A SHORT SURVEY ON QUANTUM COMPUTERS

Y. Kanamori,[*] S.-M. Yoo,[*] D.W. Pan,[*] and F.T. Sheldon [**]

[*] Electrical and Computer Engineering Department
The University of Alabama in Huntsville
Huntsville, AL 35806, USA
e-mail: {yoshitok, yoos, dwpan}@eng.uah.edu

[**] Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831 USA
e-mail: SheldonFT@ornl.gov

Abstract

Quantum computing is an emerging technology. The clock frequency of current computer processor systems may reach about 40 GHz within the next 10 years. By then, one atom may represent one bit. Electrons under such conditions are no longer described by classical physics and a new model of the computer may be necessary by that time. The quantum computer is one proposal that may have merit in dealing with the problems. Currently, there exist some algorithms utilizing the advantage of quantum computers. For example, Shor's algorithm performs factoring a large integer in polynomial time while classical factoring algorithms can do it in exponential time. In this paper we briefly survey the current status of quantum computers, quantum computer systems, and quantum simulators.

Key Words
Classical computers, quantum computers, quantum computer systems, quantum simulators, Shor's algorithm.

1. Introduction

How much can the performance of a computer be improved? According to Moore's law, if the performance keeps improving by means of technological innovations, which have been done over the last few decades, the number of transistors per chip may be doubled every 18 months. Furthermore, processor clock frequency could reach as much as 40 GHz within 10 years [1]. By then, one atom may represent one bit [1]. One of the possible problems may be that, since electrons are not described by classical physics but by quantum mechanics, quantum mechanical phenomenon may cause "tunneling" to occur on a chip. In such cases, electrons could leak from circuits. Taking into account the quantum mechanical characteristics of the one-atom-per-bit level, quantum computers have been proposed as one way to effectively deal with this predicament. In this way, quantum computers can be used to solve certain computationally intense problems where classical computers require large amounts of processing time. Notwithstanding, further improvements will be necessary to ensure quantum computers proper performance in future, but such improvements seem obtainable.

Currently, there exist some algorithms utilizing the advantage of quantum computers. For instance, the polynomial-time algorithm for factoring a large integer with $O(n^3)$ time was proposed by Peter Shor [2]. This algorithm performs factoring exponentially faster than classical computers. This algorithm could factor a 512-bit product in about 3.5 hours with 1 GHz clock rate [3] whereas the number field sieve could factor the 512-bit product in 8,400 MIPS years [4]. (1 MIPS year is the number of instruction that a processor can execute in a year, at the rate of millions of instructions per second.) Another famous quantum algorithm is a database search algorithm proposed by Lov Grover that will find a single item from an unsorted list of N elements with O($\sqrt{N}$) time [5].

In this paper we briefly survey quantum computers. First, the main characteristics of quantum computers, superposition states and interference are introduced. Then, current approaches on quantum computers are reviewed. Next, research on quantum computer simulators is introduced. Lastly, a concluding remark follows.

2. Quantum Computer Systems

2-1 Superposition State

In classical computers, electrical signals such as voltages represent the 0 and 1 states as one-bit information. Two bits indicate four states 00, 01, 10, and 11, and n bits can represent 2^n states. In the quantum computer, a quantum bit called “qubit”, which is a two-state system, represents the one-bit information. For instance, instead of an electrical signal in classical computers, an electron can be used as a qubit. The spin-up and spin-down of an electron represent two states: 0 and 1, respectively. A photon can also be used as a qubit and the horizontal and vertical polarization of a photon can be used to represent both states. Using qubits, quantum computers can perform arithmetic and logical operations as does a classical computer. The important difference however, is that one qubit can also represent the superposition of 0 and 1 states. When we represent 0 and 1 states as state vectors $|0\rangle$ and $|1\rangle$ respectively, such a superposition state is expressed as a linear combination of $|0\rangle$ and $|1\rangle$, $|\psi\rangle = a|0\rangle + b|1\rangle$. “$|\ \rangle$” is called “ket-vector” in Dirac notation and the coefficients a and b are called *probability amplitudes*. $|a|^2$ indicates a probability that we get $|\psi\rangle = |0\rangle$ as a result of the measurement on the qubit

$|\psi\rangle = a|0\rangle + b|1\rangle$. They also satisfy $|a|^2 + |b|^2 = 1$. For example, when the probability amplitudes: *a* and *b* are equal to $\frac{1}{\sqrt{2}}$, we can express a superposition state of two states as following: $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, where vectors $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$. In short, when we measure a state of $|\psi\rangle$, the state will be observed as $|0\rangle$ with probability $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ and as $|1\rangle$ with probability $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$.

