Quantum Computing for Computer Architects

Quantum Computing for Computer Architects
Tzvetan S. Metodi and Frederic T. Chong
www.morganclaypool.com

ISBN-10: 1598291181	paperback
ISBN-13: 9781598291186	paperback
ISBN-10: 159829119X	ebook
ISBN-13: 9781598291193	ebook

DOI 10.2200/S00066ED1V01Y200610CAC001

A lecture in the Morgan & Claypool Synthesis Series
SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE #1

Lecture #1
Series Editor: Mark D. Hill, University of Wisconsin, Madison

First Edition
10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Quantum Computing for Computer Architects

Tzvetan S. Metodi
University of California at Davis, Computer Science Department

Frederic T. Chong
University of California at Santa Barbara, Computer Science Department

SYNTHESIS LECTURES IN COMPUTER ARCHITECTURE #1

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Quantum computation may seem to be a topic for science fiction, but small quantum computers have existed for several years and larger machines are on the drawing table. These efforts have been fueled by a tantalizing property: while conventional computers employ a binary representation that allows computational power to scale linearly with resources at best, quantum computations employ quantum phenomena that can interact to allow computational power that is exponential in the number of "quantum bits" in the system. Quantum devices rely on the ability to control and manipulate binary data stored in the phase information of quantum wave functions that describe the electronic states of individual atoms or the polarization states of photons. While existing quantum technologies are in their infancy, we shall see that it is not too early to consider scalability and reliability. In fact, such considerations are a critical link in the development chain of viable device technologies capable of orchestrating reliable control of tens of millions quantum bits in a large-scale system. The goal of this lecture is to provide architectural abstractions common to potential technologies and explore the systems-level challenges in achieving scalable, fault-tolerant quantum computation.

The central premise of the lecture is directed at quantum computation (QC) architectural issues. We stress the fact that the basic tenet of large-scale quantum computing is reliability through system balance: the need to protect and control the quantum information just long enough for the algorithm to complete execution. To architect QC systems, one must understand what it takes to design and model a balanced, fault-tolerant quantum architecture just as the concept of balance drives conventional architectural design. For example, the register file depth in classical computers is matched to the number of functional units, the memory bandwidth to the cache miss rate, or the interconnect bandwidth matched to the compute power of each element of a multiprocessor. We provide an engineering-oriented introduction to quantum computation and provide an architectural case study based upon experimental data and future projection for ion-trap technology. We apply the concept of balance to the design of a quantum computer, creating an architecture model that balances both quantum and classical resources in terms of exploitable parallelism in quantum applications. From this framework, we also discuss the many open issues remaining in designing systems to perform quantum computation.

KEYWORDS

quantum computing, computer architecture, fault tolerance,
error correction, trapped ions, teleportation, qubit, quantum logic array.

Contents

Preface

Quantum computation (QC) may seem to be a topic for science fiction, but small quantum computers have existed for several years [1] and larger machines are on the drawing table [2]. These efforts have been fueled by a tantalizing property: while conventional computers employ a binary representation that allows computational power to scale linearly with resources at best, quantum computations employ quantum phenomena that can interact to allow computational power that is exponential in the number of "quantum bits" in the system. Architecting large scale systems to exploit this potential is the focus of this book. Our goal is to provide architectural abstractions common to potential technologies and explore the systems-level challenges in achieving scalable, fault-tolerant quantum computation. While quantum technologies are in their infancy, we shall see that it is not too early to consider scalability and reliability. In fact, such considerations are critical to guide the development of viable device technologies.

The central premise of this book is directed at architectural issues that arise during the design of QC system. We stress the fact that the basic tenet of *large-scale* quantum computing is *reliability through system balance*: the need to protect and control the quantum information just long enough for the algorithm to complete execution. To architect QC systems, one must understand what it takes to design and model a balanced, fault-tolerant quantum architecture just as the concept of balance drives conventional architectural design. For example, the register file depth in classical computers is matched to the number of functional units, the memory bandwidth to the cache miss rate, or the interconnect bandwidth matched to the compute power of each element of a multiprocessor. Through a detailed case study given in this book, we show that by applying the same concept of balance to the design of a quantum computer, it is possible to create an architecture model that balances components and both quantum and classical resources in terms of exploitable parallelism in the applications being executed.

We provide enough information and architectural analysis to enable the reader to continue the advancement of scalable quantum architecture research by identifying some of the key open questions. For example, what is the best way to integrate fault-tolerant scalable data storage structures, computational structures, scalable communication mechanisms, and classical schedulers that orchestrate the program execution. The reader should be able to identify the different tradeoffs between the various requirements for scalable quantum computation, and most importantly through clever systems design, work toward creating a quantum architecture that balances reliability, area, and time performance such that it is relevant and within the reach of future technological advances.

We shall discover that building large-scale quantum machines is extremely difficult. This difficulty is consistent with our intuition—quantum effects in nature are only observed at very small, carefully isolated physical systems such as single photons and atoms. Binary information can be stored in a single unit of quantum data, known as a *qubit*, in the distinct energy states of an atom for example, or the polarization states of a photon. Larger-scale systems naturally couple with the environment and exhibit the behavior governed by classical physics that is so familiar to our everyday experiences. A corollary of this observation is that the physics of quantum computation often defies our classical intuition and is responsible for both the potential power of QC and the difficulty in realizing reliable quantum computers. In this book, we will attempt to offer both some simple formalisms and intuitions to describe the fundamentals of quantum operations.

Despite substantial engineering challenges to implement and manipulate a number of qubits, experimental realizations have resulted in quantum machines with seven-qubit memory storage [1], and with 100-qubit machines on the drawing table [2]. To build a quantum machine of practical computational value, however, we must be capable of storing and orchestrating a system of tens of millions qubits. While the work in physics and device development has made significant progress, such a scalable machine must also involve a systems approach to design that brings together diverse expertise in architectures, compilers, and algorithms. System designers and architects have the opportunity to study and identify important technological constraints and design schemes for a truly scalable machine using existing technological models. Identifying viable system designs at this stage is critical for the success of computationally relevant quantum information processor as it will create a clear direction for testing, modeling, and building large-scale computers.

Following Feynman's famous [1] about the significant gap between classical computational models and quantum mechanical ones, the first model in the context of a *quantum Turing machine* was introduced by Benioff [3]. Subsequently, Deutsch [4, 5] described the quantum circuit model as a universal simulator for the quantum Turing machine with exponential overhead. Bernstein and Vazirani [6] followed Deutsch's work with the description of a universal quantum Turing machine constructed with only a polynomial overhead. A more comprehensive timeline of quantum computation is given in Appendix A. Since the construction of the universal quantum circuit model, the ability to control and manipulate quantum information through a sequence of gates has led to several quantum algorithms with substantial advantages over known algorithms with traditional computation. The most significant is Peter Shor's algorithm for factoring the

[1] The observation was made in Feynman's talk during the First Conference on the Physics of Computation held at MIT in 1981. Feynman noted that it is impossible to simulate the evolution of a quantum system efficiently on a classical computer.

product of two large primes in polynomial time [7]. The security of the widely used RSA public-key cryptosystem relies on the assumption that factoring large integers is very hard on conventional computers [8], where the best-known classical algorithms for factorization are super-polynomial [9].

Additional quantum algorithms include Grover's fast database search [10]; adiabatic solution of optimization problems [11]; precise clock synchronization [12]; quantum key distribution [13]; and recently, Gauss sums [14] and Pell's equation [15]. Commercially, quantum technologies have been shown to enable unconditionally secure communication[2] leading to the creation of companies offering real products [21, 22].

A practical large-scale quantum system that can utilize the full potential of these algorithms must be capable of reaching a system size of $S = KQ \geq 10^{12}$ logical operations, where K denotes the number of computational steps and Q denotes the number of computational units. The problem with sustaining such a large amount of quantum computation is that the quantum information carriers (the qubits) continuously interact with external noise sources and *decohere*, eventually losing their quantum data. In addition different quantum states are not mutually exclusive as different classical bitstrings are, but may interfere with one another. While this very interference (known as *quantum entanglement*) is responsible for the power of quantum computation, it also causes errors to spread exponentially fast across the entire system if care is not taken to limit the spread of errors at the microarchitecture level.

A key theoretical breakthrough in scalable QC was the development of a theory of fault-tolerant quantum error correction adopted from classical error correcting codes. Quantum error correction allows reliability of large systems to be arbitrarily increased through the application of exponentially increasing amounts of redundancy [23]. Similar to classical error correction, quantum error correction uses the state of two or more qubits and recovery operations to encode a single logical qubit. The redundancy is recursively increased through successive encoding of each first level qubit much like recursive 2D classical codes. In other words, k or more logical qubits might be encoded in the collective state of n physical qubits, which are in turn encoded again, and so on. To sustain reliable computation for an extended period of time through the application of noisy physical gates, logic gates must be implemented directly over the logical qubits without decoding in such a way that errors do not spread through the data interference patterns of the encoded states. Later in this book (Section 5.6), we discuss the implications of managing these somewhat horrifying overheads versus the potential benefits of quantum computation.

[2] Mathematically, quantum key distribution (QKD) has been proven unbreakable [16–19], but recent experimental observations have shown that the probabilistic nature of the protocols coupled with noisy devices used to send and receive the quantum states allow the attacker Eve, to break the system with high probability of success [20].

The physical implementation of large, redundant structures gives rise to another fundamental challenge: communication of quantum data across nontrivial distances. Consequently, we find that one of the greatest challenges toward the design of a large, practically useful quantum computer is finding an architecture that incorporates the required amount of fault-tolerance while minimizing communication and resources overhead.

We explore this challenge with several case studies based on trapped ion technology [24, 25]. Unlike other technology proposals, ion traps have known, physically realized universal elements for quantum computation with a clear scalable model. The system models we describe, however, are based on two key attributes found in many quantum technologies. First, we show that specializing architectural components into memory and computation elements is advantageous. Second, we rely upon the concept of quantum teleportation [26] for communication across long distances in the large-scale architecture. Given these two design choices, we describe a general abstraction for scalable quantum architectures in which the dominant cost is communication between compute and memory regions through teleportation. A compiler infrastructure for the scheduling of quantum computations would maximize usage of data while it is in the compute region and minimize movement in and out to memory. This problem is analogous to minimizing register spilling in conventional processors.

The key ideas computer architects can take way that are relevant for the system design of scalable quantum computers after reading this book are:

- Achieving "good" system performance is synonymous with the realization of a workable balance between reliability, communication resources, and latency of computation in a quantum architecture.
- Quantum information cannot be cloned, thus when transferred from source to destination along a quantum channel, it must be transferred in such a way that no trace is left at the source.
- Quantum information can be transported by physically moving the qubits, transferring the information to a shared medium such as a quantum bus or a secondary quantum system that allows efficient qubit movement, successive swapping between adjacent qubits, and finally through the concept of teleportation.
- The focus on reliability allows for interesting match-ups between various system components. For example, communication and computation can be overlapped at the system level due to the considerable resources and latency overhead spent for error correction during logic gate execution on encoded data.
- There are many different ways to implement universal quantum logic at the application level. Gate can be built into the communication protocols or applied on the quantum data in a traditional manner analogous to classical circuits.

- System balance and fault-tolerance rests on the balance between different encodings of the quantum data defined by the error correcting codes used across different regions in the architecture, where the cost and resources of transfer from one region to another are carefully determined by the executed application.
- Another critical balance issue is the balance between data storage and computing on encoded data through different levels of encoding. Both structures require the implementation of specially optimized fault-tolerant error correction structures.
- Memory hierarchy in classical computation is analogous to *code hierarchy* in quantum systems. The transfer from storage regions to computational regions may require transfer from one encoding of the data to another, not a transfer from one technology type to another.

The book begins with a brief background in Chapter 2 which compares the basic operations for quantum computation to the conventional computing scheme by focusing on computation rather than physics. We describe in some detail the concept of qubits, quantum logic gates, and other important components for quantum computing relevant to the circuit model for quantum computation. In Chapter 3 we introduce three high-level requirements for a scalable quantum architecture and describe each requirement independently in the following sections: reliable implementation technology in Chapter 4; efficient error correction schemes in Chapter 5; and efficient quantum resource distribution in Chapter 6. Modeling and simulating quantum computational structures and cycle-level quantum simulation methods are described in Chapter 7, including a brief introduction of the stabilizer formalism for quantum computation and error correction. A set of architectural elements for a quantum architecture is described in Chapter 8. The concept of *quantum memory hierarchy* is described in Section 8.2. In Chapter 9 we give a case study for a quantum architecture, the quantum logic array (QLA), based on our previous work [27, 28]. Chapter 11 offers a discussion into the alternate methods for achieving fault-tolerant universal quantum logic, namely performing quantum operation through the concept of teleportation. Finally, we conclude with Chapter 12, where we give a brief summary of what we have done. In Appendix A we give a timeline of quantum computation beginning with year 1973 when one of the first works on the subject was introduced by Alexander Holevo [29].

CHAPTER 2

Basic Elements for Quantum Computation

The theory of quantum information processing (QIP) uses quantum mechanical two-level systems such as the two spin states of spin 1/2 atoms, or the horizontal and vertical polarization states of a single photon to store and manipulate binary information. Such systems are used to describe the single unit of quantum data known as a qubit [30] whose two states are distinguished as the binary states "0" and "1." One of the distinguished features of QIP from classical computational theory is that the permitted states of a single qubit fill a two-dimensional vector space and can be written as the superposition of the two binary states "0" and "1." In this manner, the state of an n-qubit quantum register spans a 2^n-dimensional vector space as the superposition of all of the possible 2^n binary bitsring states. An n-bit logic operation is permitted to act on one or all possible bitstring states of the register in a single clockstep. Thus, an exponential increase in the processed information at each clockstep is paralleled by a polynomial increase in the data size manipulated.

The result of measuring a single qubit is one of the two possible states "0" or "1," while before measurement the value of the qubit is a probabilistic distribution over both possibilities. This is consistent, for example, with the discrete electronic states of an atom used in some technological proposals to physically realize a single qubit. Upon observation (i.e., measurement) of all the qubits in a quantum register the result is a single classical bitstring with an associated probability. Quantum gates used for computation that change the qubit states are such that they intrinsically operate on both possibilities because it is not known which exact state the qubit is in before it is measured. This is why a sequence of quantum gates used to calculate a given function over an n-qubit register must operate on all 2^n possible states of the register at each clockstep. The power of quantum computation is derived from the fact that the probability amplitudes of each possible bitstring state of a quantum register are not mutually exclusive, but are correlated and can be simultaneously fed as input to a given function. A single gate operating within one clockstep may transform the possible bitstring outcomes simultaneously such that the result upon measurement is an answer to some global property of the computed function over all inputs.

Perhaps, one of the closest classical analogies for the mechanics of quantum computation is classical probability theory. A single qubit can be regarded as a weighted coin that is spinning in the air. At any point in time it can land either "*heads*" or "*tails*" with probability that depends on the weighted factor of the coin; however, while in the air, we don't know what the state of the coin is and we can fairly assume that it is in a probabilistic distribution of both "head" and "tails". The action of the coin falling on the ground is synonymous with measuring a qubit, where we have destroyed the uncertainty of the "spinning" coin by not only knowing what the state is, but the coin has physically stopped at that state. Now, consider an n-qubit quantum register represented by n coins that are spinning in the air. While the coins are in the air, one needs 2^n different probability amplitudes to describe all the possible landing outcomes. The act of measuring each qubit in the register can be equated with the act of any one of the n coins falling to the ground, in either case the possible number of outcomes reduces by a factor of 2.

What quantum mechanical computation allows us to do, is to change the weighted properties of the coins *while* they are in the air. In a sense, we are changing the probabilities of the possible outcomes of the coins in a controlled, stepwise manner. The change is performed through specific transformations of the probability distribution vector that describes the probabilities of each of the possible outcomes of the spinning coins. Of course, the probabilistic distribution exists only while the coins are spinning, and just as gravity pulls them down and limits their spinning time, so do quantum systems are subject to decoherence forces that destroy their probabilistic distribution[1].

In this section we give a general overview of the background for quantum computation that will provide the reader with an understanding of how quantum data is stored and manipulated. The fact that measurement collapses the probabilistic distribution of all possible states of a quantum register giving us a single outcome, forces us to suspect that the notion of *quantum parallelism*, where a function $f(x)$ can be evaluated simultaneously for a number of inputs, is a somewhat oversimplified interpretation of the power of quantum computing. Even if we do evaluate all inputs in a single clock cycle we can only extract one result and irreversibly lose the rest. So, how is quantum computation more powerful than classical computing, namely, how is it different than randomized classical algorithms? The answer for this question can be most easily demonstrated through Deutsch's quantum algorithm [4, 31] which we briefly describe in Section 2.3.2. The algorithm uses the fact that the possible alternatives in a quantum register can *interfere* with one another, giving us some global information about the

[1] We assume that our coin experiment is executed on earth. Moreover, we may be able to change the weighted properties of the spinning coins, by bombarding each side (while spinning) with material that will affect the landing outcome of the coin. The coin analogy is straight forward and may have been used in the past, however, the authors are unaware of such an occasion.

function $f(x)$ over more than one input. In other words, the possible coin alternatives derived from classical probability theory are mutually exclusive events, while quantum mechanics allows quantum states to interfere such that changing one of them will affect the probability amplitudes of all the states that that state interferes with.

