

# Quantum Architecture: A Survey for Non-Physicists

Sean H. Whalen\*

Fall 2003

## Abstract

Quantum architecture is the organization of quantum and classical computational components for the execution of quantum algorithms. Enabled by the phenomena of superposition and entanglement, quantum computers could potentially execute classical exponential time algorithms in polynomial time. Such devices would bring revolutionary change – for instance, Shor’s quantum factoring algorithm could one day compromise public key cryptography. Extant quantum computing devices lag significantly behind current theory, but developments in quantum architecture are bringing closer the goal of a general purpose quantum computer. This paper is a survey of developments in quantum architecture, and provides appropriate background in physics.

## 1 Introduction

Quantum computing is an emerging field which combines computer science, computer architecture, and physics in hopes of conquering problems too large for classical (non-quantum) computers. Superposition and entanglement, in theory, enable quantum computation of classical exponential time algorithms in polynomial time.

It is easy to see the interest garnered by quantum computation. Many classical algorithms have run time exponential to input size, and are thus not computable by extant computers for large inputs. Two well known quantum algorithms are Shor’s for factoring  $n$ -bit integers in  $n^3$  time [8], and Grover’s for locating an item in a list of size  $n$  with  $\sqrt{n}$  queries [15].

Before a general purpose quantum computer is available, a quantum computing architecture must be designed. Quantum architecture is, in the words of Mark Oskin, “the organization and optimization of quantum and classical structures (i.e. the micro-architecture) and the interface (i.e. the ISA) for the efficient execution of quantum-enabled software”. Simply put, quantum architecture is the study of translating quantum theory into physical quantum computing components. These components can be assembled in a circuit model (resembling an ASIC) [4], or they can form a quantum Turing machine [14]. Much current research is working towards the latter, hoping to architect a general purpose, programmable quantum computing platform.

---

\*Department of Computer Science, University of California, Davis, USA, shwhalen@ucdavis.edu

In this paper, we present a survey of recent papers in quantum architecture. First we present a survey of the physics concepts needed to understand the rest of the paper.

## 2 Physics Background

Quantum mechanics is a branch of physics which models behavior of energy and matter at the atomic and subatomic level. Its name comes from *quanta*, the discrete packets of energy proposed by Max Plank in 1900.

Quantum mechanics asserts the following theories: (1) energy and matter exist in discrete particles, (2) energy and matter may behave as either particles or waves, (3) movement of these particles is unpredictable, and (4) simultaneous measurement of two complementary values of a particle cannot be achieved. Examples of such properties are energy, position, momentum, and angular momentum<sup>1</sup>.

This last theory is called *Heisenberg's uncertainty principle*. The principle states that before measurement of a quantum particle's state, it exists in all states simultaneously. This property is called *superposition*.

Another important property is called *entanglement* or *coherence*. Pairs of particles which interact become entangled, a process called *correlation*. These particles exist in complementary states, so if one entangled particle has an "up" spin, its partner has a "down" spin. As we shall see later, entanglement gives rise to a phenomenon called quantum teleportation. Together, entanglement and superposition enable quantum computation.

Superposition tells us that we cannot know a particle's state until it is measured. Before measurement, we can express a particle's state as a probability amplitude (defined in the next section). By measuring the state of one entangled particle, we are communicating this state to the partner who must assume the opposite state. This communication takes place instantly, over any distance, as long as the pair remains coherent [2].

The process of losing coherence is called *decoherence*, and can occur as a result of even a single stray photon. Decoherence is thought to explain why much of our world behaves according to classical, and not quantum, physics [1]. Remaining coherent requires complete isolation of a pair from the environment, and presents one of the greatest challenges to quantum architects. This is why quantum computation must be reversible at every stage: thermodynamics shows that a non-reversible computation must give off heat as a byproduct, which causes interaction with the environment, which leads to decoherence<sup>2</sup>. Extensive error correcting codes are used to get around the decoherence problem in practice, but with extreme time and space overhead [5].

