the superposition state. So the operation will transform the state

$$\sum_{k=0}^{2^n-1}a_k|k\rangle|v\rangle \tag{34}$$

to

$$\sum_{k=0}^{2^n-1}(2\bar{A}-a_k)|k\rangle|v\rangle \tag{35}$$

where $\bar{A}=\sum_{k=0}^{2^n-1}a_k/N$ and k, v are key and value respectively. The query process is applied $O(\sqrt{N})$ times to enhance the probability of getting the target. A final measurement will reveal the target with a high probability. Thus, we can find the target information (number) and get other associated information such as name and address at the same time.

By applying the Grover's algorithm, the query process can be speed-up without problem. However, the solution has several drawbacks. The first and the most obvious one is that the database has to be built before the search process, and it takes $O(N)$ steps to build the database. Once the database is built, it means that the entire database has been traversed once, and any searching problem can be done in constant time. The second problem is that the database would have to be completely reconstructed after each query, because each query is ended with a measurement and that will destroy the whole quantum database due to the collapse of quantum state. Another problem is that, in our quantum circuit design methodology, we say that the query condition can be applied on any field of the data (name, number, or even a subset of a name), which is an improvement in comparison to the classical way. The query condition can be decided until the query stage, with no influence on the query complexity. However, a classical database can also achieve this by making indexes on all combination of the fields before querying after database construction. (Though it would be a heavy loading for if the data is stored in N bits, it would have to be make $O(2^N)$ kinds of indexes.)

VI. Breaking Symmetric Cryptosystems

A classical phone book can be regarded as a function

$$\text{Phonebook(name)} = \text{number} \tag{36}$$

which takes a name as the input and a corresponding number as the output. This function is a *one-way* function since it is easy to find the number for a given name, but it is hard to find a name for a given number. However, the circuits we illustrated in the previous section can help find the corresponding name for a given number (which is the *reverse* function) more efficiently. Enlightened by this fact, we find this method can be used to break some cryptosystems that rely on one-way functions. In this section, we use a DES-like symmetric cryptosystem as an example to show how this can be done with quantum circuits.

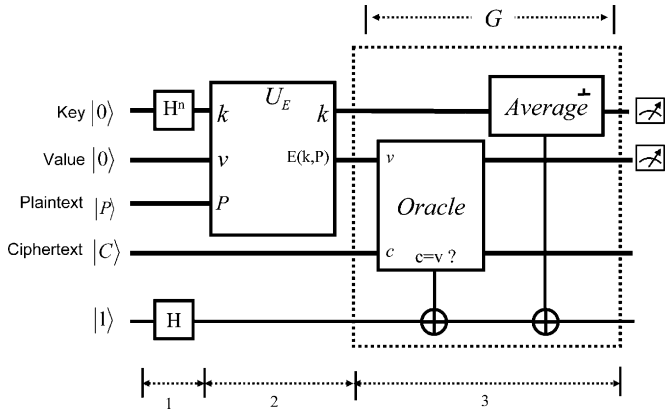


Fig. 16. Quantum circuits for attacking a symmetric encryption algorithm.

To attack a symmetric cryptosystem, the most straightforward way is to try all possible bit patterns with brute force until the correct key is identified. In order to identify the correct key, methods like "known plaintext attack" and "chosen plaintext attack" are used in case the attacker has access to some (plaintext, ciphertext) pairs. This is possible because some plaintext has special formats (like the header of an e-mail), or the attacker can feed plaintext of his choice to the encryption device and get the corresponding ciphertext. The only unknown part is the encryption key. If brute force attack is used, the exponential key space is a problem for a classical computer. This is because commands are *serially* processed in a classical computer. With quantum parallelism, we will show how quantum circuits can help attack a symmetric cryptosystem more efficiently.

A general symmetric cryptosystem takes the key and a plaintext as the input and outputs one ciphertext as

$$E(k,P)=C \tag{37}$$

where P and C stands for plaintext and ciphertext respectively. Given encryption algorithm E, one plaintext P, and the corresponding ciphertext C, the problem is to find the key k such that (37) is met.

The quantum circuits for such an exhaustive search are illustrated in Fig. 16. As shown in the figure, there are four input registers, *key*, *value*, *plaintext* and *ciphertext*. If the encryption algorithm uses a 56-bit encryption key, there would be 56 qubits in the key register ($n=56$). At the initializing stage, the key register is initialized to a superposition of all possible key states by Hadamard transformations, while the qubits in the value register are all initialized to $|0\rangle$'s. The other two registers store the plaintext P and the ciphertext C.