2.1 BITS VERSUS QUBITS

Well-characterized quantum signal states are the first requirement for realizing a quantum computer [32]. Classical computing signal states are represented as a sequence of to form a single bitstring. In the memory of a computer bit strings are encoded tightly, where each bit occupies exactly one bit of storage. During computation the "1" bit is denoted as a rise in the voltage through a silicon gate, while the "0" bit is marked as a lack of such current. Thus a classical bit exists as one of two perfectly distinguishable states, "1" or "0."

A quantum information signal (a qubit) also distinguishes between two states denoted as $|0\rangle$ and $|1\rangle$ and can be utilized, for example, by the ground and exited states of a single atom or the horizontal and vertical polarization directions of a photon. The "$|\cdot\rangle$" notation, known as the *Dirac-Ket*, is used to denote a particular quantum state. The difference is that according to the laws of quantum mechanics a single qubit exists in a superposition of the states $|0\rangle$ and $|1\rangle$, whose general state $|\Psi\rangle$ can be written as

$$|\Psi\rangle = a|0\rangle + b|1\rangle. \tag{2.1}$$

The amplitudes a and b are complex coefficients which obey the rule that $|a|^2 + |b|^2 = 1$. The quantity $|a|^2$ is the probability that the qubit will be found to exist in the state $|0\rangle$ upon observation and similarly, $|b|^2$ is the probability that the qubit will be found to exist in the state $|1\rangle$. Without direct observation, however, the state of a single qubit spans a two-dimensional vector space defined by the two-element complex valued vector $[a, b]^T$, where the most general single qubit state $|\Psi\rangle$ can be written in a vector form as

$$|\Psi\rangle = a|0\rangle + b|1\rangle = a\begin{bmatrix}1\\0\end{bmatrix} + b\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}a\\0\end{bmatrix} + \begin{bmatrix}0\\b\end{bmatrix} = \begin{bmatrix}a\\b\end{bmatrix}.$$

The state of a quantum computer with a storage total of two qubits will describe a four-dimensional vector space where each dimension can be distinguished by the four bit strings: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, where an arbitrary state of the system denoted as $|\Psi\rangle$, is described by a four-element, complex-valued vector $[c_0, c_1, c_2, c_3]^T$:

$$|\Psi\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle, \tag{2.2}$$

where the state is normalized such that the complex coefficients once again obey the restriction, $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 = 1$. Similarly, three qubits will be in a superposition of eight bit strings, encoding the numbers zero through seven in each bit string. Thus, computing a function $f(x)$ where $x = 0, 1, ..., 7$ can potentially be computed using three qubits with a *single* clock step. In general, an n-qubit quantum system may represent 2^n bit strings distinguished by 2^n complex-valued coefficients:

$$|\Psi\rangle = \sum_{i=0}^{2^{n-1}} c_i |x_i\rangle, \qquad \text{such that} \quad \sum_{i=0}^{2^{n-1}} |c_i|^2 = 1, \tag{2.3}$$

where each x_i represents the ith bitstring from 0 to 2^{n-1}. Because each additional qubit doubles the number of pure states (bitstrings) represented and subsequently manipulated by logic operations at each clock step, we can see how quantum computation has the potential to offer exponential scaling of the computing power with only a polynomial increase in the data resources.

2.2 LOGIC OPERATIONS AND CIRCUITS

A circuit in both classical and quantum computation is made up of *wires* and *logic gates*. The classical circuit model of computation is composed of acyclic circuits with a number of input and output bits which travel as an electric current, typically through copper wires, and whose states are modified by logic gates (such as the logically universal NAND gate). The actions of classical gates on bit strings are defined by boolean algebra. No matter what classical gate is executed, the fundamental operation type is a decision whether the value of one or more bits will be flipped.

A single gate in a quantum circuit with one or more input qubits in the initial state $|\Psi\rangle$ transforms the state to a different state $|\Psi'\rangle$ by changing all probability amplitudes that describe the state vector $[c_0, c_1, \ldots, c_{n-1}]^T$. Mathematically, linear algebraic operations such as matrices act on quantum mechanical vector states, thus a quantum gate on an n-qubit system is described by a $2^n \times 2^n$ matrix U that acts on all 2^n elements of the state vector $|\Psi\rangle$:

$$U|\Psi\rangle = U \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} c'_0 \\ c'_1 \\ \vdots \\ c'_{n-1} \end{bmatrix} = |\Psi'\rangle, \tag{2.4}$$

where the matrix U must ensure that the elements of the amplitude vector that describes the new state $|\Psi'\rangle$ satisfy the normalization property: $\sum_{i=0}^{n-1} |c'_i|^2 = 1$. The sum of the squares of the absolute values of a vector is known as the p-norm of a vector, and the only operators that map a vector of p-norm equal to 1 to another vector of p-norm equal to 1 are *unitary* operators. Thus, a quantum operator U is mathematically implemented as a unitary matrix. Because the inverse of a unitary operator always exists, applying U^{-1} to $|\Psi'\rangle$ will restore the state back to

$|\Psi\rangle$, thus all quantum logic is *reversible*. To preserve reversibility, an n-qubit input quantum operation must also have n output qubits.

The most general 2×2 operator U that acts on an arbitrary two-element vector that describes the state of a single qubit is a rotation matrix written as

$$U = \begin{bmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{bmatrix} \times \begin{bmatrix} \cos\theta/2 & \sin\theta/2 \\ -\sin\theta/2 & \cos\theta/2 \end{bmatrix} \times \begin{bmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{bmatrix}, \tag{2.5}$$

where the values α, β, and θ denote the angles of rotation along the different degrees of freedom. A valid rotation of the state of a single qubit can be arbitrarily small; thus there are an infinite number of possible operations that can be applied on a single qubit. Classical computation distinguishes itself with the fact that there is only one valid operation on a single bit, namely the bit-flip operation.

The overall function of the sequence of operations in an entire n-qubit quantum circuit divided into K time steps can be collectively described by a $2^n \times 2^n$ unitary operator U, where $U = U_k \times U_{k-1} \times \cdots \times U_1$. Each U_i is the $2^n \times 2^n$ unitary operator that describes the ith time step in the circuit and the collective action of the sequence is the product of all individual operators for each time step. A schematic of a quantum and a classical circuit is shown in Fig. 2.1. Fig. 2.1(a) shows a classical circuit with 3 input bits and 1 output bit. The output bit is the result of a boolean function that describes the classical circuit defined in this case by

$$f(c_1, c_2, c_3) = (c_1 \oplus c_2) \vee (c_1 \wedge c_3). \tag{2.6}$$

Given the value of the output bit and the gates performed in a classical circuit, it still may not be possible to know the values of the input bits. A quantum circuit, on the other hand, as shown in Fig. 2.1(b), has exactly as many input qubits as output qubits. In the shown schematic, time moves from left to right, where each line represents the evolution of a single qubit through the sequence of gate cycles in a circuit. The input quantum state $|\Psi\rangle = |q1, q2, q3\rangle$ is transformed

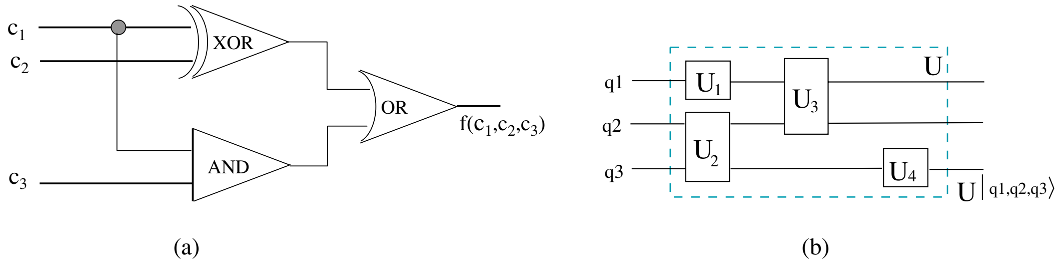


FIGURE 2.1: (a) An example classical circuit, where the output bit is a function of the input bits $\{c1, c2, c3\}$. (b) A three-time-step quantum circuit, where three input qubits $\{q1, q2, q3\}$ implies the same qubits as output. The notation $U^{(i)}$ denotes an i-qubit operator.

by multiplying $|\Psi\rangle$'s state vector by the 8×8 operator U which describes the evolution of the qubits' state through the smaller suboperators $\{U_1, U_2, U_3, U_4\}$ that make up the circuit.

The operator that describes the first time step in the circuit is the *tensor* product of the two matrices U_1 and U_2, while the operator that describes the last (third) time step is the tensor product of two identity matrices that leave qubits $q1$ and $q2$ unchanged and the one-qubit U_4 matrix. The tensor product is an operation denoted by the symbol "$\otimes$" between two matrices where each element of the first matrix is replaced by the second matrix, scaled by that element. For matrices U_1 and U_2 this is

$$U_1 \otimes U_2 = \begin{bmatrix} u_1^{(11)} U_2 & u_1^{(12)} U_2 & \cdots \\ u_1^{(21)} U_2 & u_1^{(22)} U_2 & \cdots \\ \vdots & \ddots & \cdots \end{bmatrix}, \tag{2.7}$$

which is an 8×8 matrix that describes the first time step. Thus, the final state of $|\Psi\rangle$ after the circuit completes is given by

$$|\Psi\rangle \rightarrow U|\Psi\rangle = [(I \otimes I \otimes U_4) \times (U_3 \otimes I) \times (U_1 \otimes U_2)]\,|\Psi\rangle, \tag{2.8}$$

where the state $|\Psi\rangle$ is first multiplied by $(U_1 \otimes U_2)$, then by $(U_3 \otimes I)$, and finally by $(I \otimes I \otimes U_4)$. The one-qubit matrix denoted by the letter I is the 2×2 identity matrix that does nothing on a single qubit

$$I \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{2.9}$$

Given the final state of a quantum circuit, applying the inverse of the operations in reverse will bring the state back to its input form.

There are three general ways to describe quantum operations in a quantum circuit as shown in Fig. 2.2. The leftmost schematic is of an arbitrary one-qubit operator U shown as a box with the letter "U" written inside. Similarly, an arbitrary two-qubit operator is shown as a box that spans two lines of qubit inputs with two lines of qubit outputs. The third (rightmost) schematic is of an arbitrary *controlled* operation, where the operator U is applied on the second

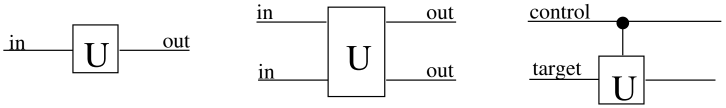


FIGURE 2.2: The three general ways to schematically represent quantum gates in the circuit notation: an arbitrary one-qubit operator U, and arbitrary two-qubit operator U, and a controlled-two-qubit operator where U is applied on the target qubit if the control qubit is set.

qubit whenever the state of first qubit is “1.” The logically universal set of quantum operations into which every n-qubit operator can be decomposed is shown in Eq. (2.10):

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \Phi_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}, \qquad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.10}$$

The first gate in Eq. (2.10) is the one-qubit Hadamard gate denoted with the letter H. The Hadamard gate takes the state $|0\rangle$ to the new state marked as $|+\rangle$ and the state $|1\rangle$ to the new state marked as $|-\rangle$. Each of the two states $|+\rangle$ and $|-\rangle$ is simply an equal superposition of the states $|0\rangle$ and $|1\rangle$ and is defined as:

$$|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.11}$$

$$|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.12}$$

The phase gate Φ_ϕ leaves the $|0\rangle$ element unchanged but applies a rotation of ϕ radians to the $|1\rangle$ state by multiplying it by the quantity $e^{i\phi}$. The Hadamard gate and the Φ_ϕ gate form a universal set of single-qubit gates, where any valid 2 × 2 unitary matrix can be approximated by these two gates. One can verify that multiplying any arbitrary two-qubit vector $[a, b]^T$ that describes the state of a single qubit by the matrix for the Φ_ϕ operator will result in a two-element vector with the a coefficient unchanged while the b coefficient will be multiplied by a factor of $e^{i\phi}$.

Finally, one of the most important gates in quantum computation is the two-qubit *controlled-NOT* gate or the CNOT gate, which allows the interaction between any two qubits. The CNOT gate flips the state of the target qubit if the control qubit is set. Its action on an arbitrary two-qubit state $|\Psi\rangle = (c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle)$ described by the vector $[c_0, c_1, c_2, c_3]^T$ is:

$$U_{\text{CNOT}}|\Psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_3 \\ c_2 \end{bmatrix}, \tag{2.13}$$

where the last two elements of the state vector have been flipped. The effect of the CNOT gate, where the first qubit is control and second is target, on the state $|ab\rangle$ is the new state $|a(a \oplus b)\rangle$:

$$|00\rangle \to |00\rangle; \quad |01\rangle \to |01\rangle; \quad |10\rangle \to |11\rangle; \quad |11\rangle \to |10\rangle.$$

Together, the gates in Eq. (2.10) form a logically universal set of quantum operations [33], much like the NAND gate is a logically universal gate for classical computation. Any n-qubit unitary transformation can be decomposed into a combination of only CNOT, Hadamard, and the phase Φ_ϕ gates, where the phase angle need only be $\phi = \pi/2$ or $\phi = \pi/4$ radians. The phase gates with angles of $\pi/2$ and $\pi/4$ radians are known as the S and T gates, respectively:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}. \tag{2.14}$$

A very important set of single-qubit gates known as the *Pauli* matrices is the four gates shown below denoted with the letters $\{I, X, Y, Z\}$:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad Y = -iZX = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \tag{2.15}$$

In fact, the phase-flip Z gate is the phase gate with the angle $\phi = \pi$ radians, while the X gate can be constructed by conjugating the Z matrix with the Hadamard matrix: $X = HZH$; and the Y gate can be obtained by multiplying the X and Z gates together with a global phase factor of $-i$: $Y = -iZX$. The X gate is the *bit-flip* gate which takes the state $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, and the Z gate is a 180° rotation of the phase known as the *phase-flip* gate which leaves $|0\rangle$ unchanged and takes $|1\rangle$ to $-|1\rangle$. The CNOT gate defined in Eq. (2.10) is nothing more than a controlled-X gate.

Consider the three-qubit circuit example shown in Fig. 2.3. The example has only two CNOT gates and two Hadamard gates, but it is an integral part of one of the most important concepts of quantum computation: teleportation [26], which we will describe in better detail when we introduce the measurement process of quantum states. The controlled-*NOT* gate's circuit representation is uniquely drawn to mark the fact that the gate's function is to perform the XOR operation between the control qubit and the target qubit. The collective state of the three qubits after any time step in Fig. 2.3 is described by an eight-element vector. The first time step involves the Hadamard gate on qubit $q2$ whose completion is necessary before we can execute the CNOT gate between $q2$ and $q3$ which marks the second time step.

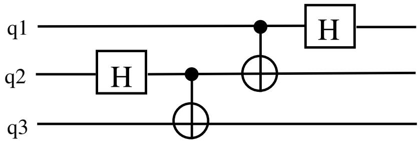


FIGURE 2.3: Example circuit consisting of two Hadamard gates and two CNOT gates.

Suppose now that the input state of the first qubit in Fig. 2.3 is an arbitrary qubit state $a|0\rangle + b|1\rangle$ and the other two qubits $q2$ and $q3$ are both initialized to $|0\rangle$. Thus before the first time step, the state of the entire system is

$$(a|0\rangle + b|1\rangle)|00\rangle = a|000\rangle + b|100\rangle, \tag{2.16}$$

where the ith entry in each bitstring state $|x_n x_{n-1} \cdots x_i \cdots x_0\rangle$ denotes the state of the ith qubit. The eight-element probability amplitude vector that describes the state of the system has all zero entries except the zeroth entry (equal to a) and the fourth entry, equal to b.