---

<sup>1</sup>[http://en2.wikipedia.org/wiki/Quantum\\_mechanics](http://en2.wikipedia.org/wiki/Quantum_mechanics)

<sup>2</sup>[http://homepage.mac.com/sigfpe/Physics/qc\\_intro.html](http://homepage.mac.com/sigfpe/Physics/qc_intro.html)

### 3 Models of Computation

Classical computers can be modeled by a Turing machine. A *Turing machine* is an abstract model of computation comprised of an infinitely long sequential tape, a read/write head, a set of symbols, and a set of rules for reading/writing and moving the head based on the symbol under it. These rules are called *transition functions*, and the symbol under the head is called the *state*. The machine moves from one state to another by executing a matching transition function. If no transition function matches, the machine halts<sup>3</sup>. In the following example, the machine will transition from state 0 to 1, from 1 to 2, and from 2 back to 0.

$$\begin{aligned} |0\rangle &\rightarrow |1\rangle \\ |1\rangle &\rightarrow |2\rangle \\ |2\rangle &\rightarrow |0\rangle \end{aligned}$$

The *Church-Turing Thesis* says the following: A Turing machine can compute any function computable by a reasonable physical device. Programming languages can be translated into Turing machines and vice versa. Thus, the computation of any reasonable function can be expressed by a programming language.

The modern Church thesis says that a probabilistic Turing machine can simulate any physical device in polynomial cost [9]. A *probabilistic Turing machine* can select among a set of transition functions with varying probability, instead of just one. In the following example, real numbers preceding a state indicate the probability of transitioning to that state.

$$\begin{aligned} |0\rangle &\rightarrow 0.5|1\rangle + 0.5|2\rangle \\ |1\rangle &\rightarrow 0.9|0\rangle + 0.1|2\rangle \\ |2\rangle &\rightarrow 0.2|0\rangle + 0.4|1\rangle + 0.4|2\rangle \end{aligned}$$

Such machines can produce solutions with a negligible probability of error, which would be non-computable if no error was tolerated. Probabilistic computation with negligible error is important for quantum computation. In fact, it was shown in [ChuangNielsen 97] that nothing can be gained by quantum execution of a deterministic algorithm.

In the 1980s, Benioff and Feynman discussed using quantum physics to perform computation faster than classical Turing machines. In 1985, Deutsch proposed a universal quantum computer which could simulate any quantum version of a Turing machine, but suffered from exponential slowdown. Bernstein and Vazirani proposed an efficient universal quantum Turing machine in 1993 which could do the same, with only polynomial slowdown [14].

It has since been shown that any function computable in polynomial time by a quantum Turing machine has an equivalent polynomial-sized quantum circuit. Equivalence has been shown between quantum Turing machines, circuits, and cellular automata. Using these results, we have the basis for a universal quantum computer as modeled by a quantum Turing machine, which can execute general purpose quantum algorithms using a quantum programming language.

---

<sup>3</sup><http://www.nist.gov/dads/HTML/turingMachine.html>

## 4 Quantum Computation

We are concerned with using quantum computation to produce solutions to problems which are considered intractable for classical computers. An *intractable* problem is one for which execution time grows exponentially as input size increases linearly. To see how quantum computers can compute classically intractable problems, we need to examine how they perform computation.

The building block of the quantum computer is the *qubit*, or quantum bit. A qubit is an abstract value, represented by nanoscale properties of physical matter. Qubits have been implemented using ion charge, photon polarization, nuclear spin of atoms, and nuclear spin of impurities in silicon chips [16][12].

Classical bits abstractly represent values of 0 or 1 by the absence or presence of voltage. In contrast, a qubit represents both a 0 and 1 simultaneously due to superposition.

Since qubits represent two values at once,  $n$  qubits represent  $2^n$  values. Thus, quantum computers operate on exponential data with only linear input. For example: two qubits represent the values  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ , and  $|11\rangle$  simultaneously. A *probability amplitude* is a complex number representing the chance that any particular value will be observed.

Probability amplitudes are mathematically analogous to a wave: they can be positive or negative, and are subject to interference. The probability amplitude of a value is the sum of probability amplitudes over all entangled particles<sup>4</sup>. The classical probability of a value is given by the square of its probability amplitude. Quantum computation is performed by manipulating probability amplitudes of qubits.