This bizarre characteristic in quantum computers makes parallel computation possible in the real sense of the term. Since each qubit represents two states at the same time, two qubits can represent four states simultaneously. For instance, when we use two qubits that are the superposition of 0 and 1 states as an input for an operation, we can get the result of four operations for four inputs with just one computational step as compared to four operations needed by the classical computer. Likewise, when using *n* qubits, we can make a superposition of 2^n states as an input and process the input in just one step to solve a problem that a classical computer requires 2^n steps. In this light, a quantum computer can process *n* inputs with only one computational step after taking the superposition state of *n* inputs.

However, a crucial problem is left before we can use this extremely valuable characteristic of quantum computers. From the input of one superposition state representing four states and processing in one step we get the superposition of four results. When we measure the output qubits, the quantum mechanical superposition collapses and each qubit will be observed as either 0 or 1 since a qubit is a two-state system. Consequently, we only get one of the four possible results: 00, 01, 10 or 11 (for $n = 2$) with the same probability. Accordingly, the superposition of qubits is governed by probability, and the measurement is necessary to determine which one of the possible states is represented. This difficulty arises from using the quantum mechanical superposition. If, however, we can increase the probability of getting the expected result by devising an algorithm, we may take advantage of the quantum mechanical superposition feature. In this way, as discussed above, we can harness the power of quantum computers to solve a problem that takes an excessive amount of computational time and energy for certain problem classes on classical computers.

2-2. Interference

In this subsection, we show a simple example that illustrates the difference between classical and quantum computation, and the importance of interference-of-states in quantum computation.

Clearly, any classical computer can be simulated by a Turing machine, a mathematical model of a general computer. Before we discuss the quantum Turing machine (QTM), we introduce a computation tree using a classical probabilistic Turing machine (PTM) [6]. Fig. 1 shows an example of a state transition diagram for the PTM, and Fig. 2 derives the PTM as a computation tree. In the tree, each vertex shows a machine state and each edge shows the probability of transition occurrence.

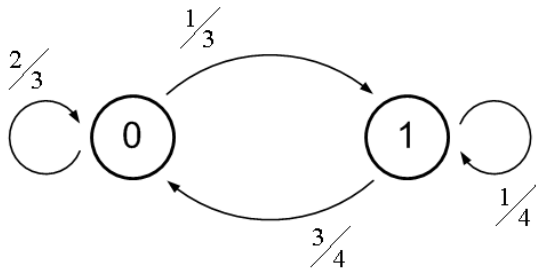


Figure 1. A state transition diagram of PTM.

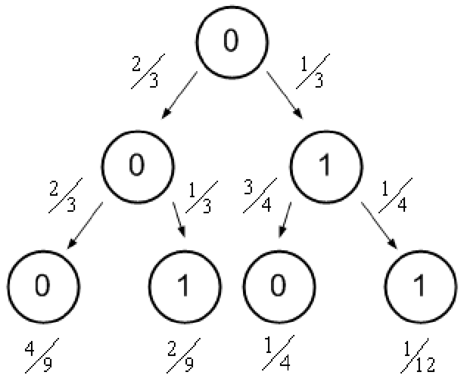


Figure 2. A computation tree of PTM.

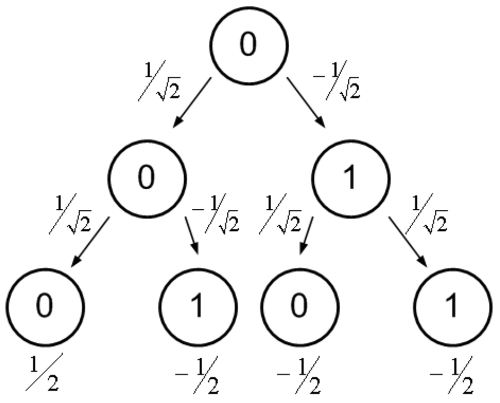
Also, each level of the tree represents a computation step and the tree's root represents the starting state. We can compute a probability of transition $0 \to 1$ after two computational steps, by summing the probabilities of the two possible paths from the root to state 1 as follows:

$$P(0 \to 1) = (\frac{2}{3} \times \frac{1}{3}) + (\frac{1}{3} \times \frac{1}{4}) = \frac{2}{9} + \frac{1}{12} = \frac{11}{36}$$