The combined state of the three qubits after the first Hadamard gate on the second qubit in Fig. 2.3 is now

$$\frac{1}{\sqrt{2}}(a|000\rangle + a|010\rangle + b|100\rangle + b|110\rangle). \tag{2.17}$$

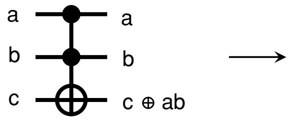
The first CNOT gate flips the state of qubit $q3$ at each bitstring state where $q2$ is set; thus the state of the three qubits becomes $\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle)$ after the first CNOT gate. After the second CNOT gate the state becomes $\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|110\rangle + b|101\rangle)$. The application of the Hadamard gate on qubit $q1$ places the final state of the three qubits into the superposition:

$$\frac{1}{2}(a(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + b(|010\rangle + |001\rangle - |110\rangle - |101\rangle)), \tag{2.18}$$

where we have factored out of common terms the probability coefficients a and b. The global phase factor of $\frac{1}{2}$ introduced by the successive application of the two H gates can be left out since it does not functionally change the probability values for the coefficients for each state relative to the other states. The coefficients are phase factors that can be moved to any location of their corresponding state. For example $a|00\rangle$ can be written as $|0\rangle a|0\rangle$. We can rewrite the final state of our example circuit by factoring out some common terms and moving the coefficients around to get

$$|00\rangle(a|0\rangle + b|1\rangle) + |01\rangle(a|1\rangle + b|0\rangle) + |10\rangle(a|0\rangle - b|1\rangle) + |11\rangle(a|1\rangle - b|0\rangle). \tag{2.19}$$

Note that the state of qubit $q3$ is any of four different arbitrary qubit states that look very much like the initial state of qubit $q1$. Thus, the state of qubit $q1$ has been recreated in qubit $q3$ with some error without directly interacting the qubits. The error depends on the values of qubits $q1$ and $q2$. In the next section we will see how extracting the values of qubits $q1$ and $q2$ can be performed through measurement and its effects on the overall state of the system.



$$U_{toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

FIGURE 2.4: Circuit representation and the three-qubit 8 × 8 matrix for the Toffoli gate.

2.2.1 The Quantum Toffoli Gate

The Toffoli gate in classical computation is defined as the *controlled-controlled-NOT* gate, which flips the state of the target bit if the states of both control bits are set: $(a, b, c) \rightarrow (a, b, c \oplus ab)$. In classical computation, the Toffoli gate is particularly important because it is the smallest universal, reversible classical operation [34]. It is universal because it can simulate the NAND gate if the third bit is fixed to 1 at input. It is reversible, because applying the Toffoli gate again will bring the state of the three bits back to (a, b, c), a property which cannot be implemented with any two-input one-output classical gate. Quantum mechanically, the Toffoli gate has the following action on a quantum bitstring state: $|abc\rangle \rightarrow |ab(c \oplus ab)\rangle$

$$\begin{array}{llll} |000\rangle \rightarrow |000\rangle; & |001\rangle \rightarrow |001\rangle; & |010\rangle \rightarrow |010\rangle; & |011\rangle \rightarrow |011\rangle \\ |100\rangle \rightarrow |100\rangle; & |101\rangle \rightarrow |101\rangle; & |110\rangle \rightarrow |111\rangle; & |111\rangle \rightarrow |110\rangle. \end{array}$$

The circuit representation for the Toffoli gate is shown in Fig. 2.4 along with the gate's 8 × 8 unitary matrix.

The Toffoli gate is an integral component in the implementation of almost all important quantum algorithms and it offers an important contrast between classical and quantum computations. A classical simulation of the Toffoli gate using one- and two-qubit gate can never be reversible, while quantum mechanically, we can completely describe its action and preserve its reversibility using *only* the universal gate set discussed in Section 2.2.

A circuit that implements the Toffoli gate made of just one- and two-qubit quantum gates from the universal gate set (Hadamard, Phase, and CNOT) is shown in Fig. 2.5. The ability to breakdown the Toffoli gate into one- and two-qubit operations is important, for there is no quantum technology implementation which allows the natural construction of a physical gate mechanism that implements a three-qubit operation. In the circuit, the gate labeled as $T^{\dagger}$ is simply the complex conjugate of the matrix that implements the T gate shown in Eq. (2.14).

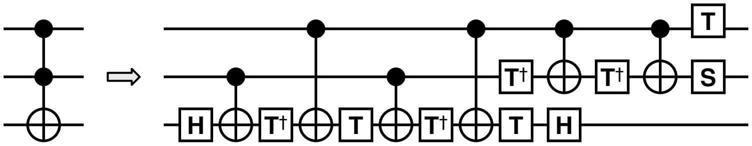


FIGURE 2.5: Circuit implementation of the Toffoli gate using only one- and two-qubit gates.

As an additional example for the usage of the Toffoli gate, the construction of a reversible 2-bit adder using Toffoli and CNOT gates is shown in Fig. 2.6. The circuit adds the two bitstrings "(x_1, x_2)" and "(s_1, s_2)," where the least significant bit is the leftmost bit. The result is stored in the bitstring "(s_1, s_2, C)," where C is the carry-out bit. For example, the addition of the input strings "(1, 1)" and "(1, 1)" should yield the result "$(s_1 = 0, s_2 = 1, C = 1)$." An additional ancillary bit is used. If the information were stored in qubits, the circuit in Fig. 2.6 becomes a quantum 2-bit adder based on the classical ripple-carry adder. If the input is a superposition of all possible combinations for the input strings, then the output would be a superposition, where each state holds the result of the addition. Adders are used to construct the circuit for quantum modular exponentiation, which is the most computationally intensive component of Shor's factoring algorithm.

2.2.2 Quantum Fourier Transform (QFT)

Another key circuit structure is the implementation of the quantum fourier transform (QFT), which lies at the heart of Shor's factoring algorithm [7]. The factoring algorithm works by using a reduction of the factoring problem to finding the period r of the periodic function $f(x) = a^x \bmod M$, where a is a randomly chosen number co-prime to M, x is an integer in $\mathcal{Z}_{2M^2}$, and M is the number being factored. By far, the dominant part of the algorithm is the modular exponentiation routine, which computes $f(x)$ in superposition, over all values of x. The quantum Fourier transform enables us to compute the period r of $f(x)$ which we can use to classically deduce the factors of M.

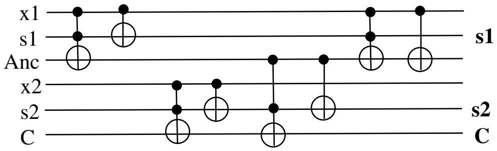


FIGURE 2.6: Two-Bit Adder composed of quantum controlled-NOT (CNOT) and Toffoli gates. The circuit adds the two bitstrings "(x_1, x_2)" and "(s_1, s_2)," where the least significant bit is the leftmost bit. The result is stored in the bitstring "(s_1, s_2, C)," where C is the carry-out bit.

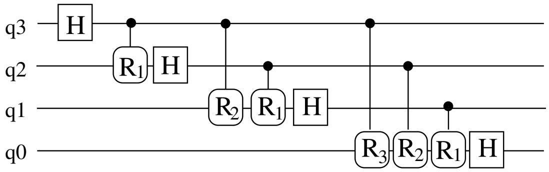


FIGURE 2.7: Circuit for the four-qubit quantum Fourier transform. Each controlled two-qubit gate R_i is a phase rotation with angles $\{\frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{8}\}$ for $i = 1, 2, 3$, respectively.

For some set of integers $\mathcal{Z}_N$ less than or equal to N, let the quantum state $|\Psi\rangle$ be in a superposition of all basis states $|x_i\rangle$ such that $x_i \in \mathcal{Z}_N$. The state $|\Psi\rangle$ is known to be in the *standard basis*, and the QFT is a unitary operator which maps $|\Psi\rangle$ to the *Fourier basis* which is a superposition of the basis states $|\chi_a\rangle$ defined by

$$|\chi_a\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i \frac{aj}{N}} |j\rangle. \tag{2.20}$$

Fig. 2.7 shows the circuit for a four-qubit QFT operator which can be synthesized into Hadamard and controlled phase gates. Each controlled two-qubit gate R_i is a phase rotation with angles $\{\frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{8}\}$ for $i = 1, 2,$ and 3, respectively. In general, if N is chosen such that it is a power of 2, then the QFT can be implemented on a quantum computer using $O((\log N)^2)$ gates [35]. The caveat is that the implementation of the controlled-phase gates shown in the circuit is not trivial. Song and Klappenecker [36] have shown that an arbitrary two-qubit controlled operator can be implemented with at most two CNOT gates and three single-qubit gates.

2.3 MEASUREMENT OF CLASSICAL AND QUANTUM STATES

Reading values from a classical register are a trivial operation. Values can be read reliably and copied to other registers. Unfortunately, this is not the case for quantum registers. Reading out the state of any qubit of a quantum register involves a measurement that destroys the superposition of that qubit, effectively terminating any quantum computation which requires that qubit.

It is important to state that, just as there are many unitary operators U that can be applied to a single qubit, there are many ways to perform a measurement on a qubit. A measurement can be performed along the eigenbasis of any one-qubit operator U, where the eigenbasis of a matrix consists of the *eigenvectors* of that matrix. An eigenvector $\vec{v}$ of operator U satisfies the equation

$$U\vec{v} = \lambda\vec{v}, \tag{2.21}$$

which means that the operator multiplied by its eigenvector equals the eigenvector scaled by some constant λ. The constant λ is known as the *eigenvalue* of U that corresponds to the particular eigenvector $\vec{v}$. When measuring a qubit, the only possible results of a measurement are the eigenvalues of the unitary operator describing this measurement.

For example, the eigenvalues of the phase-flip Pauli operator Z are "0" and "1" with corresponding eigenvectors $|0\rangle = [1, 0]^T$ and $|1\rangle = [0, 1]^T$. The eigenbasis of Z is known as the *computational basis* because its two eigenvalues are the two binary states "0" and "1," and so far, we have described qubits in the computational basis where the most general one-qubit state is $|\Psi\rangle = a|0\rangle + b|1\rangle$. When a single qubit is exposed to a Z measurement, the resulting classical bits will be "0" with probability $|a|^2$ and "1" with probability $|b|^2$. The value of the qubit's state after measurement is destroyed and forced to be equal to the result of the measurement (i.e., $|0\rangle$ or $|1\rangle$)—thus, losing all quantum superposition.

In some cases, however, the underlying technology may allow measurements along the basis of the X operator, which has eigenvalues "+1" and "−1." The eigenvectors of X are the states $|+\rangle$ and $|-\rangle$, where

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}0\\1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}, \tag{2.22}$$

where the arbitrary state $a|0\rangle + b|1\rangle$ can be written as

$$|\Psi\rangle = a|0\rangle + b|1\rangle = \frac{a+b}{\sqrt{2}}|+\rangle + \frac{a-b}{\sqrt{2}}|-\rangle. \tag{2.23}$$

Thus, when measuring the qubit in the X eigenbasis, the resulting state would be collapsed into the state $|+\rangle$ or the state $|-\rangle$ with probabilities equal to $\left|\frac{a+b}{\sqrt{2}}\right|^2$ and $\left|\frac{a-b}{\sqrt{2}}\right|^2$, respectively. For simplicity, we will consider measurement in the computational basis *only* throughout this publication and similarly, represent n-qubit states in the computational basis. Measurement along the X eigenbasis can be performed by applying the Hadamard gate followed by measuring along the computational basis.

Measuring an n-qubit quantum register in the computational basis gives a single bit string with probability calculated from the coefficient of the associated bit string, while completely discarding the rest. For example, measuring the two-qubit quantity $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ might yield the single bit string $|10\rangle$.[2] Thus one needs to be very careful during the computation

[2] With probability $\frac{1}{4}$, calculated by taking the square of the probability amplitude $\frac{1}{2}$.

of a quantum algorithm not to inadvertently expose the system to a measurement operation. Quantum algorithms are designed to apply a known sequence of gates on an initial quantum state such that the probability of measuring the correct answer at the end of the computation is greatest.

A very important by-product of the destructive nature of measurement of a quantum state is the inability to copy a quantum state, known as the *No-Cloning Theorem* [37]. Because states can be copied at will in classical computation, sending information to one or multiple destinations is easy. The state is just amplified through a FANOUT gate, placed on a wire, and sent to as many destinations as one needs provided a sufficient power source is available. In quantum computing however, we cannot copy a state, therefore the FANOUT gate is impossible. The inability to copy quantum states has great implications on our ability to communicate quantum information. We cannot simply transmit quantum information on a wire to a different destination, but can only transfer qubits without leaving a trace of the original one at the source location.

2.3.1 Quantum Teleportation

Quantum teleportation [26] allows us to transfer information from one qubit in location A to another qubit in location B without the need to locally interact the two qubits. If we look back to the example of Fig. 2.3 we see that upon measurement of qubits $q1$ and $q2$, we will obtain one of the four strings "00," "01," "10," or "11." If the result is "00" only the first state in the final superposition shown in Eq. (2.19) remains after measurement: $|00\rangle(a|0\rangle + b|1\rangle)$. Note that the state of just qubit $q3$ resembles the original state of qubit $q1$, where $q1$'s value has been collapsed to the pure $|0\rangle$ state due to the measurement operation. Thus upon observation of the string "00," and assuming perfect quantum gates and state preparation, the initial value of qubit $q1$ has been *teleported* to qubit $q3$ without having to interact qubits $q1$ and $q3$ *directly*.

Even if the result is any of the other strings "01," "10," or "11," one can see from Eq. (2.19) that the initial state of qubit $q1$ is recreated at $q3$ with some error. The error is fixed with a combination of a bit-flip X gate and a phase-flip Z gate. The full teleportation circuit complete with the two measurement operations on qubits $q1$ and $q2$ and the recovery X and Z operations on qubit $q3$ is shown in Fig. 2.8. At the end of the circuit, qubit $q1$ has been teleported to the location of qubit $q3$, while the no-cloning rule has not been violated since the original state at the location of qubit $q1$ has been destroyed by the measurement operation.

2.3.2 Deutsch's Quantum Algorithm

The discussion on the destructive nature of quantum measurement brings us back to the question of the power of quantum computing and more specifically, how can quantum parallelism be utilized? If all possibilities but one are destroyed when measuring a quantum register, then the

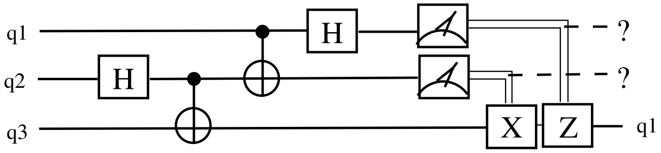


FIGURE 2.8: Complete circuit for quantum teleportation. The first four gates are the same as the ones shown in Fig. 2.3. The measurement operations and the recovery operation of X and Z gates are added to complete the teleportation of qubit $q1$ into the location of qubit $q3$. The "?"s indicate that we cannot predict the outcome of the two measurements with perfect accuracy but the final states of qubits $q1$ and $q2$ are collapsed to values that depend entirely on the probabilistic measurement outcome. In this manner, measurement is the only nonreversible quantum operation.

computational resources may seem to have been applied in vain. As mentioned at the beginning of this section, an explanation of why this is not true can be seen through the description of Deutsch's quantum algorithm [4, 31]. The algorithm uses the fact that quantum states interfere with one another, thus the probability of measuring a specific state is influenced by the values of all other states that this state interferes with.

Before we continue with Deutsch's algorithm, let us consider how quantum parallelism works when evaluating $f(x)$ as described in [38]. Suppose we start with two qubits in the initial state $|\Psi\rangle = |00\rangle$, and the two-qubit unitary transformation U_f which takes the state $|ab\rangle$ to $|a, b \oplus f(a)\rangle$. The transformation U_f can be simply the CNOT gate where $f(a) = 0$ if $a = 0$ and $f(a) = 1$ if $a = 1$. Applying the Hadamard gate on the first qubit we obtain the state

$$|\Psi\rangle \rightarrow H_1|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle). \tag{2.24}$$

The Hadamard gate is the key to accessing quantum parallelism as it transforms any state $|a\rangle$ into a superposition of all possible values of a, namely "0" and "1." After the unitary transformation U_f on the two-qubit state $|\Psi\rangle$ the state takes the form

$$|\Psi\rangle \rightarrow U_f|\Psi\rangle = \frac{1}{\sqrt{2}}(|0,\ f(0)\rangle + |1,\ f(1)\rangle). \tag{2.25}$$

Note that the state of $|\Psi\rangle$ contains the evaluated function $f(x)$ for both possible inputs "0" and "1," which we have derived with a single clockstep. The problem is that, upon measurement of one of the two qubits we will obtain information about the function $f(x)$ for only one input.