A qubit can be represented by a unit vector in the Hilbert space  $\mathbb{C}^2$ , where  $\mathbb{C}$  is the set of complex numbers. An  $n$ -qubit system can be described by the tensor product of these Hilbert spaces:

$$i_1 i_2 \dots i_n \longleftrightarrow |i_1\rangle \otimes |i_2\rangle \otimes \dots \otimes |i_n\rangle \equiv |i_1 \dots i_n\rangle$$

We can compute a function  $f$  on these qubits by applying a unitary operator  $U$ :

$$|i_1 i_2 \dots i_n\rangle \longmapsto U|i_1 i_2 \dots i_n\rangle = |f(i_1, \dots, i_n)\rangle$$

When  $f$  is one-to-one, ie reversible, we can always find a unitary operator. Any classical function we wish to compute can be made reversible using extra bits and acceptable operational overhead [9]. We can think of  $U$  as a basic quantum gate, performing the role of a classical truth table[2]. We can represent  $U$  as a unitary matrix, which by definition is reversible.

To apply  $U$  and thereby compute our function  $f$ , we find the Hamiltonian  $\mathcal{H}$  which solves the equation:

$$|\Psi_f\rangle = \exp\left(-\frac{i}{\hbar} \int \mathcal{H} dt\right) |\Psi_0\rangle = U|\Psi_0\rangle$$

A solution exists as long as  $U$  is unitary. By applying  $U$  to a vector of  $n$  qubits, we apply  $f$  to all possible  $2^n$  inputs in parallel. This parallelism is how quantum computation overcomes classically intractable problems.

---

<sup>4</sup><http://www.reed.edu/~alan/Research/Bond/BasicQM/PrAmp/pramp.html>

Obtaining the computed solution presents a new challenge. The superposition of  $n$  qubits has  $2^n$  possible states, and is called a *wave function*. By observing the value of a vector of qubits we *collapse* this wave function, selecting one state from exponentially many to produce a classical bit vector. This collapsing effectively destroys the solution. However, we can use the interference property of probability amplitudes to cancel non-solutions during the collapse of the wave function. Used properly, this gives a high probability of observing the correct answer.

Quantum computing is not without drawbacks. Firstly, the exponential parallelism inherent in quantum computation does not imply exponential computational power [9]. In addition, practical implementation is hindered by decoherence, caused by interference from the environment as well as the exponential decay times of quantum states. Quantum error correcting codes are necessary to mitigate decoherence and error propagation in the real world. These codes require many physical qubits to accurately encode a single logical qubit of information [6]. Finally, certain quantum architectures have impractical physical requirements, such as operating at near absolute zero temperature. Current quantum computers require classical control components, and CMOS transistors cannot be used in such an environment [3].

Due to these difficulties, simulation is important for the development and advancement of quantum algorithms. Quantum simulators run on classical computers and allow state observation, analysis of error accumulation, and feasibility studies. However, simulating a  $2^n$ -dimensional Hilbert space on a classical system requires manipulating bit vectors of size  $2^n$ , whereas quantum computers need only  $n$  qubits. As a result, quantum simulators are restricted to small numbers of qubits [13].

Real quantum computers do exist which operate on 5-7 qubits, with 100-qubit devices in the works. Among others, Shor's algorithm was successfully demonstrated on a 7-qubit machine [1]. Such machines are currently built in the quantum circuit model, as general purpose machines are still theoretical. We will examine the architectural components of both machine types in the following sections.

## 5 Gates

In classical computers, any circuit can be made using AND ( $\wedge$ ), OR ( $\vee$ ), and NOT ( $\neg$ ) gates, or alternately NAND gates. These form a *universal set*. Another universal set is comprised of Toffoli gates [Toffoli 82] which allow reversible computation. Given any universal set of gates, wires to connect these gates, and the ability to duplicate values carried on these wires, we can perform computation.

Building a quantum circuit has similar requirements: a universal set of quantum gates, quantum wires, and the ability to copy values from these wires. Quantum gates can be represented by a unitary matrix of size  $2^n \times 2^n$  for a gate with  $n$ -qubit inputs. They must have the same number of inputs as outputs, and must be interconnected without fanout or feedback [4].

The two-qubit quantum gate called the Controlled NOT (CNOT) together with all single-qubit quantum gates forms a universal set [16] (see Figure 1). The CNOT gate inverts its second output qubit if its first input qubit is 1, and is analogous to a classical XOR. The three-qubit quantum Toffoli gate also forms a universal set (which, unlike its classical counterpart, requires no "workspace" bits [4]).