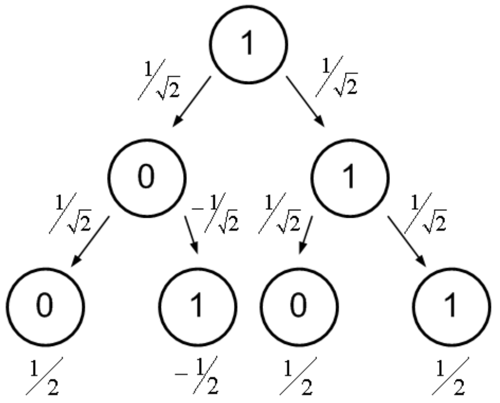
Similarly,

$$P(0 \to 0) = (\frac{2}{3} \times \frac{2}{3}) + (\frac{1}{3} \times \frac{3}{4}) = \frac{4}{9} + \frac{3}{12} = \frac{25}{36}.$$

We can interpret this result in the following way. In two steps, starting from state 0 the PTM will occupy state 1 with probability $\frac{11}{36}$ and state 0 with probability $\frac{25}{36}$.



(a) QTM starts from state 0



(b) QTM starts from state 1

Figure 3. A computation tree of a QTM [7]

Similar to PTM, we describe a computation of QTM using the computation tree shown in Fig. 3. Each edge of the tree in QTM represents a probability amplitude while in the PTM each edge represents a transition probability. Only one state in the same level of the PTM tree occurs at a time, but all states in the same level of the QTM tree occur simultaneously! For this example, the probability of $0 \rightarrow 1$ from the root after one computational step is

$$(-\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$$

and the probability of $0 \to 0$ from the root after one computational step is

$$(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}.$$

Let us compute the probability of transition $0 \to 1$ after two steps. First, we need to find the probability amplitudes of the two possible paths: $\Psi(0 \to 0 \to 1)$, and $\Psi(0 \to 1 \to 1)$.

$$\Psi(0 \to 0 \to 1) = \frac{1}{\sqrt{2}} \times (-\frac{1}{\sqrt{2}}) = -\frac{1}{2},$$

$$\Psi(0 \to 1 \to 1) = (-\frac{1}{\sqrt{2}}) \times (\frac{1}{\sqrt{2}}) = -\frac{1}{2}$$

We add both amplitudes.

$$\Psi(0 \to 1 \text{ after two steps}) = \Psi(0 \to 0 \to 1) + \Psi(0 \to 1 \to 1) = (-\frac{1}{2}) + (-\frac{1}{2}) = -1.$$

Thus, the probability of transition $0 \to 1$ after two steps is

$$P(0 \to 1 \text{ after two steps}) = |\Psi(0 \to 1 \text{ after two steps})|^2 = |(-1)|^2 = 1.$$

Similarly, we compute the probability of transition $0 \to 0$ after two steps.

$$\Psi(0 \to 0 \to 0) = \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} = \frac{1}{2},\ \Psi(0 \to 1 \to 0) = (-\frac{1}{\sqrt{2}}) \times (\frac{1}{\sqrt{2}}) = -\frac{1}{2}$$

$$\Psi(0 \to 0 \text{ after two steps}) = \Psi(0 \to 0 \to 0) + \Psi(0 \to 1 \to 0) = \frac{1}{2} + (-\frac{1}{2}) = 0$$

$$P(0 \to 0 \text{ after two steps}) = |\Psi(0 \to 0 \text{ after two steps})|^2 = |0|^2 = 0$$

This is a very remarkable result. After one computational step, the probability $0 \to 1$ and $0 \to 0$ were both $\frac{1}{2}$. But, after two computational steps from the same root the probability $0 \to 1$ is 1 and probability $0 \to 0$ is 0. This result occurs because the probability amplitudes can have negative values. We interpret this result as the states of the QTM interfered with each other. In short, the case “$0 \rightarrow 1$ after two steps” had a constructive interference [$(-\frac{1}{2}) + (-\frac{1}{2}) = -1$] and the case “$0 \rightarrow 0$ after two steps” had a destructive interference [$\frac{1}{2} + (-\frac{1}{2}) = 0$].