If instead, we start with the state $|\Psi\rangle = |01\rangle$ and apply a Hadamard gate to both qubits before we apply the unitary transformation U_f we have the state

$$|\Psi\rangle \rightarrow H_1 H_2|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.26}$$

Applying another Hadamard gate on the first qubit *after* the two-qubit U_f transformation, we obtain the following final state:

$$|\Psi\rangle_{\text{final}} = \pm|f(0) \oplus f(1)\rangle \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]. \tag{2.27}$$

We see that the state of the first qubit is the quantity $f(0) \oplus f(1)$ for the function $f(x)$, which is a global property of the function $f(x)$ that depends on both inputs. Unlike probabilistic classical computation where the two alternatives of $f(x)$ exclude one another, they have the option to interfere with each other in quantum computation. The interference was caused by the third Hadamard gate, which was applied to the first qubit in Deutsch's algorithm.

In general, the design of quantum algorithms involves the identification of a function $f(x)$ which possesses some global property over its inputs that is easy to achieve through some clever quantum transformation. The evaluation of $f(x)$'s global property should be chosen such that it will help us obtain a solution to a problem that is difficult to compute classically. One example, is using the Fourier transform to force a quantum state into a superposition such that all superposition states whose value is the period of some periodic function have a higher probability of being measured [39]. The calculated period is subsequently used to find the factors of a large number N in Shor's factoring algorithm [7].

2.4 QUANTUM ENTANGLEMENT AND EPR PAIRS

Three qubits are in a superposition of eight different states in Eq. (2.18) with varying probability amplitudes a and b. There is no way to rewrite the equation such that the state of any of the three qubits can be distinguished independently from the rest. Herein lies one of most amazing and powerful tool of QIP, the unique occurrence of *quantum entanglement*. Quantum entanglement is another way to describe the interference of different quantum states that is needed for quantum algorithms, by inseparably linking the superposition states of a collection of qubits.

Independently, each qubit is its own entity where there is some probability associated with obtaining either a $|0\rangle$ or $|1\rangle$ when measured and any logic gate can be applied on a single qubit. However, any action such as a gate or measurement on a single qubit will affect the states of the other qubits entangled with it. The exploitation of this enormously parallel interconnect has led to the application of entanglement to many of the most important quantum applications such as teleportation, quantum key encryption, and superdense coding [40, 41]. In addition, entanglement plays a major role in the difficulty of implementing quantum computation since a small error on one qubit is distributed across all qubits entangled with it.

The most important entangling gate between two qubits is the CNOT gate. Consider Fig. 2.9, where we begin with two qubits initialized at the state $|00\rangle$. The application of a Hadamard gate on the first qubit sends the system into the state: $(|0\rangle + |1\rangle)|0\rangle$, which can be

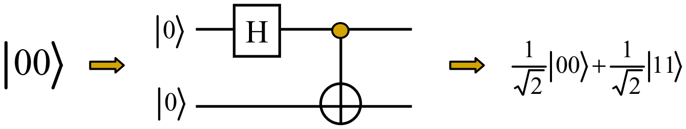


FIGURE 2.9: Creation of a maximally entangled EPR pair.

rewritten as $|00\rangle + |10\rangle$, where the first qubit is in an equal superposition of $|0\rangle$ and $|1\rangle$ and the second qubit remains $|0\rangle$. A CNOT gate with the first qubit as target flips the state of the second qubit only when the first qubit is $|1\rangle$ giving us the fully entangled state $(|00\rangle + |11\rangle)/\sqrt{2}$. Notice that in this case the unitary transformation U_f discussed in Section 2.3.2 *is* the CNOT gate. The fully entangled state is known as an EPR pair named after its discoverers, Einstein, Podolsky, and Rosen in 1935. The two qubits are completely correlated. If the first qubit is measured and we obtain the bit "0," then not only have we destroyed the state of the first qubit, but also the state of the second qubit, which would also yield "0" with almost near certainty if measured immediately after. EPR pairs are also known as two-qubit *cat states*. An n-qubit cat state can be generalized to $|\Psi\rangle = |00...0\rangle + |11...1\rangle$. An analogy for a four-qubit cat state using four cubes drawn without a particular frame of reference is shown schematically in Fig. 2.10.

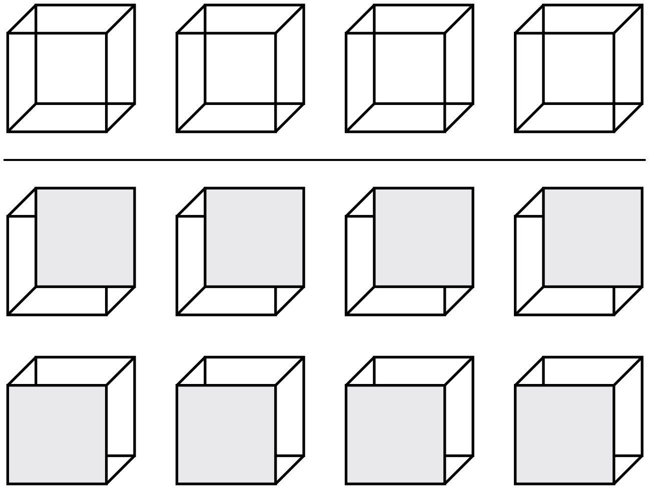


FIGURE 2.10: Four cubes that aid in visualizing a four-qubit cat state [42, 43]. The four cubes are initially drawn without particular frame of reference. The moment an observer is shown a *single* cube with the south face brought forward, or the north face brought forward, the observer's mind will be immediately fixed to the shown frame of reference for *all* cubes. Thus, showing an observer a particular frame of reference for one cube is equivalent to measuring not just the shown cube, but the entire entangled set of four cubes. The figure shows the two possible outcomes of observing either frame of reference in the bottom rows of four cubes each.

2.4.1 Teleportation (Revisited)

EPR pairs play an integral part in the quantum teleportation protocol described in Fig. 2.8. Note that the first Hadamard gate and the first CNOT gate are used to prepare an EPR pair between qubits $q2$ and $q3$, which is then entangled with the data qubit $q1$ through the second CNOT gate in Fig. 2.8. In general, quantum teleportation works by interacting an arbitrary data qubit with a previously prepared two-qubit EPR pair, such that the state of the data qubit is recreated into the state of one of the EPR qubits up to some error. Teleporting one qubit state from one location to another allows us to send quantum information through very large distances without directly distributing the information in the data qubit itself, but rather the physical distribution of EPR pair qubits between the source and destination. Even though EPR pairs are physically moved, they are replaceable and thus with enough quantum resources we may have a way to communicate valuable data at very large distances. After the EPR pair has been created one of the two EPR qubits travels to the data qubit and is entangled with it through the second CNOT gate in the circuit of Fig. 2.3. The other EPR qubit travels to the destination. The measurement result of the source qubit and its local qubit from the EPR pair yields the correcting X and Z operations that will recreate the data over the qubit that traveled to the destination.

CHAPTER 3

High-Level Architecture Criteria and Abstractions

The QIP model described in Chapter 2 follows the circuit model for quantum computation. In summary, the circuit model allows the execution of algorithms in the form of a sequence of operations applied on a number of qubits, where each qubit is a quantum system with the two states $|0\rangle$ and $|1\rangle$. We will restrict ourselves to a small set of universal quantum gates composed of any arbitrary single-qubit operation, the two-qubit controlled-NOT (CNOT) gate, and measurement. Other examples of computational models are adiabatic quantum computation [44, 45, 11], cluster state quantum computation [46–49], geometric quantum computation [50], and the theory of topological quantum computation [51]. In adiabatic quantum computation the computer is initialized with some initial Hamiltonian, H_i. H_i is then adiabatically deformed into a final Hamiltonian, H_f, that represents the solution to the problem being calculated. Cluster states are a collection of highly entangled qubits with the property that arbitrary quantum computation can be performed purely through single-qubit measurement operations. Topological quantum computation uses hypothetical quantum systems with particular kinds of topological excitations to avoid decoherence. Recent studies suggest that such systems may exist in nature. Combined, the variety of quantum computation models provides different methods for extending the application space for quantum computation, and may some day redefine the system design of a large-scale machine. In this book, we focus on the circuit model to describe a clocked, scalable quantum architecture scheme that overcomes the primary scalability issues of size and resource distribution. The model we describe is capable of performing any arbitrary quantum computation.

3.1 A HIGH-LEVEL ARCHITECTURE VIEW

The high decoherence rate of qubits forces us to design quantum architectures aimed at minimizing the time and spatial scope each qubit of data is used throughout the application, especially when quantum information is shuttled frequently without the ability to copy it. The physical

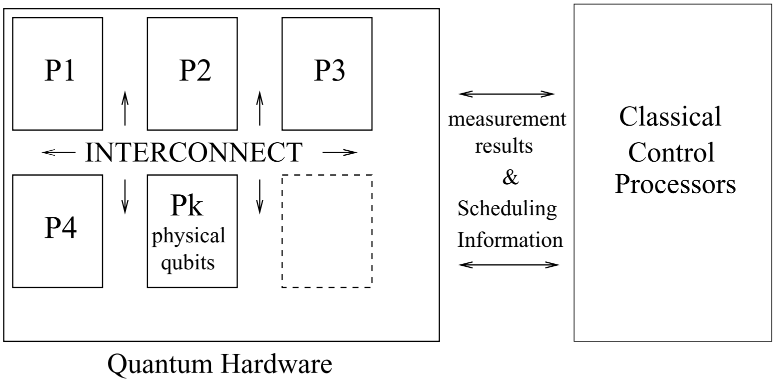


FIGURE 3.1: High level schematic for a quantum computer architecture. The computer model is composed of a number of processing elements composed of some collection of physical qubits. Each of the processing elements is designed to execute a localized piece of the larger application, and communication between processing elements is implemented through the teleportation-based interconnect. Classical control processors orchestrate the scheduling of quantum operations, where the only means of communication between the classical and quantum hardware is through measurement results.

model of a potential quantum architecture may consist of a number of qubit structures who exploit the principle of locality to compute and protect as much qubits as possible by limiting the transmission distance of the quantum data. The qubit structures can be connected by a carefully designed teleportation-based interconnect that allows information to be preserved over significantly large distances.

A high-level schematic of a quantum architecture is shown in Fig. 3.1. The architecture shown in the figure is composed of a number of processing elements. Each of the processing elements is designed to execute a *localized* piece of the larger application. Communication between processing elements can be implemented through the teleportation-based interconnect if the distances are too large, while communication within, is implemented through physical qubit movement as allowed by the underlying technology. Classical control processors orchestrate the scheduling of quantum operations, where the only means of communication between the classical and quantum hardware is through measurement results. A conventional quantum architecture compiler run by the classical control processors should have the freedom to fully orchestrate computation and communication in order to optimize the usage of quantum data such that the data is maximally protected through the course of the application. Quantum error correction codes are used to encode quantum data for continuous state stabilization through the application execution.

3.2 REQUIREMENTS FOR QUANTUM ARCHITECTURES

The trapped ion scheme proposed by Cirac and Zoller in 1995 was the first work that described a clear model for physically implementing a quantum computer in the laboratory [24]. Subsequently, DiVincenzo [32] from IBM put together a set of rules that generalized the task of the physical realization of a quantum computer and demonstrated that if the technology can satisfy the proposed rules then it could, in principle, be used to build a working computer. The quantum technology roadmap [52] uses DiVincenzo's requirements to describe the current state of existing technologies. The set of rules proposed by DiVincenzo can be summarized with the following four bullet points:

- A quantum register described as a collection of well-defined single-qubit states must be initialized to a well-known starting state (i.e., $|00 \ldots 0\rangle$).
- A "universal" set of quantum logic must be available, where the gate time cycle must be much shorter than the relevant decoherence time cycle of the quantum register.
- Reliable measurements must be performed on any single-qubit state.
- The ability to transmit quantum information between specified locations, either through the direct physical movement of the qubit, or by passing the information to "flying" qubits which can then pass it back to "stationary" qubits for gate manipulation.

In the next Chapter we describe in better detail the implications of DiVincenzo's requirements on the existing physical proposals for implementing a computer. There is a difference, however, between physically implementing the necessary components needed for a quantum computer, and designing a complete, large-scale quantum architecture that is intended to perform arbitrary computationally relevant programs. Previous work in large-scale quantum architecture design based on the circuit model [53–55, 27] has allowed us to extrapolate the chief requirements for building a large-scale quantum architecture. The scalability requirements can be summarized with three main bullet points, which are:

- Reliable and realistic implementation technology, that adheres to the DiVincenzo requirements [32] for implementing quantum computation.
- Robust, fault-tolerant structures encoded using efficient error correction algorithms. This requirement provides system-level fault tolerance that will allow the execution of an arbitrarily large sequence of universal quantum logic operations within the architecture decoherence time.
- Efficient quantum resource distribution at both the application level and the physical qubit level that allows maximum overlap of computation and error correction.

We review each of the three scalability requirements in detail in the subsequent Chapters. In Chapter 4 we review the existing technologies for realizing large-scale quantum computation and provide an overview of some of the components for trapped ion quantum computation and optical quantum computers. In Chapter 5 we build intuition about how quantum error correction codes can be used to build robust, fault-tolerant structures that allow the necessary scalability for large applications. Finally, we review the different meanings behind the notion of quantum communication in Chapter 6 and provide an idea of how data can be distributed in scalable quantum computers. Good system-level design choices will lead to a quantum computer where the underlying logic structures are synchronized such that the time cycles of refreshing, moving, and computing on quantum data are fully synchronized.

CHAPTER 4

Reliable and Realistic Implementation Technology

The basic quantum information processing (QIP) components described in Chapter 2 are carefully chosen to encompass the necessary low-level elements of the large-scale architecture framework we describe in this work. The physical realization of these components has gathered much attention in the recent years and realistic technologies have emerged that have made the concept of QIP a feasible prospect. There are now physical schemes that have demonstrated every major low-level architectural component needed for scalable computing. Even so, the limitations of existing QIP technologies are significant and the technology models vary to such an extent that identifying a clear winner at such an early stage may adversely affect future development. This is especially true for large-scale system design publications, which if carefully written have the potential to impact the direction of development for future computing machines.

The major limitations for technologies that can be used to build quantum hardware for a large-scale computer are two-fold:

- A number of qubits must be prepared and isolated from the environment such that they are protected from external forces that cause decoherence. Unlike classical bits which are robustly implemented through an electric current, a qubit may be contained in a single fragile ion [24] with very limited time before the qubit decoheres and loses its superposition. Limited coherence time for physical quantum states is the main reason for classical behavior in both the microscopic and macroscopic world.
- The second major difficulty for emerging quantum technologies is the fundamental inability to copy a quantum state combined with the need to perform logic operations and measurement on any one or a pair of qubits.

The second limitation is particularly difficult to overcome. Classical data can be replicated through a FANOUT gate and transmitted on wires from the memory elements to the processing units. An imperfect classical gate or a leaking wire may have some effect on parts of the classical state, but usually not enough to outweigh the multitude of electrons used to encode a single bit of information. On the other hand, to perform computation and apply gates on a number of qubits, we must be able to build them not only extremely weakly coupled to external decoherence forces, but be strongly coupled to each other and to an external gate device for the duration of a quantum logic gate. In addition, the transmission of the quantum information without the ability to leave any trace behind requires that information must be carefully guarded while physically moving.

Physical implementations of qubits that move the quantum information easily and ones that allow operations easily are two very contradicting concepts. A qubit defined by the polarization states of photons is ideal for movement because it does not interact easily with its environment easily and moves very fast. Photons, however, are hard to contain and two-qubit gates are very difficult to implement, since it is very hard to couple two photons. Heavy atoms are ideal for computation because they are relatively easy to slow down and apply operations on (usually by the application of lasers), but they are difficult to transport. A middle ground qubit is one that is not only exposed to the environment for computation, but also moves with relative ease and speed. Unfortunately, the qubit's ease of exposure also exposes it to uncontrollable forces from the environment both during computation and movement, making any choice for a qubit a choice with fundamentally limited reliability and decoherence time.

Physical realizations of the circuit model of quantum computation divide into several experimental proposals from very diverse fields of physical science, such as nuclear magnetic resonance (NMR) quantum computation [56, 57]; ion trap quantum computation [24] both optically through the coupling of neutral atoms with photons [58, 59] and physical segmented traps [60, 25, 61]; cavity quantum electro-dynamic (QED) computation [62]; optical quantum computation [63, 64]; solid state spin-based quantum computation [65–68]; quantum dots [69, 70]; superconducting quantum computation where the circuits are made with Josephson Junctions operating at millikelvin (0 kelvin = −273.15 celsius) temperature [71, 72]; and "unique" qubits such as electrons floating on liquid helium [73], the quantum Hall effect [74], and qubits encoded in the charge distribution of a single electron on two donors [75]. The distinguishing feature for all proposed technologies is the implementation of the qubit, which in turn guides the control infrastructure of the computer itself.