$$\begin{array}{l}
\text{X Gate} \\
\text{Bit-flip, Not} \\
\text{Z Gate} \\
\text{Phase-flip} \\
\text{Y Gate} \\
\text{T Gate} \\
\pi/8 \\
\text{H Gate} \\
\text{Hadamard}
\end{array}
\begin{array}{l}
\boxed{X} \\
\boxed{Z} \\
\boxed{Y} \\
\boxed{T} \\
\boxed{H}
\end{array}
\equiv
\begin{array}{l}
\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}
\end{array}
=
\begin{array}{l}
\beta|0\rangle + \alpha|1\rangle \\
\alpha|0\rangle - \beta|1\rangle \\
-i\beta|0\rangle + i\alpha|1\rangle \\
\alpha|0\rangle - e^{i\pi/4}\beta|1\rangle \\
\frac{\alpha + \beta|0\rangle + \alpha - \beta|1\rangle}{\sqrt{2}}
\end{array}$$
  

$$\begin{array}{l}
\text{Controlled} \\
\text{Not Controlled CNot} \\
\text{Swap}
\end{array}
\begin{array}{l}
\begin{array}{c} \bullet \\ | \\ \boxed{X} \end{array} \\
\begin{array}{c} \bullet \\ | \\ \oplus \end{array} \\
\begin{array}{c} \times \\ | \\ \times \end{array}
\end{array}
\equiv
\begin{array}{l}
\begin{array}{c} \bullet \\ | \\ \oplus \end{array} \\
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \\
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
\end{array}
=
\begin{array}{l}
a|00\rangle + b|01\rangle + d|10\rangle + c|11\rangle \\
a|00\rangle + c|01\rangle + b|10\rangle + d|11\rangle
\end{array}$$

Figure 1: A universal set of quantum gates [1]

While quantum gates have been implemented, there is currently no actualized quantum *arithmetic logic unit* (ALU). A quantum ALU would perform operations for computation and error correction [1]. These operations are simply sequences of unitary transforms, as implemented by a universal set of quantum gates (see Figure 1). The ALU would require direction from classical components.

Unfortunately, performing quantum computation is not as simple as using a quantum ALU to compute a function over some input. Total separation from the environment is not currently possible, making decoherence unavoidable. Errors caused by decoherence propagate quickly due to entanglement. To enable accurate computation the ALU must perform extensive error correction, discussed next.

## 6 Error Correction

Error correction has been said to be the single most important concept in quantum architecture [1]. Unlike classical bits, we must worry about both bit flips and phase shifts. Errors propagate and compound through each gate. Decoherence takes place at a rate per operation,  $p$ , ranging from  $10^{-3}$  to  $10^{-14}$  for current systems. This is the base probability for error, and is determined by the physical implementation of the qubits.

Recursion level ( $k$ )	Storage overhead $7^k$	Operation overhead $153^k$	Minimum time overhead $5^k$
0	1	1	1
1	7	153	5
2	49	23,409	25
3	343	3,581,577	125

Figure 2: Error correction overhead for a one-qubit operation using a [7,1] Steane code [1]

In addition, the fidelity of a qubit decays exponentially with time. Theoretic quantum memory would have to periodically refresh qubits, just as with classical RAM. An unexpected benefit of quantum error correction is the reduction of this exponential decay into first-order linear decay.

Correcting quantum errors is difficult, because measuring for error collapses the superposition of states. Superposition must be maintained throughout the computation. We cannot measure qubits for error without destroying their computed value.

Von Neumann showed how to perform classical computation with faulty components using error correction. His method encoded  $c \log(t)$  bits to redundantly encode a single bit. A similar approach is taken for quantum error correction. The qubits are redundantly encoded in a way which allows measurement of error, but not of the values themselves. By operating directly on encoded values, errors introduced by decoding are avoided. Universal computation on encoded values is possible [10].

Coding schemes are labeled as  $[n,k]$ , where  $n$  physical qubits encode  $k$  logical qubits. The extra  $n - k$  physical qubits, called *ancilla qubits*, come from an entropy exchange unit which produces qubits with an initial value of  $|0\rangle$ . The ancilla qubits are entangled with the desired physical qubit. By measuring the ancilla qubits, we collapse continuous errors into discrete errors: right/wrong amplitude and right/wrong phase. Amplitude and phase correction are performed by  $X$  and  $Z$  gates [6].

One such coding scheme is the Steane [7,1] code. A single qubit operation using Steane codes requires 153 physical quantum gates. The error probability then changes from  $p$  to  $cp^2$ . Since  $p$  is a fraction much less than one ( $10^{-3}$  for the Shor machine), for a reasonable  $c$  we have  $p < cp^2$ . We can apply this encoding recursively, so a second round of encoding would yield an error probability of  $c(cp^2)^2$ . This allows exponential decrease of error with polynomial overhead [1] (see Figure 2).