In the previous subsection, we mentioned that the result of a computation involving the superposition of *n* input states is a superposition of *n*-output states. For example, if we need to perform factorizing of an *n*-digit binary number into two prime factors, we must test 2^{n-1} numbers with Eratosthenes' sieve as the worst-case scenario. Therefore, we must make a superposition of 2^{n-1} integers as input giving the result from factoring as the superposition of 2^{n-1} outputs.

If we can design an operation such that, a constructive interference occurs at desired outputs (e.g., prime factors) of the superposition of 2^{n-1} outputs and, a destructive interference occurs at unnecessary outputs, we can find prime factors with only one computational step as compared to the classical computer which takes 2^{n-1} steps. This is an immense improvement in computation time.

Shor’s algorithm performs factoring of large integers, though it is not just a single-step operation as described. The algorithm consists of both quantum and classical processing. The quantum processing part utilizes quantum interference and the superposition state to find the period r of the function $f_{x,n}(a) = x^a \bmod n$ where n is an integer to be factored and x is an integer chosen at random that is coprime to n (i.e., $\gcd(x, n) = 1$). The classical part makes use of a result from classical number theory to find a factor of n by using x and r from the quantum part.

3. Current Approaches on Quantum Computers

In this section we consider how such a quantum computer can be built. There are five experimental requirements for building a quantum computer [8, 9]. The first requirement necessitates the ability to represent quantum information robustly. Since a qubit is a simple two-level system, a physical qubit system will have a finite set of accessible states. Some examples are the spin states of a spin ½ particle, the ground states and first excited states of an atom, and the vertical and horizontal polarization of a single photon. Second, a quantum computer requires the ability to set a fiducial initial state. This is a significant problem for most physical quantum systems because of the imperfect isolation from their environment and the difficulty of producing desired input states with high fidelity. Third, a quantum computer requires long decoherence times much longer than the gate operation time. Decoherence is the coupling between the qubit and its environment, which results in a loss of the quantum phase coherence. After decoherence, the quantum mechanical property associated with coherence (e.g., superposition, entanglement) can be no longer observed. The fourth requirement is the capability of measuring output results from specific qubits. The outcome from a quantum algorithm is, in general, a quantum superposition. Therefore, it is necessary to read out a result from the quantum state using the classical system with high fidelity. The fifth requirement concerns the ability to construct a universal set of quantum gates. Similar to a classical computer, a quantum computer has universal gates, which implement any legitimate quantum computation. DiVincenzo proved that just two-qubit gates at a time are adequate to build a general quantum circuit [10]. Using two-qubit controlled-NOT gate and single-qubit gates, we can compose any multiple qubit logic gates. Moreover, once we can construct a two-qubit controlled-NOT gate, we can also build a quantum computer with combinations of these gates.

Several implementations for a quantum computer have been proposed. One of the well-researched implementations is a nuclear magnetic resonance- (NMR) based quantum computer. This computer uses a vial of a liquid filled with sample molecules as qubits. In this way, this experimental quantum computer solves a problem by controlling nuclear spins using NMR techniques and retrieves the results observing the ensembled average of some property of the nuclear spins in the vial. A seven-qubit NMR based quantum computer has been built, and the computer can perform Shor's algorithm finding factors of the number 15 [11]. This is currently the most advanced quantum computer.

An ion-trap-based quantum computer uses a string of ions confined in a linear trap [12]. Each ion represents a qubit and is manipulated by laser beams. Photons from ions are observed as a result of an operation by photo detectors. A two-qubit controlled-NOT gate has already been demonstrated [13], and a quantum computer with a large number of trapped ions has been proposed [14].

A cavity quantum electrodynamics (QED) based quantum computer has been proposed [15]. This scheme uses photons as qubits and implements a controlled-NOT gate using the interaction of a linearly polarized photon as a target bit and a circularly polarized photon as a control bit through cesium atoms inside an electromagnetic cavity [1]. They measure a phase shift of the photon from the cavity as an output qubit.