Each approach has different strengths and weaknesses for implementing a truly scalable computer. For example, the Kane technology where qubits are realized by the electronic states of phosphorus atoms embedded in a silicon substrate, has the advantages that it draws from existing investments in silicon fabrication techniques. Current measurement methods, however, can take as long as 4 days with a qubit lifetime of less than 60 ms, in addition to nonexisting laboratory gate implementations [68, 76]. In another well-developed work, qubits are held in pairs of energy levels of ions trapped in space by the electric potentials of metal electrodes [24, 25]. The ion-trap scheme is the only technology where every universal element for quantum computation has been realized with a clear scalable communication model [77, 78]. The caveat is that the ion-trap scheme is spatially expensive and it is not clear if it will remain a good technology for universal quantum computation in the distant future. It is important to realize that the importance of a certain technology must be judged as much for its potential promise as for its current experimental state. Reference [79] offers a very comprehensive review of the available technologies and their current parameters that are useful for building small computer prototypes. Here we give a brief description of two of the most successful experimental techniques so far: optical quantum computers and trapped-ion quantum computers.

4.1 OPTICAL QUANTUM COMPUTATION: PHOTONS AS QUBITS

The importance of photons as qubits is evident in their application to experimentally and commercially realizing quantum cryptography protocols [13, 80]. In addition, the proposal for quantum computation is based on photons as qubits [63], along with the fact that photon-based qubits are the first physical system used to experimentally demonstrate entanglement [81–83], teleportation [84–88], and various small-scale quantum algorithms [89–91]. The photon is the smallest physical unit for quantum information and has the advantage that it is virtually free of decoherence when implementing single-qubit gates and during transport. This stability is also the source of a severe experimental challenge, since quantum information tends to be "trapped" in the photon making two-qubit gates very difficult to utilize with sufficient success rate. Photons do not interact easily with each other and it was generally believed that they would be unsuitable for scalable quantum computation although ideal for quantum key distribution. The first experimental implementation needed exponential photon and control resources to achieve two-qubit gates and measurement of single photon qubits. An excellent review on optical quantum computation can be found as [52].

Knill, Laflamme, and Milburn developed a scheme in 2001 [64] which demonstrates that, in principle, it is possible to create highly efficient scalable quantum computers using linear optical components made up of phase shifters and beam splitters, single photons, and photo detectors with only a polynomial resource overhead. Before this scheme was proposed it had been shown that any unitary operator can be realized with linear optical components [92]. Single-qubit operators are relatively straightforward with beam splitters and phase shifters, which are mathematically described as 2×2 operators along the Z and Y axes of the qubit state representation. As shown in [38], any single unitary operator can be decomposed to a combination of Z and Y rotations.

For two-qubit gates, linear optics alone is not sufficient. Photo detectors are used to perform measurement, which combined with teleportation can be utilized to implement two-qubit operations. The most reliable CNOT implementation with linear optics succeeds with probability of nearly 7% [93], while single-qubit operations are virtually noiseless. The use of a generalized beam splitter combined with the Fourier transform can help bring the success of the teleportation procedure close to unity at the expense of exponential photon resources. Scalable quantum computation is possible, when the reliability of two-qubit gates through teleportation is reduced to a level such that quantum error correction can be utilized [93, 94].

Photon-based qubits offer ideal environment for distributed quantum computation. Photons are an attractive medium for shuttling quantum information from one part of the processor to another where the processor itself is composed of qubits that allow efficient quantum computation such as ion traps. Optical qubits offer by far the most advance experimental implementation for entangling two remote ions [43, 95, 96], or inducing qubit–qubit interactions between solid-state qubits using a common laser beam acting as a shared quantum bus [97, 98]. Entangling two remote qubits will allow the transfer of quantum information from one location to another through the teleportation procedure. Recently, linear optics quantum computation received a significant boost with the development of cluster state quantum computation [46–49], where initially entangled states are created that represent every necessary qubit resource in the architecture. Through single-qubit measurements on the entangled states, arbitrary quantum circuits can be simulated.

4.2 TRAPPED-ION QUANTUM COMPUTERS

Recent experiments with trapping ionized atoms in the form of trapped ions have shown so far the greatest promise for the development of quantum hardware capable of performing large-scale computations. Ion-trap quantum computation, initially proposed by Cirac and Zoller in 1995 [24], uses a number of atomic ions that interact with lasers to quantum compute. Quantum data is stored in the internal nuclear and electronic states of the ions, while the traps themselves are segmented metal traps (or electrodes) that allow individual ion addressing. The electrodes are placed typically on a 2D alumina substrate together with the needed electronics that control the trapping potentials. Two ions in neighboring traps can couple to each other forming a linear chain of ions whose vibrational modes provide qubit–qubit interaction used for multi-qubit quantum gates [99, 100]. Together with single–bit rotations this yields a universal set of quantum logic. All quantum logic is implemented by applying lasers on the target ions, including measurement of the quantum state [101, 60, 25, 102]. Multiple ions in different trap arrays can be controlled in parallel by focusing lasers through MEMS mirror arrays [2]. Additional *sympathetic cooling* ions are used to absorb unwanted vibrations from data ions, which are then dampened through laser manipulation [103, 59].

Fig. 4.1 shows a schematic of the physical structure of a trap element in an ion-trap computer. In Fig. 4.1(a) we see a single ion group trapped in the middle trapping region. An ion-group will be abstracted as an inseparable pair of a data ion and a sympathetic cooling ion

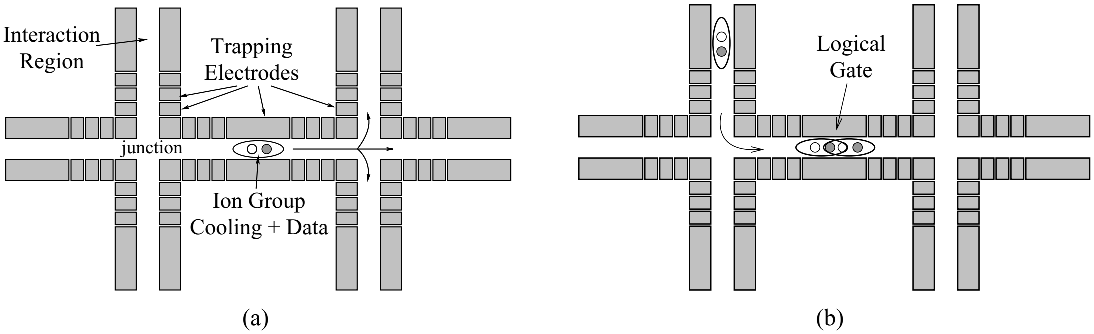


FIGURE 4.1: (a) The physical structure of an ion-trap quantum computer. An optimistic size of the trapping electrodes is in the order of tens of micrometers [105]. The data ion is kept together with a cooling ion and cooled before and after each movement step or logic gate. The ion-group can move to any of the six adjacent trapping regions for interaction with another ion group. (b) A two-qubit gate sequence, where the ion group in the top left junction moves to the middle for a two-qubit gate. The gate is implemented with an external laser beam acting on the two ion-groups.

that will always move together. In reality, it may be technologically unfeasible to implement reliable two-qubit quantum operations with the cooling ions present between the data ions, in which case the cooling ions must be provided separately. The cooling ions are needed to absorb the vibrational heating of the ion qubits. Trapping regions are the locations where ions can be prepared for the execution of a logical gate, which is simply an external laser source shining on the ion group. Fig. 4.1(b) demonstrates an ion group moving from the top left trapping region to the middle for the execution of a two-bit logical operation. A fundamental time step, or a clock cycle, in an ion-trap computer will be defined as any logical operation (one-bit or two-bit), a basic move operation from one trapping region to another, and measurement. It has been suggested in the literature that optimistic expectations for the failure rate of fundamental operations in ion-traps are on the order of 10^{-7} [104], and the time duration is of approximately 10 μs [52, 105]. This time is sufficient for the absorption of cooling and additional join and split operations needed for each fundamental operation [105].

4.2.1 Scalable Ion-Trap Model

Recent experiments that realize quantum teleportation using trapped ions [77, 78] have demonstrated all the necessary elementary components needed to build a large-scale ion-trap processor such as ions trapped in segmented electrode structures, laser induced ion cooling and manipulation, measurement using a pump laser that causes a state-dependent scattering of photons from the ion, and finally the ability to move ions around by changing the trapping potentials. In reference [105], Steane combines the increasing confidence in the experimental methods for laser controlled trapped ions with the quantum error correction requirements for a scalable, computationally relevant quantum computer to outline a natural ion-trap model that is experimentally feasible and does not omit any significant technological challenges. The computer is based on the quantum charge coupled device (QCCD) architecture proposed by Kiepinski *et al.* [25] that describes a scalable ion-trap design by creating a linear array of ion traps such as the ones shown in Fig. 4.1. The QCCD structure is intended to keep the number of ions chained together in a single trapping region as small as possible to avoid the technical difficulties in preserving and manipulating large chains of ions. Ions in different interconnected trap arrays can be *ballistically* shuttled from trap to trap by changing the trapping potentials in the electrodes. This allows the interaction of any two ions in the system, provided that the accumulation of errors during ion shuttling does not destroy the state of the stored qubit in each ion.

A schematic of Steane's ion-trap computer is shown in Fig. 4.2. His model of the ion chip is composed of ions trapped between segmented gold electrodes deposited on aluminum substrate [106]. An electrode structure that allows greater scalability as outlined in [2] are the planar ion traps where the ions are trapped above a set of individually addressable electrodes in a plane [107–109]. The planar traps allow the ions to "float" above the surface of the electrodes, thus allowing greater freedom for the angles at which the lasers can enter the vacuum chamber that holds the ion chip. The ion-trap electronics that are marked under the ion chip in Fig. 4.2 allow the control of the trapping voltages from one trap location to another which in turn allows the controlled shuttling of ions from one trap location to another. The implementation of a

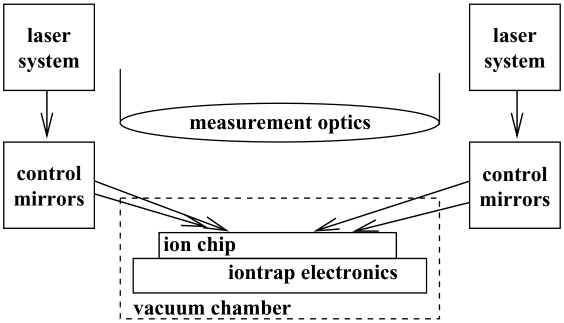


FIGURE 4.2: Schematic for a scalable ion-trap computer as shown in [105]. A large number of ions are trapped in an ion-trap chip whose implementation is suitable for efficient communication of ions around the chip for qubit–qubit interactions during the course of the algorithm. The ion chip rests in a vacuum chamber together with specialized electronics that control ion motion through the trapping electrode voltages. Qubit manipulation such as preparation, measurement, various logic gate implementations, and ion cooling are implemented with the different laser pulses generated by the laser system. The laser beams are distributed to different regions of the ion chip through the mirror control system.

high-density control electrode interconnects that allow the individual control of millions of trap locations remains a significant technological challenge. As described in [2], control voltages can be supplied vertically to the trapping electrodes through the use of via technologies that are currently being made at the densities and dimensions required by an ion-trap computer [110, 111]. A major optimization goal for system designers is to create scalable ion-trap geometries that minimize the needed electronics infrastructure in addition to the need to allow relatively easy access of the laser beams across the entire ion chip.

The laser systems outlined in Fig. 4.2 provide the different laser pulses needed for manipulating the ion qubits such as qubit preparation, logic gate operations, measurement, and cooling of ions after logic gates or movement. As shown in Fig. 4.1, sympathetic ions are used to absorb the accumulated heating from ion movement and gate operations. The sympathetic ions are cooled using cooling laser beams that are needed for both the sympathetic ions and the data ions [103]. Similarly, different laser beams with different wavelengths are needed for gate operations. These laser beams must precisely address individual ions for the reliable realization of both single- and two-qubit gates. In addition to the implementation of logic operations the computer must be capable of reading out qubit states quickly and reliably. Fault tolerant system designs that rely on error correcting codes require repeated measurements of individual qubit states throughout the application execution, which in turn require the implementation of measurement pump lasers that cause state-dependent scattering of photons from the ions. The scattered photons are detected by a CCD chip through the measurement optics region in Fig. 4.2. The precision and sheer size of the measurement optics region may force system designers to create ion-trap geometries that divide the ion chip into an ion interaction/storage region and a separate measurement region. Finally, the system-level parameters, control settings, and optimization techniques of the ion-trap computer infrastructure will depend heavily on the choice of ion species used for computation. Making a concrete choice for a general ion-trap computer is difficult at this stage of development since significant tradeoffs exist between different system requirements for different ion types [105]. The quantum computing roadmap, ARDA [52] contends that the choice of ion species best suitable for scalable quantum computation will be accepted by existing experimental groups by the year 2012.

CHAPTER 5

Robust Error Correction and Fault-Tolerant Structures

The existence of "good" error-correcting codes that allow the design of efficient fault-tolerant structures that overcome decoherence is, perhaps, the most critical requirement for a truly useful scalable machine. Due to the high volatility of quantum data, actively stabilizing the system's state through error correction (EC) will be one of the most resource-intensive operations through the course of a quantum algorithm. Unlike classical computation, which relies on the fact that failures are so rare that it is better to take longer for recovery than to spend extra resources for error correction [112], errors are frequent enough in quantum computing that recovery times are critical for the latency of the computation.

Quantum error correction and quantum fault-tolerance constitute a significant field of research [23, 113–119] that has produced some very powerful quantum error correcting codes analogous to, but fundamentally different from, their classical counterparts. The most important result, for our purposes, is the Threshold theorem [116], which says that an *arbitrarily reliable* quantum gate can be implemented using only *imperfect gates*, provided the imperfect gates have failure probability below a certain *threshold* value. This remarkable result is achieved through four main ideas: (1) using quantum error-correction codes; (2) performing all computations on encoded data; (3) using fault tolerant procedures; and (4) recursively encoding until the desired reliability is obtained. A successful architecture must be carefully designed to minimize the overhead of recursive error correction and be able to accommodate some of the most efficient error correcting codes.

The basic goal of quantum error correction is to purify an unknown n-qubit state $|\Psi\rangle$ from accumulated decoherence through some sequence of operations. The amount of decoherence can be abstracted as a random unitary, error operator that acts on $|\Psi\rangle$. Provided that the error operator acts nontrivially on $t < n$ qubits, error correction provides recovery procedures that can correct t errors on a register of n qubits by transferring the errors to a set of ancillary qubits to avoid direct measurement of the data qubits. After the transfer, the ancillary qubits are discarded or initialized to $|0\rangle$ for reuse. In the subsequent sections we give a brief overview of quantum error correction and the error correction codes we use in our architecture analysis by first describing the noise model we adopt throughout the rest of this book. The reader may look at [38] for a more detailed description of quantum error correction theory. We base our description on a class of quantum error correcting codes known as Calderbank–Shor–Steane codes [120, 121] that allow relatively straightforward quantum computation using the circuit model without the need to decode the encoded states.

5.1 NOISE MODEL

The most prevailing assumptions for noise on classical systems are a noise model called *White noise*: (1) the noise is *stochastic* where there is an equal probability ε of an error occurring in each position; and (2) errors are *uncorrelated*, and occur independently of each other. In practice, errors cannot be completely uncorrelated and may appear in bursts rather than independently, but the noise problem will then become an equipment design related problem and is thus not considered by error codes. Given the noise assumptions, if there are A locations where an error may occur in a classical circuit and an error occurs at each location with probability ε, the probability that t errors occur is given by

$$\binom{A}{t}\varepsilon^t(1-\varepsilon)^{A-t}, \tag{5.1}$$

which can be understood as the number of possible ways to have t locations that fail and $(A-t)$ locations that do not.

Classically, let a bit be in the state "0" with probability p_0 and the state "1" with probability p_1 initially. After the occurrence of a noise operator which flips the bit with the transition probability ε, the bit will be in the state 0 with probability q_0 and the state 1 with probability q_1. Then the evolution of the classical system for each independently occurring noise operation can be modeled as

$$\begin{bmatrix} q_0 \\ q_1 \end{bmatrix} = \begin{bmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}, \qquad \text{or} \qquad \vec{q} = E\vec{p}, \tag{5.2}$$

where E is the matrix of transition probabilities. In quantum computation, the evolution of a quantum system can be modeled in a similar manner. Suppose that we would like to apply the gate U on an n-qubit quantum register $|\Psi\rangle$. Even if the technology allows the state to be completely isolated from the environment, the physical mechanism used to implement the gate will most certainly introduce an error with some probability ε to our original state due to the fact that the possible unitary operators that can be applied on a state $|\Psi\rangle$ form a continuum. The final state after we apply the gate U can be written as

$$|\Psi\rangle \rightarrow \sum_a E_a U|\Psi\rangle \otimes |a\rangle, \tag{5.3}$$

where $|a\rangle$ are states of the environment (unentangled from the data register), and E_a are summed over 2^{2n} possible error operators that act on our state after the gate has been applied. Each error operator is a string of n Pauli matrices $\{I, X, Y, Z\}$ given in Eq. (2.15) and below:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, Y = -iZX = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

For example, after a three-qubit gate on a three-qubit register, the error on the three qubits may be any of the possible combinations of

$$\{I, X, Y, Z\}^{\otimes 3} = \{X \otimes I \otimes I\}, \{Z \otimes I \otimes Y\}, \ldots, \text{etc.},$$

where in each n-bit Pauli operator, the ith entry acts on the ith qubit. In the subsequent text we will omit the $\oplus$ signs within each n-qubit Pauli operator. The weight w of an n-qubit Pauli operator is defined as the number of elements which are not the identity matrix I. A general noise channel that can be used to estimate the effects of noise on a register of n-qubits that is also correctable by current quantum error correcting codes is known as the *depolarizing channel*, where at each location in a quantum circuit each of the n-qubits undergoes a transformation by one of the Pauli operators with probability ε and remains unaffected with probability $(1 - \varepsilon)$.