This result is important: we can use noisy quantum gates as long as errors are corrected faster than they are introduced [16]. This is known as the *threshold theorem*. Implementing a fault-tolerant circuit of size  $t$  requires gates with a maximum error rate of  $\frac{1}{\log(t)^c}$  [5]. An estimate of  $c$  for the Steane [7,1] code is  $10^{-4}$  [6].

We have seen how quantum gates perform computation with error correction, bridging the gap between theory and implementation. Building circuits from gates requires wiring, which poses a challenge at the quantum level. Quantum wires are discussed next.

## 7 Wires

Communication and computation have a fundamental tension in quantum systems [2]. Communication requires particles which do not interact with the environment, while computation requires highly interactive particles (not in the decoherence sense). As such, photon-based systems make communication easy and computation difficult, while nuclear spin-based systems are the opposite. Transferring states between different systems proves difficult, and so this tension must be accepted by architects.

Unlike bits, qubits cannot simply be copied and transmitted over a wire [2]. Quantum wires must destroy the source of information and re-create it at the destination. Wires cannot fanout or amplify information – they must be point to point.

The simplest quantum wire is a swap channel. A *swap channel* is a series of CNOT gates which relays a qubit, much like the classical XOR swap algorithm<sup>5</sup>. However, error is accumulated through each gate, making this method viable for short distances only. Swap channels require classical control mechanisms.

For movement of quantum values over longer distances, quantum teleportation is used. *Quantum teleportation* (see Figure 3) is not instantaneous as inferred by the name, but rather uses an entangled pair of particles along with two classical bits to re-create a quantum value at a destination. These particles, called an *EPR pair*, are in a *cat state* (after Schrödinger’s cat). This means their probability amplitude is  $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ . Since these particles are entangled, measuring one also determines the state of the other.

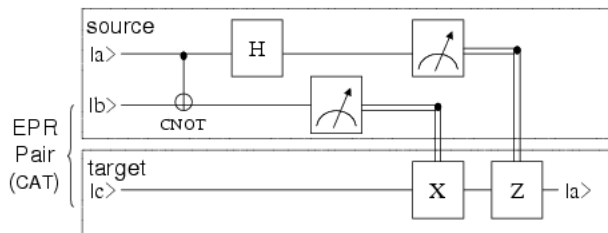


Figure 3: Quantum teleportation [2]

Say the qubit we want to teleport is called  $|a\rangle$ , and our EPR pair is  $|b\rangle$  and  $|c\rangle$ . The pair is separated,  $|b\rangle$  being kept at the source and  $|c\rangle$  sent to the destination. We are not yet concerned with error. The source qubit  $|a\rangle$  is then interacted with  $|b\rangle$  through a CNOT gate. Both are measured, and this measurement is sent as two classical bits over a wire to the destination. The interaction of  $|a\rangle$  and  $|b\rangle$  causes  $|c\rangle$  to resemble  $|a\rangle$  due to entanglement. The only concern is possible amplitude and phase errors in  $|c\rangle$ . The two classical bits allow correction of these errors using  $X$  and  $Z$  gates, if necessary. We have then destroyed  $|a\rangle$ , but re-created its state in  $|c\rangle$ .

There are additional benefits to teleportation. Pre-communication of EPR pairs is possible, and can be pipelined. In addition, teleportation allows conversion between error correcting schemes without introducing new errors.

<sup>5</sup>[http://en.wikipedia.org/wiki/Xor\\_swap\\_algorithm](http://en.wikipedia.org/wiki/Xor_swap_algorithm)

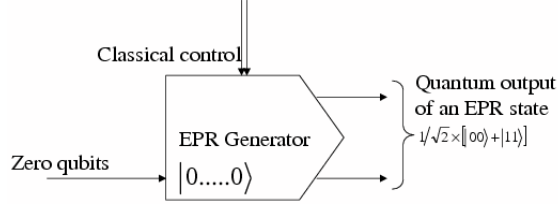


Figure 4: EPR generator [2]