In the second stage, the encryption algorithm E is represented by a unitary gate U_E, with the function

$$U_E\left(|k\rangle|v\rangle|P\rangle\right)=|k\rangle\,|v\oplus E(k,P)\rangle\,|P\rangle. \tag{38}$$

The state of the qubits before the unitary gate is $\sum_k |k\rangle|0\rangle|P\rangle$, which represents a superposition of all possible keys, each with a corresponding value as $|0\rangle$. After applying the U_E operation, the state would then be transformed into $\sum_k |k\rangle|E(k,P)\rangle|P\rangle$. In other words, with a single operation, we get all possible (key, ciphertext) pairs.

In the third stage, we use Grover's operation to find the one that matches the real ciphertext (C) from all possible cases $(\sum_k E(k,P))$. For these matches, the corresponding key is the answer. The oracle simply compares the two inputs. It outputs true when they are the same, otherwise it outputs false. Then the inversion-about-average follows as usual to amplify the probability amplitude of the target. Therefore, after $O(\sqrt{N})$ Grover's iterations, a final measurement gives the key that satisfies $E(k,P)=C$.

It is noteworthy that the probability of successfully finding the correct target does not increase with the number of iterations. Furthermore, as denoted in [4], the precise number of iterations of Grover's operation depends on the number of solutions. In other words, the precise number of iterations with multiple solutions is different from that with only one solution. In Fig. 16 we assume that there is one and only one target, so we use one (plaintext, ciphertext) pair to check the validity of the target key. However, there might be another key that also satisfies (37). Therefore, the more (plaintext, ciphertext) pairs are used, the more precise we can determine the correct key. With this consideration, we illustrate the circuits for multiple known (plaintext, ciphertext) pairs in Fig. 17. In this example, (P_1,C_1) and (P_2,C_2) are two known pairs, two unitary encryption operations are used and the oracle has to test whether both known pairs are satisfied, i.e.,

$$E(k,P_1)=C_1,\ \text{and}\ E(k,P_2)=C_2. \tag{39}$$

This eliminates the ambiguity when multiple keys satisfying (37) and enhances the efficiency.

VII. Implementation and Analysis

Physical realization of a scalable quantum computer is a challenge [26]–[28]. One of the primary difficulties is how to keep the components of the computer in a coherent state, while still allowing initialization, control, and measurement. David DiVincenzo [27] has listed some criteria for building a practical quantum computer. Different technologies including silicon-based solutions [29]–[31], polarized photons [32], [33], and nuclear magnetic resonance (NMR) [34]–[36] have also been experimentally demonstrated recently. Taking NMR as an example, information is carried by nuclei in molecules and Radio Frequency (RF) pulses that manipulate nuclear spin orientations are used to implement a quantum operation. The effect of a RF pulse depends on three factors: the power level, the duration, and the direction. For example, a single-qubit gate R_x which performs $\pi/2$ rotations about $\hat{x}$ can be implemented with RF pulses along $\hat{x}$ in the rotating frame for a specific time under a predefined power level. Besides, a CN gate can be implemented by a free evolution time of $(1/2)J$, where J is the coupling constant [37]. Since the set of single-qubit rotation and CN gate is universal [38], [39], these two gates are sufficient to implement any quantum circuits. In the following paragraphs, we use the DES cracking problem as an example

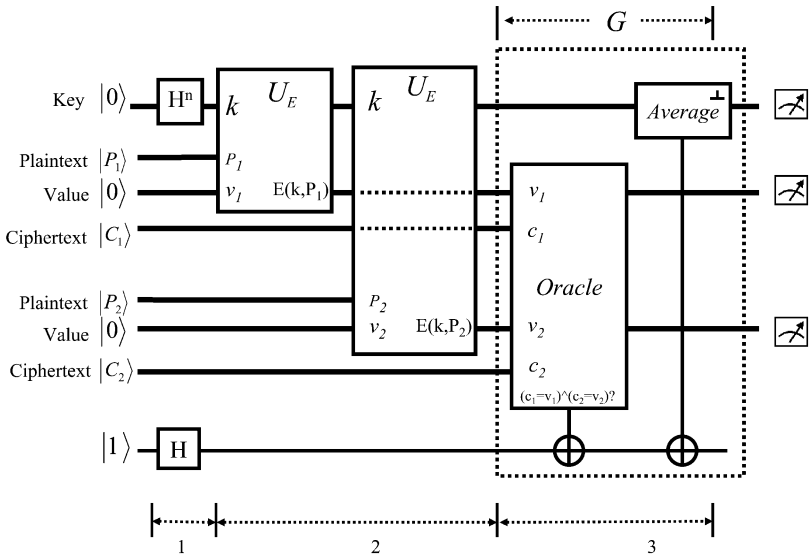


Fig. 17. Attacking a symmetric cryptosystem with multiple known (P,C) pairs.