Most error correcting protocols rely on the fact that the weight of the n-bit Pauli operators is small, and that the occurrence of highly correlated errors that damage more than one qubit at each step is very rare. It is, however, very unlikely that the technologies will allow the complete elimination of uncorrelated errors. The Kane technology [66], for example, stores qubits in the electronic spins of phosphorous atoms embedded in silicon. Qubit interactions are controlled via metallic control structures built on the surface of the silicon substrate. To perform a two-qubit operation, the electron which stores the qubit from one atom is transferred to the other atom. Along the transfer process, the charge fields introduced by the control structures interact with the qubit states stored in the data electrons and in reality, pose the biggest difficulty for physically realizing reliable quantum operations using the Kane technology.

An error on a single qubit happens when a failure occurs during the execution of a gate on that qubit. A failure of the two-qubit CNOT gate can introduce two errors in a quantum circuit, one on the control qubit and one on the target qubit. Based on this assumption, the noise model we use to study the behavior of quantum architectures is as follows:

- Failures are uncorrelated and stochastic. This means that an error on qubit q_i will not result in error on qubit q_j unless the two qubits are explicitly entangled in the quantum circuit.

- An arbitrary error on a single qubit can be written as a superposition of the Pauli operators, where a failure at a single time step takes the density operator of our original system state $|\Psi\rangle$ to

$$|\Psi\rangle_{\text{new}} = (1-\varepsilon)I|\Psi\rangle + \frac{\varepsilon}{3}(X|\Psi\rangle + Z|\Psi\rangle + Y|\Psi\rangle).$$

  In other words, before the execution of a gate the qubit undergoes a rotation by X, Z, or Y with probability ε, and remains unchanged with probability $(1-\varepsilon)$.

- A two-qubit gate introduces two errors with probability ε on the input qubits equivalent to any of the 15 possible error patterns on two qubits: $\{IX, XI, IZ, ZI, IY, YI, XY, YX, XZ, ZX, ZY, YZ, XX, ZZ, YY\}$, each with probability $\varepsilon/15$. Single-qubit gates introduce an error with probability $\varepsilon/3$ any of the three possibilities between $\{X, Y, Z\}$. The T gate is the only exception, which introduces an error that can be written as a superposition of the X and Z gates.
- Memory failure rates and movement failure rates are equivalent to a gate failure rate per cycle. A particular technology model has a predefined distance that each qubit can travel in the duration of a single-gate cycle, with a specific failure rate ε that the MOVE gate will introduce an error. Similarly a memory cycle is equivalent to the qubit staying idle for a single-gate cycle, with a specific failure rate ε that the qubit will decohere.
- Steane [119] makes the important distinction that qubits participating in a gate at a given cycle undergo only gate noise and not memory noise. Similarly, qubits that move undergo movement noise introduced by the channel and not memory noise.

As mentioned earlier, errors cannot be completely uncorrelated. Initially, the state of a quantum computer is prepared such that it is independent of the environment system as much as the implementation technology will allow. As the computer state becomes entangled with the environment, the amount of entanglement governs how strongly correlated errors are between single qubits in the quantum computer. In addition, the application of a logic gate on a qubit also causes an unknown error operator to be applied on the state of the environment, thus the noise at each time step is shared between the computer system and the environment. If the entanglement between the two systems is small compared to uncorrelated gate failure rates ε, correlated errors do not asymptotically affect the scale of reliability achieved due to error correction. The qubit states in the ion-trap technology, for example, are affected by phase changes due to the fluctuating global electric and magnetic fields on the ion-trap chip. By fixing a single ion-qubit to be encoded using two physical ions such that

$$|0\rangle \rightarrow |01\rangle; \qquad |1\rangle \rightarrow |10\rangle; \qquad |+\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

known as a *decoherence free subspace* (DFS) [122], we can significantly reduce the correlation of our qubits with the environment by protecting them from a phase rotation on both qubits. Any phase error on both qubits will flip the sign of both the encoded $|0\rangle$ and $|1\rangle$ states, which would make the error global and it can be factored out.

An example of nonstochastic errors in the computer are small rotations in each of the qubits introduced by the classical control mechanisms at each gate. These rotations can be a constant change in the phase or a random rotation at each gate. If the state is randomly rotated by a very small angle θ, then the total angle of rotation after m operations will be approximately $\sqrt{m}\theta$ with probability $m\theta^2$ [119]. As system designers we must make the assumption that coherent nonstochastic errors will eventually add up to sufficiently larger rotations which can be discretized into a superposition of the Pauli operators as assumed in the incoherent noise model. A quantum computer would not be possible if nonstochastic contributions from the apparatus are such that the coherent errors at each time step are larger than correctable.

5.2 ERROR CORRECTION: BASIS AND NOTATION

The simplest way to deal with errors is to detect them without the need for correcting them. Errors are detected through *error-detecting codes*, which are used in classical computation in the transferring of information packets through noisy channels. Error detecting codes can be so computationally inexpensive that if the classical transmission channel introduces a sufficiently small number of errors, then it may be cheaper to retransmit the information packet upon the detection of error rather than calculating the exact error location.

In general, an error code C is defined by two parameters n and k, where n is the number of bits used to encode a piece of information and k is the minimum bits necessary to represent the information (C is denoted as an $[\![n, k]\!]$ code). A *single* error can always be detected in a n-bit binary bitstring by using an $[\![n, n-1]\!]$ *parity check* code that introduces only one bit of overhead. The parity check codes work by counting the number of times the bit 1 appears in the original n-bit binary message string. If the number of 1's is even, an extra bit of 0 is appended to the string, otherwise a 1. For example, the original message codeword 101 has an even number of 1's, so the check digit should be 0, changing the codeword to 1010, which is now a codeword in a $[\![4, 3]\!]$ code. A single error on any of the original bits will change the parity of the example codeword from even to odd. Thus, upon receiving the codeword we do a parity check by counting the number of 1's and determining whether the parity bit at the end matches the parity of the received codeword. If not, then the message is discarded and a duplicate can be sent. Note that the parity check code only allows us to detect the existence of an odd number of errors and provides no information about the actual location of any error that has occurred.

To *correct* errors requires the ability to encode the data in such a way that the location of the errors can be distinguished. Perhaps, the simplest error correcting code is the 3-bit

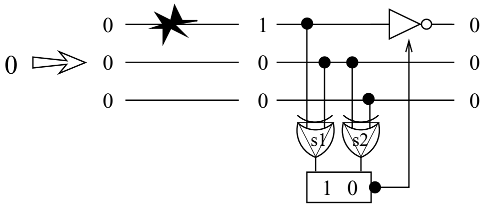


FIGURE 5.1: Correction procedure with the 3-bit classical repetition code. After the error occurs on the first bit, the syndrome bits ($s1, s2$) are set by measuring the parity between bits (1,2) and (2,3). The correction operation is just a NOT gate on the bit where the error has occurred.

repetition code with codewords "000" and "111." Each bit in the original information bitstring is redundantly encoded three times, where the bit "0" becomes "000" and "1" becomes "111." If one of the bits is flipped through some error occurrence, a majority vote is taken to determine the location of the error. For example, if the received codeword is "110" and the error probability ε is sufficiently low, we can safely assume that the string encodes the bit "1" and recover the original value, or we can correct the error by flipping the value of the third bit to "1."

The error correction procedure is illustrated in Fig. 5.1, where initially the bit "0" is repeated three times as 000 and sent through a noisy channel. An error on first bit flips its value to 1. The majority vote is taken by measuring the parity (i.e., applying the XOR gate) between the first and second bits, and the second and third bits whose result is stored in the syndrome string (s_1, s_2). In Fig. 5.1, the measured syndrome is $(1, 0)$ which tells us that the error is in the first bit. The syndromes $(1, 1)$ and $(0, 1)$ would tell us that the error is in the second and third bits respectively. Clearly, the 3-bit repetition code cannot help us if more than one error occurs. In fact, two errors will cause the error correction to correct the wrong bit, or simply return the opposite original data bit, thus introducing an error in our computation. A majority vote for a 5-bit repetition code (i.e., $0 \rightarrow 00000$ and $1 \rightarrow 11111$) will distinguish between any one and two-bit errors, but not three errors.

For a quantum error correcting code a little more is needed. Due to the no-cloning theorem, logical qubit states are highly entangled physical qubit states rather than a single physical qubit replicated a number of times. In addition, we need to worry about sign errors due to the phase-flip Z operator as well as bit-flip errors. The simplest code is the Shor code [23], which is similar to the classical repetition codes and can correct both types of errors uses nine physical qubits to encode a single qubit of information as three blocks of three qubits each:

$$\begin{aligned} |0\rangle \longrightarrow |\bar{0}\rangle &= \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \\ |1\rangle \longrightarrow |\bar{1}\rangle &= \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \end{aligned} \tag{5.4}$$

where the logical $|0\rangle$ and $|1\rangle$ states are written as $|\bar{0}\rangle$ and $|\bar{1}\rangle$, and an arbitrary *encoded* one-qubit superposition state can be written as

$$|\overline{\Psi}\rangle = \alpha|\bar{0}\rangle + \beta|\bar{1}\rangle. \tag{5.5}$$

Bit-flip errors can be detected and corrected by comparing the values of qubits within blocks, while by comparing the signs of the three blocks we can detect and correct phase-flip errors. Because all errors are a combination of X and Z errors, this code can correct an arbitrary single-qubit error on any of the nine qubits used in the encoding.

The circuit that encodes nine qubits to represent a single encoded qubit as a superposition of the $|\bar{0}\rangle$ and $|\bar{1}\rangle$ states is shown in Fig. 5.2, where the arbitrary single qubit $|Q\rangle = \alpha|0\rangle + \beta|1\rangle$ is encoded to $\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$. The data that we need to encode and protect is stored in qubit $q1$ as the arbitrary state $|Q\rangle$, which is entangled with eight additional qubits $\{q2-q9\}$ individually initialized to the $|0\rangle$ state. The first two CNOT gates distribute the state of qubit $q1$ into qubits $q4$ and $q7$ similar to the 3-bit repetition code: $|q1, q4, q7\rangle \longrightarrow \alpha|000\rangle + \beta|111\rangle$. The three Hadamard gates transform the three-qubit state into

$$|q1, q4, q7\rangle \longrightarrow \frac{\alpha}{\sqrt{8}}|+++\rangle + \frac{\beta}{\sqrt{8}}|---\rangle, \tag{5.6}$$

where $|+\rangle$ is the familiar $(|0\rangle + |1\rangle)/\sqrt{2}$ state. The state the three qubits are in after the Hadamard gates allows us to correct a phase-flip Z error on any of the three qubits if we compare the signs between qubits $(q1, q4)$ and $(q4, q7)$. To enable the correction of bit-flip errors, we encode the three qubits with the 3-bit repetition code using the other six qubits

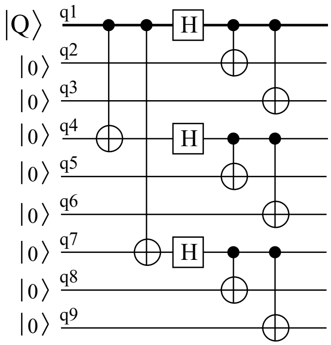


FIGURE 5.2: Encoding procedure for the 9-bit code. A three-qubit state is prepared initially that allows for the detection and correction of Z errors. Each of the three qubits is encoded with the quantum 3-bit repetition code to protect against bit-flip errors using 6 additional qubits.

$\{q2, q3, q5, q6, q8, q9\}$, where each $|+\rangle$ and $|-\rangle$ become

$$|+\rangle \longrightarrow \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$
$$|-\rangle \longrightarrow \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle).$$

The result is the encoded arbitrary single qubit state $(\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle)$ as given in Eq. 5.4. Extracting the syndrome for a bit-flip error in any of the three qubits within each group of three is identical to the classical 3-bit repetition code. Each of the three blocks of three qubits are in the state:

$$|q_1q_2q_3\rangle = |000\rangle + |111\rangle, \tag{5.7}$$

where the global phase factor of $\frac{1}{\sqrt{8}}$ has been omitted. The 2-bit syndrome string that would tell us which qubit was flipped can be obtained by performing the parity checks $(q_1 \oplus\ q_2)$ and $(q_2 \oplus\ q_3)$. For example, if a bit-flip error on qubit $q3$ occurs:

$$|q_1q_2q_3\rangle = |000\rangle + |111\rangle \rightarrow |001\rangle + |110\rangle, \tag{5.8}$$

the syndrome measurement should yield the syndrome bitstring (0, 1). The correction step is then performed by applying an X gate on the flipped qubit. Similarly, bit-flip errors are determined for the remaining two blocks of three qubits, $|q_4q_5q_6\rangle$ and $|q_7q_8q_9\rangle$.

The phase-flip Z errors are detected and corrected on any one of the nine qubits by comparing the signs of the three blocks. If a phase-flip error occurs, for example, on qubit $q6$, then the sign of the middle block will be flipped as shown below:

$$|\bar{0}\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)$$
$$|\bar{1}\rangle = \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle - |111\rangle). \tag{5.9}$$

Thus, the resulting syndrome string obtained by measuring the parity between the block 1 and block 2 and the parity between block 2 and block 3 should give us the syndrome (1, 1) indicating that there was a Z error in the middle block. Curiously, we can apply the correction on any one of the three qubits in the middle block $q4$, $q5$, or $q6$, and the sign will be flipped to the original state. The nine-qubit code is guaranteed to correct any one X or Z error in any of the nine-qubits in the state. It will not correct more than one Z error, but it may correct some higher weight X errors. For example, the error operator "$IIXIIXIIX$" of weight $w = 3$ where there is an X error on qubits $q3$, $q6$, and $q9$ causes all three errors to be in separate blocks, thus the nine-qubit code will be able to detect and correct them. On the other hand, the error operator "$XXIIIIIII$" will cause the first block to correct qubit $q3$, which will be wrong and the entire encoding will be taken out of the codespace, destroying the data that we are trying to protect.

The nine-qubit code is just one error correcting code and is perhaps the simplest truly quantum error correcting code that is capable of correcting both bit-flip and phase-flip errors. Many more quantum error correcting codes are known, where in general a quantum error code C encodes k qubits in n qubits and can correct errors on up to t qubits. Typically codes are identified by the three parameters $[\![n, k, d]\!]$, where d is the *code distance* such that $t = (d - 1)/2$. The nine-qubit Shor code can be thought of as a $[\![9, 1, 3]\!]$ code, whose distance d is equal to 3. A code that corrects any combination of two errors in its encoded codewords will have distance equal to 5.

It is not enough. However, to simply store quantum information, we must also have a way to reliably operates on it for the duration of the algorithm. If a qubit is encoded and protected with some $[\![n, k, d]\!]$ error-correcting code, decoding it for processing will prove fatal, for the gates in quantum computation introduce an error with probability ε each time a gate is applied. Classical circuits are extremely reliable, where after the application of each gate the process of dissipation is used to "cool" each bit by releasing some of the accumulated error into the environment. Clearly, we cannot couple a qubit to the environment after each unitary operator U, nor at any stage of the computation. Von Neumann [123] proposed that a classical computer with noisy gates can be made more reliable by performing each gate a number of times and accepting the majority of agreeing gates as the correct gate function. This would require to create multiple copies of the data to be sent through the same gate type, something that we cannot do in quantum computation. The solution is to perform operations on states that are already encoded. In addition, we need to do it *fault-tolerantly*, where more errors are not introduced than it is possible to correct. A nine-qubit encoded state that forms a single *logical qubit* guarantees protection of the encoded data from any one error which happens with probability ε. The data will be lost if more than one uncorrectable error occur, but if we never decode, higher errors occur with exponentially smaller probability (see Eq. 5.1).