In [16,17], a linear optics quantum computer is proposed using photons. An optical mode (e.g., horizontal or vertical polarization) of a photon represents a state of qubits. Quantum gates can be realized only with linear optical elements. Placing beam splitters and phase shifters between the paths of photons can control the states of qubits for computations. As a two-qubit gate operation, a non-deterministic controlled-NOT gate has been proposed. This gate operation requires additional ancillary photons, which are not part of the computation, and single-photon detections.

A quantum-dot-based quantum computer uses spins [18] or energy levels [19] of electrons confined in quantum dots (QDs) as qubits that are fabricated in semiconductor materials. Since we can control states of qubits electrically, as we do in classical circuits, this scheme has an advantage because current semiconductor technology may be applied to the fabrication of a quantum computer.

A superconducting quantum computer uses the Josephson-junctions in superconducting circuits as qubits [20]. Charge or energy levels in a junction represent information of qubits. A controlled-NOT gate operation on the charged qubits was demonstrated, but the phase evolution during the gate operation has not yet been examined [21]. An implementation of the real quantum controlled-NOT gates is the next challenge toward the realization of universal logic gates.

Although each proposed quantum computer has difficulties in its realization, a common critical problem is that real quantum memory registers incur errors caused by environmental coupling (e.g., cosmic radiation, spontaneous emission, and decoherence.) As it is extremely difficult to isolate quantum registers perfectly from their environment, a real quantum computer must be designed considering the effect of errors on the state of the quantum registers.

To protect quantum states against the effects of noise, several quantum error-correcting (QEC) schemes have been proposed [22-25]. QEC codes could be developed based upon principles similar to a classical error correcting code. However, we need to circumvent the following three difficulties to design a QEC code [8]. First, we cannot produce a repetition code (e.g., logical 0 and 1 is encoded as “000” and “111”, respectively) by duplicating the quantum state several times because the no-cloning theorem [26] states that replication of an arbitrary quantum state is not possible. Second, unlike a classical bit, inspecting the state to assess its correctness can destroy a qubit. Third, since the state of qubit depends on certain continuous parameters (e.g., a rotation angle θ), quantum errors are continuous. Consequently, infinite precision is required to determine which error occurred to correct them.

By implementing the QEC codes on a quantum circuit, we can reduce the effect of noise on quantum registers and transmissions. However, it is not sufficient for quantum computation because in practice gate operations (e.g., encoding, decoding, and error correction) on the quantum circuit are themselves prone to errors. Moreover, these errors are propagated and accumulated continuously until the computation is completed.

To prevent the propagation and accumulation of errors on the quantum states, each procedure block in the quantum circuit (e.g., encoder, decoder, and error-correcting circuit) should be designed carefully so that any failure during the procedure can only propagate to a smaller number of qubits than can be corrected by the QEC codes. Such procedures are called as fault-tolerant procedures [8]. The detailed techniques are presented in the theory of the fault-tolerant quantum computation [27-33]. According to the threshold theorem [8], an arbitrarily large quantum computation can be efficiently performed if the error probability per gate (EPG) is less than a certain constant threshold. Recent research [34] indicates that the estimates of the EPGs are as high as three percent if sufficient resources are available.

4. Quantum Computer Simulators

As introduced above, the number of groups attempting to realize physical qubits has increased of late, however it will take many more years before quantum gates are available for the computer scientists/engineer to use. In the mean time, we need a quantum computer simulator to find new algorithms. Quantum computer systems can be mathematically represented by using vectors and matrices. When we define $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$, a NOT operation for one qubit can be expressed with 2×2 unitary matrices as

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle .$$

We can represent an operation that an initial condition $|1\rangle$ is converted to a superposition state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ by using a matrix: $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$

$$H \cdot |1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

In particular, this operation is known as *Hadamard transformation* [8].

Multiple qubits are represented as a tensor product of two vectors $|0\rangle$ and $|1\rangle$. For example, two qubit resisters are represented as following.

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}^T, \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix}^T$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}^T, \qquad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}^T$$

The controlled-NOT operation is

$$|00\rangle \to |00\rangle, |01\rangle \to |01\rangle, |10\rangle \to |11\rangle, |11\rangle \to |10\rangle$$

The first bit is called the *controlled bit* and the second bit is the *target bit*. A unitary matrix of controlled-NOT operations for two qubits is represented

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

For an *n*-qubits resister, a $2^n \times 2^n$ matrix is needed. We can also define the effect of errors on a qubit (i.e., a bit flip, a sign shift, both bit flip and sign shift) as the sum of the Pauli matrices:

$$\sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \quad \sigma_y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}, \quad \sigma_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}.$$

Thus, by using the vectors and unitary matrices, a theoretical quantum computer can be simulated mathematically.