to compare the performance of quantum circuits with classical computers.

The DES algorithm [40] enciphers a 64-bit plaintext by a 64-bit key, of which 56 bits as effective key and 8 bits for parity checking. The enciphering consists of three stages: an initial permutation IP, 16 rounds of key-dependent computation, and finally a permutation IP^{-1}, which is the inverse of the initial permutation. It is noteworthy that the permutation for classical bits and the key scheduling process correspond to nothing but changing the order of qubits and hence no quantum gate has to be applied. As a result, we have to deal with only the ciphering function. The ciphering function $f(R, K)$ used in this algorithm has four stages: expansion, key mixing, substitution, and permutation.

1) Expansion: The input R is expanded from 32 bits to 48 bits by linear permutation and duplicating some bits according to a predefined table. The permutation takes no quantum gate and the duplication process can be performed by $48-32(=16)$ CN gates. Since the 16 CN gates can be done at the same time, expansion can be accomplished in one (quantum) gate delay.
2) Key mixing: Perform XOR on the generated 48 bits and a subkey. All the XOR operations can be done by $\mathbf{CN}$ gates at the same time and therefore be executed in one (quantum) gate delay.
3) Substitution: The output is then divided into 8 blocks of 6 bits each. Each block goes through its selection function, which uses a table to specify the mapping from the 6-bit value to its corresponding 4-bit value. In quantum circuits, each table can be performed using generalized $\mathbf{f-CN}$ gates [like the key-value gate in Fig. 15(a)]. Since these operations are applied sequentially, it takes 2^6 quantum gates to accomplish the substitution.
4) Permutation: A final permutation is applied. However, as before, no quantum gate is needed in this process.

In summary, the total number of quantum gates in the ciphering function is $1+1+2^6(=66)$. The ciphering function is then used in the key-dependent part of the algorithm, which includes 16 rounds of identical computation (under control of 16 subkeys). Assuming the 64-bit input of each round is represented as LR[L is the left half block (32 bits) and R is the right half block (32 bits)], each round passes R and a subkey K through the ciphering function f, then performs an XOR on the output of f and L. The XOR result becomes the new R' and the original R becomes the new L'. The new cascaded $L'R'$ is the input to the next round. In this stage, the exchange of LR takes no quantum gate as discussed before and the XOR operations can be performed by CN gates. Since the XOR operations are performed on each qubit in parallel, each round takes 66 (ciphering function) plus 1 (XOR) quantum gates. As a result, this key-dependent part takes a total of $16*(66+1)=1072$ quantum gates (delay).

Other quantum gates involved in searching the target include 1 level of Hadamard gates to prepare the initial superposition and some quantum gates for Grover's iteration (oracle and inversion-about-average). The oracle in Grover's operation simply compares the two registers by performing XOR on each qubit and then uses a generalized C^mN gate to check if these two registers are the same. Since the XOR on each qubit can be performed in parallel, this process totally costs 2 quantum gates (delay). As to the circuits for inversion-about-average, 4 quantum gates (delay) are needed, as illustrated in Fig. 6. Since the oracle and the inversion-about-average are iterated for $(\pi/4)\sqrt{N}$ times, there will be a total of $(\pi/4)\sqrt{2^{56}}*(2+4)$ gates delay. Overall, it takes $1+1072+(\pi/4)\sqrt{2^{56}}*(2+4)$ quantum gates in terms of duration.