In general, performing quantum computation on registers composed of n logical qubits $\{Q_1, Q_2, \ldots, Q_n\}$, where the qubits are encoded with clearly defined logical computational states $|\overline{0}\rangle$ and $|\overline{1}\rangle$, is functionally not different than computing with physical qubit registers. A logical gate $\overline{U}$ is constructed from a number of physical gates such that the function of $\overline{U}$ on an arbitrary logical qubit state is the same as the function of a corresponding physical gate U on functionally the same arbitrary physical qubit state. For example, applying the operator "$IIZIIZIIZ$" on an arbitrary nine-qubit logical qubit state encoded with the nine-qubit code will change the sign of each of the three blocks that make up the logical states $|\overline{0}\rangle$ and $|\overline{1}\rangle$, effectively flipping the value of the logical qubit from $|\overline{0}\rangle$ to $|\overline{1}\rangle$, or $|\overline{1}\rangle$ to $|\overline{0}\rangle$ Thus, with the nine-qubit encoding described in this section, the logical bit-flip operator $\overline{X}$ is implemented by applying a Z gate on qubits $q3$, $q6$, and $q9$. Similarly, the nine-qubit operator "$XXXXXXXXX$" (i.e., applying an X gate on all nine-qubits) is equivalent to applying a logical $\overline{Z}$ gate, taking the state

$\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$ to the state $\alpha|\bar{0}\rangle - \beta|\bar{1}\rangle$. Unfortunately, the implementation of other logical gates is not as straightforward with the nine-qubit code, thus it is important to consider the universal gate implementation circuitry when choosing an error correcting code for a given application. During computation each logical gate may be followed by a syndrome extraction procedure which would correct any errors (X, Z, or both) that have occurred during the sequence of operations that implement the gate.

Due to the no-cloning theorem, logical qubit states are highly entangled physical qubit states rather than a single physical qubit replicated a number of times. There are three major obstacles to overcome when performing error correction on encoded qubit states:

- Quantum states live in a continuous space identified by the probability amplitudes of the state vector, thus errors are continuous and in principle it should take an infinite number of resources to determine the exact error that has occurred. On single qubits we may see bit-flip X operator errors, along with phase-flip Z errors, or a combination of phase- and bit-flip errors such as the Y operator denoted as $-iZX$, or even a tiny, almost insignificant rotation of the qubit state.
- Measurement destroys the superposition of quantum data, but the only way to extract the error syndrome is by measuring an encoded qubit. Thus, we must indirectly measure the qubit such that its quantum information is not destroyed.
- Quantum data is fundamentally more faulty than classical data. Even if an implementation technology becomes extremely reliable, it may not be better than 1 error for every 10^8 operations [104] for ion traps, for example. In addition, quantum data is entangled. Thus quantum error correcting codes must prevent decoherence not only at higher than classical error rates, but against the exponential spread of errors introduced by entanglement. Section 5.4 details how quantum fault-tolerance achieved through concatenated quantum error correction can greatly reduce the error rate of quantum operations.

One of the most remarkable characteristics and breakthroughs in the theory of quantum error correction (QEC) is that errors can be *discretized* [23], thus solving the first obstacle for QEC. Unlike classical analog systems, any arbitrary error on one or more qubits may be corrected by correcting a small discrete set of errors: namely X, Z, and the combined X and Z errors. After an arbitrary error operator E_i on the ith qubit in the encoded logical qubit, the data state $|\Psi\rangle$ can be written as a superposition of the original state $|\Psi\rangle$, $X_i|\Psi\rangle$, $Z_i|\Psi\rangle$, and $Z_iX_i|\Psi\rangle$. No matter how small the error is, the error syndrome extraction procedure collapses the data state into one of the four elements of the superposition, which can then be corrected by applying either an X, Z, or both. The nice property in error discretization is that extracting the error within a logical qubit can be done simply by extracting a syndrome for X errors and then a syndrome for Z errors, each followed by the corresponding correction operation.

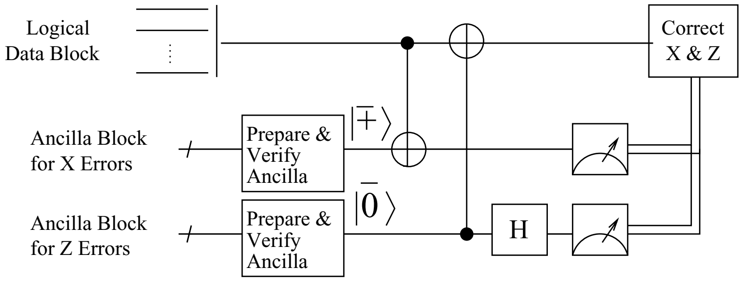


FIGURE 5.3: Extracting the syndrome using the Steane method. Two n-qubit ancilla blocks are prepared, where each line represents a logical block of physical qubits marked by a diagonal dash at the beginning of the line. The ancilla blocks are used to absorb information about the X and Z errors from the data, after which the data can be corrected. The first ancilla block is prepared in logical $|\overline{+}\rangle$ state which absorbs the X errors from the data. The second ancilla block is used to correct Z errors. Usually each of the two logical CNOT gates between the data and the ancilla blocks is a transversal CNOT gate composed of n physical CNOT gates applied in parallel.

The second obstacle is the inability to measure an encoded qubit directly to extract the error syndrome. Interestingly, this obstacle is not fatal either; however, it does introduce a large auxiliary qubit overhead. The error syndrome is transferred from the encoded qubit to a number of specially encoded *ancillary* qubits, which are then decoded and measured to reveal the location of the error. Commonly in the QEC codes we discuss here, interaction between the encoded data and the *ancilla* to extract the error syndrome for an $[\![n, k, d]\!]$ code is done in a method known as the *Steane Method* for syndrome extraction [124] which is shown in Fig. 5.3.

The Steane method for X and Z syndrome extraction is commonly used for Calderbank–Shor–Steane (CSS) quantum error correcting codes [120]. Two sets of n ancilla qubits are encoded using the same error code as the data. To measure X errors, the ancilla is prepared in the logical $|\overline{+}\rangle = \frac{1}{\sqrt{2}}(|\overline{0}\rangle + |\overline{1}\rangle)$ state and a logical CNOT gate is applied between the data block of n qubits as control and the ancilla block as target. A CNOT gate propagates bit-flip errors forward (i.e., control → target), thus the bit-flip X errors from the data block will be transferred to the ancilla. The errors and the location of the error can be extracted by measuring each of the ancilla qubits in the computational basis. To detect and correct phase-flip Z errors, the ancilla is prepared in the logical $|\overline{0}\rangle$ state and the ancilla is used as the control block during the interaction with the data (Z errors propagate backward in a CNOT gate). Applying a logical Hadamard gate on the ancilla forces the Z errors into bit-flip errors, which can be detected upon measurement in the computational basis.

Measurement of an encoded block of qubits works much the same way as measuring a physical qubit, where the state is collapsed to either the logical $|\overline{0}\rangle$, or $|\overline{1}\rangle$ basis states. If the encoded block is intended for a code that corrects up to t errors, measuring a state with any errors of weight $w \leq t$ present will yield the correct measurement unless some of the measurement gates fail themselves.

The Ancilla factory concept. In Fig. 5.3 we show two n-qubit ancillary blocks, one for the X errors syndrome and one for the Z errors syndrome. What is not shown, is the fact that the ancilla blocks, once prepared must be verified against the presence of X and Z errors themselves to ensure that errors created when preparing the ancilla (1) do not propagate to the data causing errors of higher than correctable weight, or (2) do not cause an incorrect syndrome to be measured. Once verified the ancilla blocks can be used for interaction with the data to extract the error syndrome of the data.

The {Ancilla factory} refers to the idea that ancilla blocks must be constantly prepared and verified throughout the error correction procedure. Ancilla blocks can be verified by preparing additional ancilla blocks much like the error correction process and measuring those. More optimal verification structure can be explored by studying circuit synthesis rules and the ways errors propagate through gates such that we need to verify only against errors that could have propagated to the end of the ancilla preparation networks. Since error correction needs to be done frequently in quantum computation, the ancilla preparation process will be critical for the latency of the computation. It is possible to use only a single ancilla block for both X and Z errors and perform the syndrome extraction sequentially by re-preparing the ancilla for each error type, which would increase the error correction time, but reduce auxiliary resource usage. Alternately, we can prepare many ancilla at once that guarantee that, when error correction is needed, there will always be a prepared and verified ancilla block ready for syndrome extraction for both X and Z errors.

5.3 EXAMPLE: THE STEANE $[\![7, 1, 3]\!]$ CODE

For the case studies in the large-scale architecture model presented in this publication we use the Steane $[\![7, 1, 3]\!]$ code [113], which encodes a single logical qubit in 7 physical qubits and can correct up to any single-qubit error. It is based on the classical $[\![7, 4]\!]$ Hamming code, which allows the correction of any single-bit error where the error location is given by the syndrome string that represents the binary numbers between zero and seven. The syndrome string of "000" denotes no error and the string "010" denotes an error on the second qubit.

The $[\![7, 1, 3]\!]$ quantum code is a member of the family of some of the most powerful error correcting codes known today as Calderbank–Shor–Steane (CSS) codes which allow *transversal* logical CNOT gate operations and whose error correction procedure requires only CNOT and Hadamard gates as shown in Fig. 5.3. A logical operator $\overline{U}$ is transversal if its implementation is achieved by applying $\overline{U}$ in parallel to all n encoded physical qubits in a logical qubit block. Furthermore, the $[\![7, 1, 3]\!]$ code is the smallest CSS code that allows transversal implementation of quantum operations which are members of the *Clifford Group*. The clifford group is composed of

$$\{H,\ \mathtt{CNOT},\ X,\ Z,\ Y = -i\,ZX,\ S\}, \tag{5.10}$$

where the S gate is the familiar phase rotation along the $\hat{z}$-axis of a qubit with a phase angle equal to $\phi = \pi/2$ as defined in Eq. 2.14. The T gate (the other phase rotation gate defined in Eq. 2.14) is all that is necessary to complete the logically-universal gate set for quantum information processing; however, the logical construction of this gate is considerably more complicated when encoding our data with the [[7, 1, 3]] code. The $|\bar{0}\rangle$ codeword for the Steane [[7, 1, 3]] code is given by the seven-qubit state:

$$\begin{aligned}|\bar{0}\rangle = {} & |0000000\rangle + |1111000\rangle + |1100110\rangle + |1010101\rangle \\ & + |0011110\rangle + |0101101\rangle + |0110011\rangle + |1001011\rangle,\end{aligned}$$

where the $|\bar{1}\rangle$ state is obtained by applying the logical $\bar{X}$ operator, which is simply seven one-qubit X operators on each of the 7 qubits in the Steane state. It is straightforward to verify that the action of any of the Clifford group gates transversally on an arbitrary logical qubit state $|\bar{\Psi}\rangle = \alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$ for the [[7, 1, 3]] code is equivalent to the action of the corresponding physical gate on an arbitrary single-qubit state. In addition, the measurement operation is also transversal. Measuring each of the seven qubits and calculating the parity of the resulting bitstring, will identify correctly if we have measured the logical $|\bar{0}\rangle$ or $|\bar{1}\rangle$ state.

Fig. 5.4 shows the circuit used to correct a logical data bit for X errors with the [[7, 1, 3]] code. For the correcting procedure we use the Steane method, where the steps are:

- First we prepare a block of ancilla in the encoded $|\bar{0}\rangle$ state as described in [124] and shown in the expanded encoding gate of Fig. 5.4. Traditionally the preparation network involves just nine CNOT gates; however, this would require additional block of seven ancilla qubits for the verification, which is applied after the encoding. The circuit shown uses only one ancilla verification bit, and the verification is part of the encoding procedure.
- Second, a transversal Hadamard gate is applied which places the ancilla in the $|\bar{+}\rangle$ state. The ancilla is then interacted with the data block using a transversal CNOT gate, where the data block is the control qubit and the ancilla block is the target qubit.
- Measuring the ancilla block allows us to extract the error syndrome. If the syndrome is nontrivial (e.g., shows an error) we repeat the process again until we get two identical syndromes with a maximum of three repetitions.
- Finally we apply the corrective X gate on the corrupted data bit.

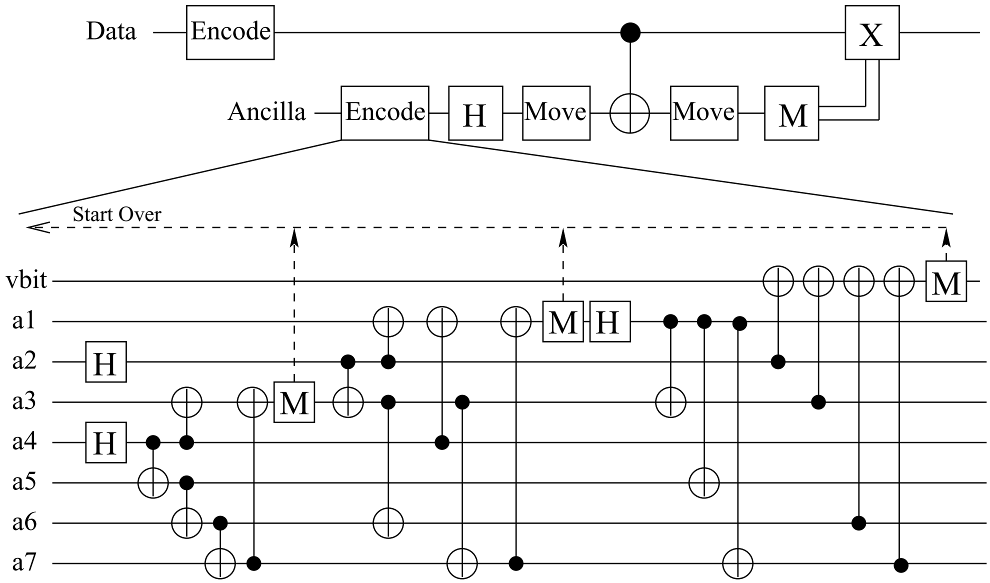


FIGURE 5.4: Circuit for extracting X-error syndrome and correcting X errors for the Steane $[[7,1,3]]$ code using only one verification qubit when the ancilla is prepared. After the preparation network completes (lower part of the figure), the ancilla is in the logical $|\overline{0}\rangle$ state and is placed in the needed $|\overline{+}\rangle$ state for X-error correction by the logical Hadamard gate which follows the preparation procedure (top circuit). Either the data or the ancilla is then "moved" for the implementation of the logical CNOT gate which transfers the X-error information from the data to the ancilla. The measurement operation denoted by the letter M measures each of the physical ancilla qubits in parallel and the syndrome is extracted by multiplying the measurement string by the parity-check matrix for the 7-bit Hamming code.

The same syndrome extraction is repeated for correcting Z errors on the logical data qubit, with the only difference being the flipped control-target blocks for the transversal CNOT gate and the placing of the Hadamard gate after the transversal CNOT (see Fig. 5.3). The repetition of the syndrome extraction before the corrective operation is necessary with this encoding procedure because the encoder does not verify the ancilla for Z errors. Subsequently, by applying the transversal Hadamard gate, all Z errors are converted to X errors, causing us to measure the wrong error syndrome. By repeating the syndrome extraction we ensure that the probability of measuring the wrong syndrome due to Z errors in the encoder is a second-order event.

Generally, it is preferable to choose error correcting codes that allow the implementation of as much transversal logical gates as possible. The fact that the Steane $[[7,1,3]]$ code is a CSS code guarantees a transversal CNOT gate between two logical data qubits and the transversal implementation of the clifford group gates only makes this code more desirable. Codes exist

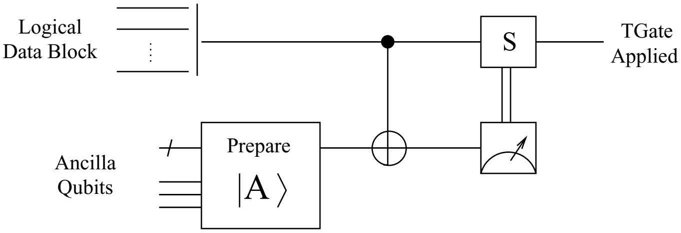


FIGURE 5.5: T-gate implementation with the ⟦7, 1, 3⟧ code.

that allow transversal implementation of the T gate; however, the other gates are not transversal, and clifford group gates are by far the most dominant set of gates executed during the course of a large-scale quantum application.