Many quantum computer simulators have been proposed and implemented [35, 36]. Some researchers have simulated a quantum computer with commercial mathematics software packages. For example, Williams provided a simulator as a Mathematica notebook [1]. This simulator shows some basic operations on quantum computers and

Shor’s algorithm. Next, a commercial software “quantum computer simulator” was released [37]. This software allows users to simulate many sample algorithms (e.g., Shor’s algorithm, Grover’s algorithm) and user-designed circuits with a cleaver graphical user interface.

The theoretical quantum computer simulators, in general, perform highly idealized unitary operations. In practice, the unitary operations on a physical system are more complicated. Therefore, another type of quantum computer simulator has been developed as an emulator of quantum computer hardware [38]. This type of emulator simulates more realistic models strictly following the law of quantum mechanics. Michielsen simulates a NMR-like quantum computer [39]. The hardware in the simulator is modeled in terms of quantum spins that evolve in time according to the time-dependent Schrödinger equation [40]. The detailed explanation is given in [36].

A general and significant problem of quantum computer simulators is their inability to simulate quantum computers with a large number of qubits (e.g., 500, 1000 and more bits required for RSA encryption algorithm). To represent a large number of qubits, an exponentially large memory is required (described earlier). Therefore, when we simulate a quantum computer with a large number of qubits, we need to use a parallel computer [41]. For example, in [42] a quantum computer with up to 30 qubits was simulated using an 8-processor parallel computer.

5. Conclusions and Further Research

In this paper we reviewed the principles, algorithms and hardware considerations for quantum computing. Several research groups are investigating qubits and quantum logic circuitry using different resources (i.e., atom, ion, electron, photon among others). The realization of a practical quantum computer is expected before we encounter the limit of Moore's law with respect to improvements that may be possible using the classical computer model. A current realizable quantum computer is based on 7-bit NMR, which can factor 15. Further research is needed, for example, via simulation, on quantum computers using classical computers. Such a simulator must be able to handle quantum computers that operate on a practically large number of qubits. To this end, we need to employ large scale parallel processing methods to acquire more meaningful results within a practical time frame. By applying the methods/concepts of classical computers such as hardware abstraction to quantum computers, the research progress may be accelerated. For example, some groups proposed quantum programming languages that allow us to think of quantum computer operations in an abstract manner as we do with a classical computer [43-45].

The history of realization for quantum computers has just begun. Undoubtedly, we need more intensive research in a physical realization of components of quantum computers [46]. Computer scientists/engineers will need to consider the various architectural solutions for quantum computers as well as the various new (practical) quantum algorithms to advance the state of the art for quantum computers.

Acknowledgement

The authors would like to thank the anonymous reviewers who helped a lot in improving the paper. This manuscript has been authored by UT-Battelle, a contractor of the U.S. Government (USG) under Department of Energy (DOE) Contract DE-AC05-00OR22725. The USG retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

References

[1] C.P. Williams & S.H. Clearwater, *Exploration in quantum computing* (New York, Springer-Verlag, 1997).

[2] P.W. Shor, Algorithm for quantum computation: discrete logarithm and factoring, *Proc. 35th IEEE Annual Symp. on Foundations of Computer Science*, Santa Fe, NM, Nov. 1994, 24-134.

[3] M. Oskin, F.T. Chong, & I. Chuang, A Practical architecture for reliable quantum computers, *IEEE Computer,* Jan. 2002, 79-87.

[4] B. Preneel, ed., Factorization of a 512-bit RSA modeules, vol. 1807, *Lectrure Notes in Computer Science* (Springer-Verlag, Berlin, 2000).

[5] L.K. Grover, A fast quantum mechanical algorithm for database search, *Proc. STOC*, Philadelphia, 1996, 212-219.

[6] D.R. Simon, On the power of quantum computation, *Proc. 35th Annual Symp. on Foundations of Computer Science,* 1994, 116 – 123.

[7] T. Nishino, *Introduction to Quantum Computer* (in Japanese, Tokyo Denki University Press, Tokyo, 1997).