Taking one experiment we have performed on a Bruker Avance DMX-500-MHz NMR system [36] as an example. With a quantum computer which uses ${}^1H$ atom in carbon-13 labelled chloroform $({}^{13}CHC13)$ as the information carrier and sets the power of the pulses at 3.00 dB, the duration for a $\pi/2$ pulse is 9.5 μs. In this scale, it takes about 3.3 h to crack a DES key. Compared with a classical technology record [41], in which a DES key is cracked in 22 h and 15 min (by Distributed.Net [42], Electronic Frontier Foundation's DES Cracker [43], and nearly 100 000 computers on the Internet),

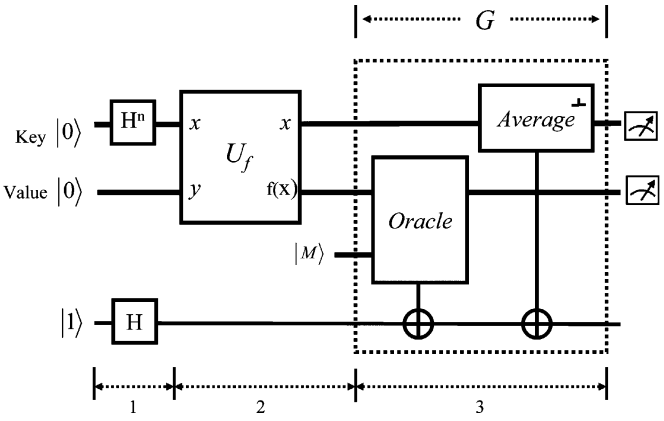


Fig. 18. Template quantum circuits for a general exhaustive search.

the proposed quantum circuits outperform the classical solution in a significant way. It is worth noting that this example is in a relatively small key space. Since the speed-up of Grover's algorithm is quadratic, there will be a dramatic difference as the key space increases.

VIII. Conclusion

Due to its wide application in classical engineering problems, Grover's algorithm is an example application that can take advantage of quantum mechanics to perform nanoscale computing. This algorithm allows a quadratic improvement compared with classical brute force search. The algorithm is built on top of two key operations, namely: selective inversion and inversion-about-average. In this paper, we present how quantum Boolean circuits can be used to implement the selective inversion and inversion-about-average in the quantum search algorithm. We give detailed circuit designs for three applications. The first application is for solving problems that are expressed in Boolean function. The other two applications include searching in a phone book and attacking a symmetric cryptosystem. As previously discussed, though these two examples are usually referred in explanation of Grover's algorithm, they are quite different when implemented with quantum circuits. The main reason is that each record in a database is different and cannot be formulated effectively. As a contrast, the key-value pair in a cryptosystem has a very specific formulation, which can be applied to all the keys.

As a matter of fact, **not** any problem involving a brute-force search can be solved by the proposed quantum circuit design method. The problem can only be applied in the following situation: Given a one-way function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ and a m-bit integer M, the purpose is to find an n-bit integer x such that $f(x) = M$. The template circuits for solving such a one-way function is shown in Fig. 18. The first step in Fig. 18 prepares a superposition of $(1/\sqrt{2^n})(\sum_x |x\rangle|0\rangle)$. Then the second step applies a unitary gate implementing the function f to get $(1/\sqrt{2^n})(\sum_x |x\rangle|f(x)\rangle)$. The third step is the Grover's iteration which consists of the oracle and inversion-about-average as described before. Finally, the result can be obtained with a measurement.