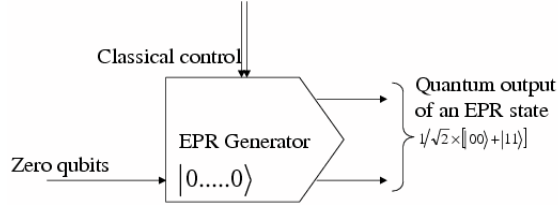


Figure 5: Purification unit [2]

Creating the EPR pairs involves an entropy exchange unit feeding an EPR pair generator. An *entropy exchange unit* gathers entropy from a subset of qubits in the system, leaving another subset in a near-zero state [2]. An *EPR generator* (see Figure 4) applies a Hadamard gate to two of these near-zero qubits, entangling them. A *purification unit* (see Figure 5) takes EPR pairs corrupted by error introduced through teleportation, along with near-zero qubits from the entropy exchange unit, and corrects their error. These are the building blocks which enable quantum teleportation.

## 8 Conclusion

With proper design, quantum computation exploits superposition and entanglement to achieve parallelism not possible in classical computers. Superposition enables quantum bits to exist in both a logical 0 and 1 state simultaneously, and may be implemented using different physical nanoscale properties. Entanglement allows measurement of one qubit to determine the value of another qubit across any distance, as long as the system remains coherent.

Specialized quantum circuits have implemented algorithms such as Shor's on a handful of qubits. General purpose quantum computers exist only in theory, but building blocks are in place. A universal set of quantum gates has been identified. Short distance wiring is achieved by swap channels, and long distance wiring uses quantum teleportation. Using universal gates and wiring, a quantum ALU may not be far off. Error correcting schemes enable implementation of real quantum systems, mitigating the unavoidable effects of decoherence.

In comparison to classical architecture, quantum architecture is still in its infancy and is dealing with many fundamental issues. Designing a radically new architecture from the ground up gives us a chance to incorporate the vast architectural knowledge generated since the dawn of classical computing, in order to overcome its limitations.

## References

- [1] M. Oskin, F. Chong, and I. Chuang, “A Practical Architecture for Reliable Quantum Computers”, *IEEE Computer*, Jan. 2002.
- [2] M. Oskin, F. Chong, I. Chuang, J. Kubiawicz, “Building Quantum Wires: The Long and the Short of it”, *International Symposium on Computer Architecture*, Jun. 2003.
- [3] M. Whitney, Y. Patel, N. Isailovic, and J. Kubiawicz, “Can we build Classical Control Circuits for Silicon Quantum Computers?”, *Second Workshop in Non-Silicon Computing*, Jun. 2003.
- [4] D. DiVincenzo, P. Shor et al, “Elementary Gates for Quantum Computation”, *Physical Review A*, Mar. 1995.
- [5] P. Shor, “Fault-Tolerant Quantum Computation”, *IEEE Symposium on Foundations of Computer Science*, 1996.
- [6] D. Copley, M. Oskin, F. Chong, I. Chuang, and K. Abdel-Ghaffar, “Memory Hierarchies for Quantum Data”, *Workshop on Non-Silicon Computing*, Feb. 2002.
- [7] A. Smith et al., “Secure Multi-Party Quantum Computation”, *Symposium on the Theory of Computing*, Sep. 2002.
- [8] P. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”, *SIAM Journal on Computing*, 1997.
- [9] D. Aharonov, “Quantum Computation”, *Annual Review of Computational Physics*, 1998.
- [10] D. Copley, M. Oskin, F. Chong, and I. Chuang, “The Effect of Communication Costs in Solid-State Quantum Computing Architectures”, *Symposium on Parallel Algorithms and Architectures*, Jun. 2003.
- [11] D. Copley, M. Oskin, F. Chong, I. Chuang et al., “Towards a Scalable, Silicon-Based Quantum Computing Architecture”, *Journal of Selected Topics in Quantum Electronics*, 2003.
- [12] S. Bettelli, T. Calarco, and L. Serafini, “Towards an Architecture for Quantum Programming”, *The European Physical Journal*, Nov. 2003.
- [13] J. Wallace, “Quantum Computer Simulators - A Review”, Oct. 1999.
- [14] A.C. Yao, “Quantum Circuit Complexity”, *IEEE Symp. Foundations of Computer Science*, 1993.
- [15] L. Grover, “A Fast Quantum Mechanical Algorithm for Database Search”, *ACM Symp. Theory of Computation*, 1996.
- [16] P. Shor, “Quantum Computing”, *Proceedings of the International Congress of Mathematicians*, 1998.