[8] M.A. Nielsen & I.L. Chuang, *Quantum computation and quantum information* (Cambridge University Press, 2000).

[9] D.P. DiVincenzo, The physical implementation of quantum computation, *Fortschr. Phys.* 48, (9-11), 2000, 771-783.

[10] D.P. DiVincenzo, Two-bit gates are universal for quantum computation, *Phys. Rev. A*, 51, 1995, 1015–1022.

[11] IBM research news, IBM's test-tube quantum computer makes history, first demonstration of Shor's historic factoring algorithm, http://www.research.ibm.com/resources/news/20011219_quantum.shtml.

[12] J.I. Cirac & P. Zoller, Quantum computations with cold trapped Ions, *Phys. Rev. Lett.* 74, 1995, 4091.

[13] C. Monroe, D.M. Meekhof, B.E. King, W.M. Itano, & D.J. Wineland, Demonstration of a fundamental quantum logic gate, *Phys. Rev. Lett.* 75, 1995, 4714.

[14] D. Kielpinski, C. Monroe, & D.J. Wineland, Architecture for a large-scale ion-trap quantum computer, *Nature,* 417, 2002, 709 – 711.

[15] Q.A. Turchette, C.J. Hood, W. Lange, H. Mabuchi, & H.J. Kimble, Measurement of conditional phase shifts for quantum logic, *Phys. Rev. Lett.* 75, 1995, 4710–4713.

[16] E. Knill, R. Laflamme, & G. J. Milburn, A scheme for efficient quantum computation with linear optics, *Nature*, 409, 2001, 46–52.

[17] T.C. Ralph, A.G. White, & G.J. Milburn, Simple scheme for efficient linear optics quantum gates, *Phys. Review A*, 65, 2002, 012314.

[18] D. Loss & D.P. DiVincenzo, Quantum computation with quantum dots, *Phys. Review A*, 57, 1998, 120-126.

[19] M.S. Sherwin, A. Imamoglu, & T. Montroy, Quantum computation with quantum dots and terahertz cavity quantum electrodynamics, *Phys. Review A*, 60, 1999, 3508-3514.

[20] Y. Makhlin, G. Schön, & A. Shnirman, Quantum-state engineering with Josephson-junction devices, *Reviews of Modern Physics* 73, 2001, 357–400.
[21] T. Yamamoto, Y.A. Pashkin, O. Astafiev, Y. Nakamura, & J.S. Tsai, Demonstration of conditional gate operation using superconducting charge qubits, *Nature,* 425, 2003, 941–944.
[22] P.W. Shor, Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A* 52, 1995, R2493–R2496.
[23] A. R. Calderbank & P.W. Shor, Good quantum error-correcting codes exist, *Phys. Rev. A* 54, 1996, 1098–1105.
[24] A. Ekert, & C. Macchiavello, Quantum error correction for communication, *Phys. Rev. Lett.* 77, 1996, 2585–2588.
[25] A.M. Steane, Error correcting codes in quantum theory, *Phys. Rev. Lett.* 77, 1996, 793–797.
[26] W.K. Wootters, & W.H. Zurek, A single quantum cannot cloned, *Nature*, 299, 1982, 802-803.
[27] P.W. Shor, Fault-tolerant quantum computation, *Proc. 37th IEEE Annual Symp. on Foundations of Computer Science*, Burlington, VT, 1996, 56 – 65.
[28] J. Preskill, Reliable quantum computers, *Proc.Roy.Soc.Lond.*, A454, 1998, 385-410.
[29] A.M. Steane, Efficient fault-tolerant quantum computing, *Nature*, 399, 124-126, 1999.
[30] D. Gottesman, Fault-tolerant quantum computation with local gates, *J.Mod.Opt.*, 47, 2000, 333-345.
[31] E. Knill & R. Laflamme, Concatenated quantum codes, quant-ph/9608012, 1996.
[32] P.O. Boykin, C.P. Roychowdhury, T. Mor, & F. Vatan, Fault tolerant computation on ensemble quantum computers, *International Conf. on Dependable Systems and Networks*, 2004, 157 – 166.
[33] E. Knill, R. Laflamme, & W. Zurek, Accuracy threshold for quantum computation, quant-ph/9610011, 1996.
[34] E. Knill, Quantum computing with realistically noisy devices, *Nature* 434, 2005, 39-44.
[35] J. Wallace, Quantum computer simulators, *Proc. of the 4th International Conf. on Computing Anticipatory Systems (CASYS 2000)*, Liege, Belgium, 7-12 August 2000.
[36] H. De Raedt & K. Michielsen, Computational methods for simulating quantum computers, quant-ph/0406210, 2004.
[37] World's first universal quantum computation simulator, Quantum computer simulator, SENKO corporation, http://www.senko-corp.co.jp/qcs/index.html.
[38] H. De Raedt, A.H. Hams, K. Michielsen, & K.De Raedt, Quantum Computer Emulator, *Computer Physics Communications*, 132 (1-2), 2000, 1-20.
[39] K. Michielsen, H. De Raedt, & K. De Raedt, A simulator for quantum computer hardware, *Nanotechnology* 13, 2002, 23-28.
[40] D.J. Griffiths, Introduction to quantum mechanics, (New Jersey, Prentice Hall, 1995, 1-2).
[41] K.M. Obenland & A.M. Despain, A parallel quantum computer simulator, *High Performance Computing*, 1998, quant-ph/9804039.