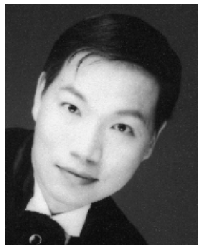
References

[1] C. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proc. Int. Conf. Comput. Syst. Signal Process.*, 1984, pp. 175–179.
[2] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Foundations Comput. Sci.*, Santa Fe, NM, USA, 1994, pp. 124–134.
[3] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Ann. ACM Symp. Theor. Comput.*, 1996, pp. 212–219.
[4] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte Der Phys.*, vol. 46, pp. 493–506, 1998.
[5] C. Zalka, "Grover's quantum searching algorithm is optimal," *Phys. Rev. A*, vol. 60, pp. 2746–2751, 1999.
[6] G. Chen, S. A. Fulling, and J. Chen, , R. K. Brylinski and G. Chen, Eds., "Generalization of grover's algorithm to multiobject search in quantum computing, part i, continuous time and discrete time," in *Mathematics of Quantum Computation*. Boca Raton, FL: CRC, 2002, pp. 135–160.
[7] B. Ofer, S. Daniel, and S. Yishai, "Analysis of grover's quantum search algorithm as a dynamical system," *Phys. Rev. A*, vol. 68, p. 022326, 2003.
[8] L. Accardi and R. Sabbadini, A Generalization of Grover's Algorithm. ArXiv Quantum Physics e-Prints, Cornell Univ., Ithaca, NY, Dec. 2000 [Online]. Available: http://www.arxiv.org/abs/quant-ph/0012143
[9] G. Alber, M. Musinger, and A. Delgado, "Dynamical stabilization of Grover's algorithm with embedded quantum codes," in *Proc. SPIE*, 2001, vol. 4429, pp. 37–51.
[10] A. Galindo and M. A. Martin-Delgado, "Family of Grover's quantum-searching algorithms," *Phys. Rev. A*, vol. 62, p. 62303, 2000.
[11] S. Aaronson, "Lower bounds for local search by quantum arguments," in *Proc. 36th. ACM Symp. Theor. Comput.*, 2004, pp. 465–474.
[12] A. Ambainis, "Quantum search algorithms," *SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.
[13] A. N. Soklakov and R. Schack, Efficient State Preparation for a Register of Quantum Bit. ArXiv Quantum Physics e-Prints, Cornell Univ., Ithaca, NY, Nov. 2005 [Online]. Available: http://www.arxiv.org/abs/quant-ph/0408045
[14] C. Durr and P. Hoyer, A Quantum Algorithm for Finding the Minimum. ArXiv Quantum Physics e-Prints, Cornell Univ., Ithaca, NY, Jan. 1999 [Online]. Available: http://xxx.lanl.gov/quant-ph/9607014
[15] H. Buhrman, C. Durr, M. Heiligman, P. H. Yer, F. Magniez, M. Santha, and R. de Wolf, "Quantum algorithms for element distinctness," in *Proc. Computational Complexity. 16th Annu. IEEE Conf.*, Chicago, IL, 2001, pp. 131–137.
[16] S. Okubo, T. Nishino, K. Ohta, and N. Kunihiro, "A quantum algorithm for finding the minimum on nmr quantum computers," in *Proc. ERATO Workshop Quantum Inf. Sci.*, 2004, pp. 152–153.
[17] G. Brassard, P. Hoyer, and A. Tapp, "Quantum cryptanalysis of hash and claw-free functions," in *Proc. LATIN'98*, 1998, pp. 163–169.
[18] G. Brassard, P. Hoyer, and A. Tapp, "Quantum counting," in *Proc. 25th Int. Colloquium Autom.*, 1998, pp. 820–831.
[19] Z. Diao, Z. M. Suhail, and G. Chen, "A quantum circuit design for grover's algorithm," *Phys. Rev. Lett.*, vol. 57, pp. 701–708, 2002.
[20] J. Roland and N. J. Cerf, "Quantum-circuit model of Hamiltonian search algorithms," *Phys. Rev. A*, vol. 68, p. 62303, 2003.
[21] L. Xiao and J. A. Jones, "Robust logic gates and realistic quantum computation," *Phys. Rev.*, vol. 73, p. 032334, 2006.
[22] I. M. Tsai, S. Y. Kuo, and D. Wei, "Quantum boolean circuit approach for searching an unordered database," in *Proc. 2nd IEEE Conf. Nanotechnology*, 2002, pp. 315–318.
[23] S. Lomonaco, Grover's Quantum Search Algorithm. ArXiv Quantum Physics e-Prints, Cornell Univ., Ithaca, NY, Oct. 2000 [Online]. Available: http://www.arxiv.org/abs/quant-ph/0010040
[24] I. M. Tsai and S. Y. Kuo, "Quantum Boolean circuit construction and layout under locality constraint," in *Proc. 1st IEEE Conf. Nanotechnology*, 2001, pp. 111–116.
[25] S. A. Cook and D. G. Mitchell, "Finding hard instances of the satisfiability problem: A survey," in *Satisfiability Problem: Theory and Applications*, Du, Gu, and Pardalos, Eds. New York: American Mathematical Society, 1997, vol. 35, pp. 1–17.
[26] IBMs Test-Tube Quantum Computer Makes History: First Demonstration of Shors Historic Factoring Algorithm, IBM Research News, Yorktown, NY, Dec. 2001 [Online]. Available: http://www.domino.research.ibm.com/comm/pr.nsf/pages/news.20011219_quantum.html
[27] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte Der Phys.*, vol. 48, no. 9-11, pp. 771–783, Sep. 2000.