[42] J. Niwa, K. Matsumoto, & H. Imai, General-purpose parallel simulator for quantum computing, *Phys. Review A*, 66, 2002, 062317.
[43] J.W. Sanders & P. Zuliani, Quantum programming, *Mathematics of Program Construction, Springer LNCS*, 2000, 80-99.
[44] B. Ömer, Classical concepts in quantum programming, *Quantum Structures*, 2002.
[45] P. Selinger, Towards a quantum programming language, *Mathematical Structures in Computer Science*, *14*, 2003, 527.
[46] Quantum Computation Roadmap, http://qist.lanl.gov/qcomp_map.shtml.

Yoshito Kanamori received his M.Sc. degree in Computer Engineering from University of Alabama in Huntsville (UAH) in December 2002. He is a Ph.D candidate in Electrical and Computer Engineering Department, UAH. His research interests include quantum information processing, computer network security and optics.

Seong-Moo Yoo received the B.S. degree in economics from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degree in computer science from the University of Texas at Arlington in 1989 and 1995, respectively. Since September 2001, he is an associate professor in Electrical and Computer Engineering Department of the University of Alabama in Huntsville, Huntsville, Alabama, U.S.A. From September 1996 to August 2001, he was an assistant professor in Computer Science Department of Columbus State University in Columbus, Georgia, U.S.A. Dr. Yoo was the conference chair of ACM Southeast Conference 2004, April, 2004, Huntsville, Alabama, U.S.A. He was the co-program chair of ISCA 16th International Conference on Parallel and Distributed Computing Systems (PDCS-2003), August 2003, Reno, Nevada, U.S.A. Dr. Yoo's research interests include cryptography, computer network security, and mobile ad hoc networks. Dr. Yoo is a senior member of IEEE and a member of ACM.

W. David Pan is an Assistant Professor in the Department of Electrical and Computer Engineering, the University of Alabama in Huntsville. He received his Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles, in 2002, and his M.S. degree in Computer Engineering from the University of Louisiana at Lafayette in 1998. He has held summer research positions at the Advanced System Lab, SGS-Thomson (ST) Microelectronics, San Diego, California, and at AT&T Labs – Research, Middletown, New Jersey. His research interests include image and video compression, distributed source/channel coding for sensor networks, multimedia information assurance, as well as quantum computing.

Frederick T. Sheldon has over 22 years of experience in the field of computer science. He currently is a senior research staff member at ORNL. Formerly, he was assistant professor at WSU, CU and research staff at DaimlerChrysler, Lockheed Martin, Raytheon and NASA Langley/Ames. He received his Ph.D./MS at Univ. of Texas at Arlington in '96/'89 (also has two degrees from Univ. of Minnesota in Computer Science and Microbiology). He founded the Software Engineering for Secure and Dependable

Systems Lab in 1999. He is a Senior Member of the IEEE and member of ACM, IASTED, AIAA, including the Tau Beta Pi and Upsilon Pi Epsilon honor societies and has received the Sigma Xi award for an outstanding dissertation. He has published over seventy papers in various journals and international conferences (http://www.csm.ornl.gov/~sheldon/pubs.html).