[28] Y. Kanamori, S. M. Yoo, W. D. Pan, and F. Sheldon, "A short survey on quantum computers," *Int. J. Comput. Applicat.*, vol. 28, pp. 227–233, 2006.
[29] B. E. Kane, "A silicon-based nuclear spin quantum computer," *Nature*, vol. 393, pp. 133–137, 1998.
[30] T. D. Ladd, J. R. Goldman, F. Yamaguchi, Y. Yamamoto, E. Abe, and K. M. Itoh, "An all silicon quantum computer," *Phys. Rev. Lett*, vol. 89, p. 017901, 2002.
[31] R. G. Clark *et al.*, "Progress in silicon-based quantum computing," *Roy. Soc. London Trans. Series*, vol. 361, pp. 1451–1471, 2003.
[32] P. G. Kwiat, J. R. Mitchell, P. D. D. Schwindt, and A. G. White, "Grover's search algorithm: An optical approach," *J. Modern Opt.*, vol. 47, pp. 257–266, 2000.
[33] J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning, "Demonstration of an all-optical quantum controlled-not gate," *Nature*, vol. 426, pp. 264–267, 2003.
[34] N. Gershenfeld and I. Chuang, "Bulk spin-resonance quantum computation," *Sci*, vol. 275, no. 5298, pp. 350–356, Jan. 17, 1997.
[35] T. F. Havel, S. S. Somaroo, C.-H. Tseng, and D. G. Cory, "Principles and demonstrations of quantum information processing by NMR spectroscopy," *Appl. Alg. Eng., Commun., Comput.*, vol. 10, pp. 339–374, 2000.
[36] I. M. Tsai, S. Y. Kuo, S. L. Huang, Y. C. Lin, and T. T. Chen, "Experimental realization of an NMR quantum switch," in *Proc. ERATO Conf. Quantum Inf. Sci.*, 2004.
[37] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[38] D. DiVincenzo, "Two-bit gates are universal for quantum computation," *Phys. Rev.*, vol. 51, no. 2, pp. 1015–1022, Feb. 1995.
[39] A. Barenco, "A universal two-bit gate for quantum computation," *Proc. Roy. Soc. Lond.*, vol. 449, pp. 679–683, 1995.
[40] Data Encryption Standard, NIST, Boulder, CO, Oct. 1999 [Online]. Available: http://www.csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[41] DES Challenge III, RSA, Bedford , MA, Jan. 1999 [Online]. Available: http://www.rsa.com/rsalabs/node.asp?id=2108
[42] Distributed.Net, Distributed Computing Technologies, Jan. 1999 [Online]. Available: http://www.distributed.net/des
[43] *Electronic Frontier Foundation, Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. Sebastopol, CA: O'Reilly, 1998.

Yi-Lin Ju received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2000 and 2002, respectively. She is currently working toward the Ph.D. degree in the Department of Electrical Engineering at the same university.

Her current research interest is quantum information processing.



I-Ming Tsai received the B.S. degree in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., the M.S. degree in telecommunications from the University of Pittsburgh, Pittsburgh, PA, and the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., 1988, 1993, and 2003, respectively.

Since 1993, he has been with Chunghwa Telecommunication Laboratories, Taoyuan, Taiwan, R.O.C., where he works on circuit design, broadband switching, information security, and network architecture evolution. His research fields include experimental and theoretical aspects of quantum computing and quantum communication.

Sy-Yen Kuo (S'85–M'88–SM'98–F'01) is a Chair Professor and Dean of the College of Electrical and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan. He is also a Distinguished Professor at the Department of Electrical Engineering, National Taiwan University where he is currently taking a leave of absence and was the Chairman at the same department from 2001 to 2004. He received the BS (1979) in Electrical Engineering from National Taiwan University, the MS (1982) in Electrical & Computer Engineering from the University of California at Santa Barbara, and the PhD (1987) in Computer Science from the University of Illinois at Urbana-Champaign. He spent his sabbatical years as a Visiting Professor at the Computer Science and Engineering Department, the Chinese University of Hong Kong from 2004–2005 and as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000, respectively. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty fellow at Jet Propulsion Laboratory of California Institute of Technology. His current research interests include dependable systems and networks, software reliability engineering, mobile computing, and reliable sensor networks. Professor Kuo is an IEEE Fellow. He has published more than 270 papers in journals and conferences, and also holds several patents. He received the distinguished research award between 1997 and 2005 consecutively from the National Science Council in Taiwan and is now a Research Fellow there. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference(DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.