The T gate implementation with the ⟦7, 1, 3⟧ code requires an additional ancilla block specially encoded such that the concept of *1-bit teleportation* can be used [125]. Any arbitrary single-qubit unitary operator U can be implemented using one-bit teleportation. Particularly for the implementation of the T gate the one-bit teleportation method is shown in Fig. 5.5. A seven-qubit $A_{\pi/8}$ ancilla state is prepared using additional ancillary qubits and interacted with the logical data block to which we want to apply the gate. Because phase information propagates backward in CNOT gates, the action of the T has been applied on the logical data block with some error. A measurement of the $A_{\pi/8}$ qubit will tell us if we should correct the error by applying the S gate on the data block.

5.4 QUANTUM FAULT TOLERANCE: THE THRESHOLD RESULT

The theory of QEC is powerful and much deeper than we can possibly present here; however, for it to be truly useful for scalable, computationally relevant quantum information processing, there needs to be a way to overcome the exponential spread of errors in an entangled quantum system during the execution of an algorithm. This is especially important because not only are the gates from the application faulty, but so are the gates involved during error correction. The formulation of fault-tolerant quantum circuits and the threshold result [116, 126] has made all discussions for scalable, reliable quantum computation possible. The threshold result states that an arbitrarily long quantum computation can be executed with arbitrary reliability using faulty physical gates, provided that the failure rate of each gate is below a certain *accuracy threshold* value. Strict requirements for the existence of the threshold value are

- The noise on the quantum hardware occurs independently at each location in a quantum circuit. A location in a quantum circuit is defined as any operation on a qubit such as a gate, or even an idle cycle while the qubit waits for a gate on another qubit to complete.

Idle cycles and movement operations on qubits can be abstracted as a WAIT gate and a MOVE gate, respectively.

- Each location in a quantum circuit must introduce an error on the qubit with probability ε, and must work perfectly with probability $(1 - \varepsilon)$. In other words, the noise is stochastic, where the failure probability ε depends entirely on the operation type.
- If n qubits are encoded to form a single logical qubit, the logical circuit structures for gates and error-correction routines such as encoding networks and syndrome extraction networks, must be fault-tolerant. A fault-tolerant circuit is a circuit where a single error on any lower-level physical qubit with probability ε will not spread to $(t + 1)$ or more errors elsewhere in the circuit. The assumption is that we have an error-correcting code capable of correcting at most t errors.

In a logical circuit each line implies an encoded set of physical qubits using a certain $[\![n, k, d]\!]$ code with a sequence of logical gates. Given that the lower level circuit structures and hardware noise satisfy the fault-tolerant requirements, a fault-tolerant logical gate is followed by an error-correction step on the logical qubit block. The abstraction for a fault-tolerant CNOT gate is shown in Fig. 5.6. The physical CNOT gate is shown to the right, where the control and target qubits are both physical qubits. At the encoded logical level, both the control and target are logical qubit structures of n qubits in an $[\![n, k, d]\!]$ code for the data and the additional ancilla needed for error correction. For the $[\![7, 1, 3]\!]$ code the logical CNOT gate is transversal and is composed of seven physical CNOT gates applied in parallel. The error-correction step that follows every logical gate overlaps with the error correction step that precedes the next logical gate on the same set of qubits. In essence, each gate in a logical circuit is followed by an error-correction step. The central assumption is that the number of errors that slip through the logical gate construction network will be corrected by the following error-correction procedure, provided

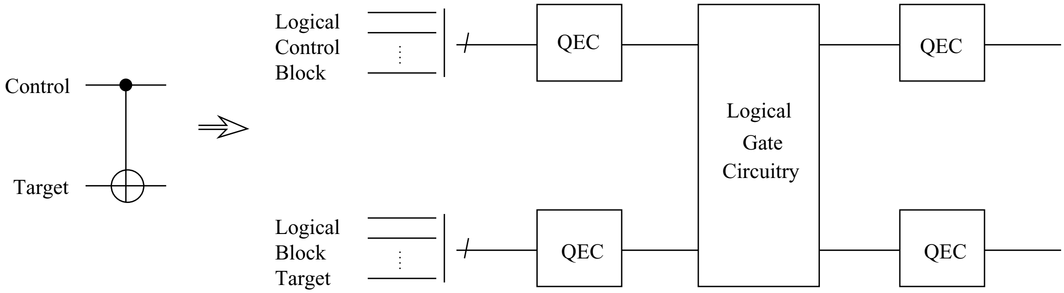


FIGURE 5.6: Physical → logical gate construction, where a fault-tolerant logical gate is preceded and followed by error correction.

that the gate construction and the error-correction procedures are constructed fault-tolerantly where the probability of errors of weight greater than t is a second-order event.

The failure rate of each logical operation for level one encoded data, which is preceded and followed by error correction (as shown in Fig. 5.6) can be bounded as

$$\varepsilon_1 \leq A\varepsilon^{(t+1)}, \tag{5.11}$$

where A is the number of locations in the logical gate circuit shown on the right-hand-side of Fig. 5.6 that cause greater than $(t+1)$ errors to appear at the output of the circuit. The "1" subscript on ε denotes a *single level* of encoding, while ε without a subscript denotes the failure rate of a physical gate, which is at level 0 encoding. If a logical qubit is encoded in a block of n qubits, it is possible to encode each of those n qubits again with an m-qubit code to produce an mn encoding. Such recursion, or *concatenation*, of codes can reduce further the logical operations failure rates, provided that the physical failure rates are below the threshold value.

Concatenated error correction introduces an exponential cost with each increasing level of recursion. If each logical qubit block, or each logical line in Fig. 5.6, is implemented with an $[\![n, k, d]\!]$ code concatenated L times, then each line consists of at least n^L physical qubits. Fig. 5.7 shows the structure of a logical qubit at level L encoding, where level 1 encoding is defined as the encoding of n *physical* qubits. Encoding once more for a cost of n^2 physical qubits we have a logical qubit at level 2.

Logical circuits composed of logical gates, which themselves are composed of self-similar lower level logical gates must obey the same rules of fault tolerance as the rules for the physical

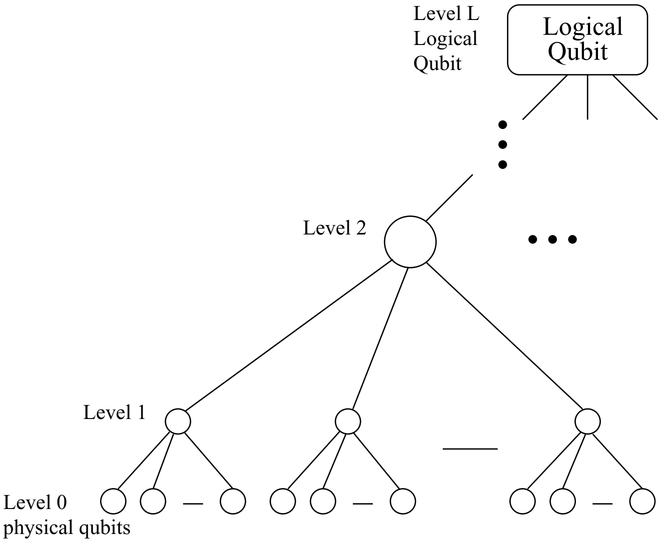


FIGURE 5.7: The tree structure for a logical qubit using concatenated error-correcting codes.

circuit outlined above. An upper bound for the failure rate of a level L logical gate can be defined as:

$$\varepsilon_L \leq A(\varepsilon_{L-1})^{(t+1)} = \frac{1}{A}(A\varepsilon)^{(t+1)^L}. \tag{5.12}$$

Notice that the "$\leq$" sign will not hold for ε_L if the physical component failure rate ε is greater than A^{-1}. Therefore the accuracy threshold value ε_{th} for an $[\![n, k, d]\!]$ error-correcting code is given as $1/A$, where A is directly affected by the error-correcting code used. For a given error-correcting code, if the physical component failure rate is below $\varepsilon_{\text{th}} = 1/A$ we can increase the level of recursion until we reach a desired reliability of computation, or even a reliability that is good enough to sustain computation until the application completes.

As a system designer, calculating the threshold value for a chosen error-correction code will help determine the amount of reliability obtainable with the code at different levels of recursion. The most commonly cited threshold value is $\varepsilon_{\text{th}} = 10^{-4}$ for the Steane $[\![7, 1, 3]\!]$ code [38]. The existence of this value, however, assumes perfectly noiseless and instantaneous qubit communication along with fast and reliable measurement operations. Gottesman [118] showed that a threshold value exists in a local setting where qubit communication is considered. In his work he allows qubits to interact with their nearest neighbors only where movement is performed through successive swapping of the qubit states. The threshold for a local architecture based on Gottesman's specifications was subsequently computed to be on the order of 10^{-5} [127]. The Steane method for syndrome extraction has allowed a significant simplification in the error-correction networks and thus much higher threshold values have been recently calculated when neither movement nor WAIT gates are considered [119, 128].

In general, any assumption made about the model of a quantum circuit and its threshold value is accounted for by the total number of fault locations A. Any existing threshold calculation has made simplifying assumptions to make the task of calculating the number of fault-locations tractable. Quantum architecture designers' main concern, however, is not the exact threshold value, but the design of a fault-tolerant system such that computation can be sustained throughout the application with the minimal number of resources. A qubit at level L may be encoded using one $[\![n, k, d]\!]$ code, while its lower level qubits may use another. The best way to predict the value of the threshold and the system behavior is through repeated simulations of each component if exact values of the fault locations A are not available.

5.5 CONSTRUCTION OF A LOGICAL QUBIT TILE

From a computer architect's perspective, stabilizing an n-qubit quantum state is equivalent to recursively building logical qubit tiles (or blocks) such that the error rate per logical operation followed by error correction falls below some desired value that will allow us to sustain the needed computation. Each logical qubit tile at level L recursion must be crafted in such a way that the failure rate per tile scales as $O(\varepsilon^{t+1})$, where ε is the failure rate of each level $(L-1)$ tile used to encode the higher level qubit. In other words, the physical design and construction of each level L logical qubit must be *fault-tolerant*.

The efficiency of the design of a fault-tolerant logical qubit tile can depend on several design choices that are not orthogonal: (1) the first and most important design choice is the $[\![n, k, d]\!]$ error-correcting code that serves best the functionality of the qubit tile in relation to the entire processor design. (2) Once a satisfactory code is chosen, the lower level qubits that are encoded to form each higher level qubit must be arranged in a fault-tolerant manner such that the communication pattern over the error correction network with those qubits is minimized. The more efficient the physical arrangement of the qubits is, the less the negative impact of erratic qubit movement will be on the accuracy threshold value. In addition, the more efficient the physical structure of the networks is, the lower the chances that the preparation of the encoded ancilla used in error correction will fail. (3) Another very important design choice is the allocation of physical qubit resources for error correction. A number of ancilla blocks may be allocated for a single-error correction procedure such that they are prepared in parallel and we are guaranteed that at least one ancilla block will have passed verification for the extraction of the syndrome. Alternately, we may allocate qubits for only one ancilla block and wait with the syndrome extraction until the ancilla has been prepared and passed verification.

A hypothetical schematic of a recursively constructed logical qubit tile is shown in Fig. 5.8 without any specific low-level constructions. The figure as a whole shows a logical qubit at

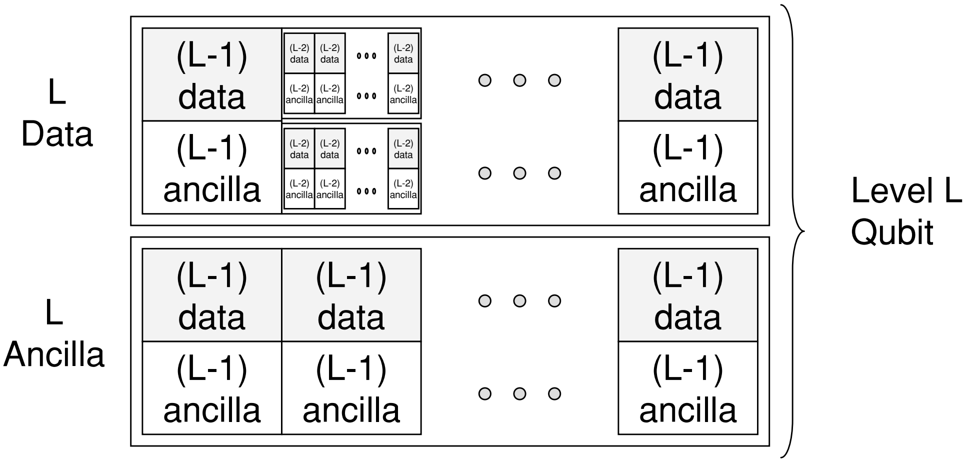


FIGURE 5.8: Tile-based logical qubit structure.

level L, which is composed of a level L ancilla block and a level L data block. The ancilla block is needed for the Steane syndrome extraction method. Each qubit block at level L is constructed using $2n$ level $(L-1)$ blocks, which in turn are constructed of level $(L-2)$ blocks as shown in the figure. One requirement for the existence of a threshold value and thus, the ability to reduce the reliability with each higher level construction, is that a level L data block be near the level L ancilla block used for syndrome extraction [118]. In addition, Fig. 5.8 does not show any additional ancilla blocks that are needed for extracting X and Z syndromes in parallel, nor for verification of the ancilla blocks used in the syndrome extraction process.

5.6 COST OF QUANTUM ERROR CORRECTION

While quantum computation promises computation that may be exponentially powerful in the number of qubits, coping with decoherence introduces a time and space overhead that is also exponential in the number of qubits and the running time of an algorithm. In this section, we examine this "contest of two exponentials" and outline how to design systems that win this contest and retain the computational advantages of quantum systems.

Concatenated error correction introduces an exponential cost as the level of concatenation increases at the physical resources, number of operations, and time per logical operation in a single application. The physical resource increase may prove to be the most costly parameter as we recurse, since we must provide ancillary qubits for each logical qubit to perform quantum error correction at each operation. In addition a data qubit at level L encoding must also support several ancillary qubits at level L encoding if the Steane QEC method is used, or $O(n^L)$-qubit cat-states at level $(L-1)$ per line, which must be error corrected and verified. The increase in computational resources, however, comes with super-exponential decrease in the probability of failure per logical operation. As shown in Eq. 5.12, the reliability gain from concatenated error correction increases as $(t+1)^L$ rather just L at the exponent. The probability of failure ε_L per logical operation decreases doubly-exponentially with L for distance three quantum codes such as the Steane $[[7, 1, 3]]$ code. Therefore, reaching a desired level of reliability for a given application may only require a few levels of recursion preserving the exponential improvement over the application's execution on conventional computers.

5.6.1 System Size

The system size S for a given application can be defined as the product of Q logical qubits and K time steps [119]. The duration of a time step is taken to be the time it takes to perform the logical operation, which includes error correcting the n lower level qubits that are encoded in the logical qubit, followed by the time to error correct each logical qubit. The failure rate necessary to achieve a system size $S = KQ$ per logical operation is $\varepsilon_{\text{desired}} = 1/KQ = 1/S$. A quantum computer with sufficient computational resources may take as many resources as necessary for each application to encode data at the desired level of encoding for a carefully chosen error-correcting code to reach the desired system size. For applications with small KQ parameter this would leave many qubit resources unused. Another method would be to assume a fixed error-correcting code and level of recursion with the hope that too large applications will be unrealistic to achieve. Allowing for a very large system size KQ at all times, however, would be like driving an all-time four-wheel-drive automobile. The system reliability is not always necessary and only causes computation at higher than needed levels of recursion.

Clearly, a lower failure rate could be achieved faster with $[\![n, k, d]\!]$ error-correcting codes with $t > 1$ as opposed to the Steane $[\![7, 1, 3]\!]$ code we describe in Section 5.3. Such codes, however, use a much higher number of encoded lower level qubits for each logical qubit and the number of locations A that may produce a fault have not been clearly identified, especially when qubit communication is considered within the error-correction procedure. In addition, careful studies [119] exist for larger error-correcting codes that suggest much more efficient logical circuit structures in terms of resources and latency when $k > 1$. Codes that encode n qubits in $k > 1$ qubits are known as *block codes* and n is usually quite large. The usefulness of these codes, however, for large-scale quantum architecture is still unclear, as the error-correction procedures themselves are very complicated.

To evaluate the expected logical gate failure rate at some level of recursion L for quantum codes where $k = t = 1$, one can use Gottesman's estimate for local architectures [118] shown below

$$\varepsilon_L = \frac{1}{Ar^2r^L}(Ar^2\varepsilon)^{2^L} = \frac{\varepsilon_{th}}{r^L}(\varepsilon_{th}^{-1}\varepsilon)^{2^L}, \tag{5.13}$$

where the value for r is the communication distance within level 1 encoded blocks defined as the average number of MOVE operations per physical qubit. Equation (5.13) is a rather pessimistic estimate that assumes that the distance the qubits travel before they are being corrected increases exponentially with the recursion level L. While this is true, for sufficiently long distances the concept of teleportation may be used to change the movement model and allow for lower failure rate ε_L estimates.

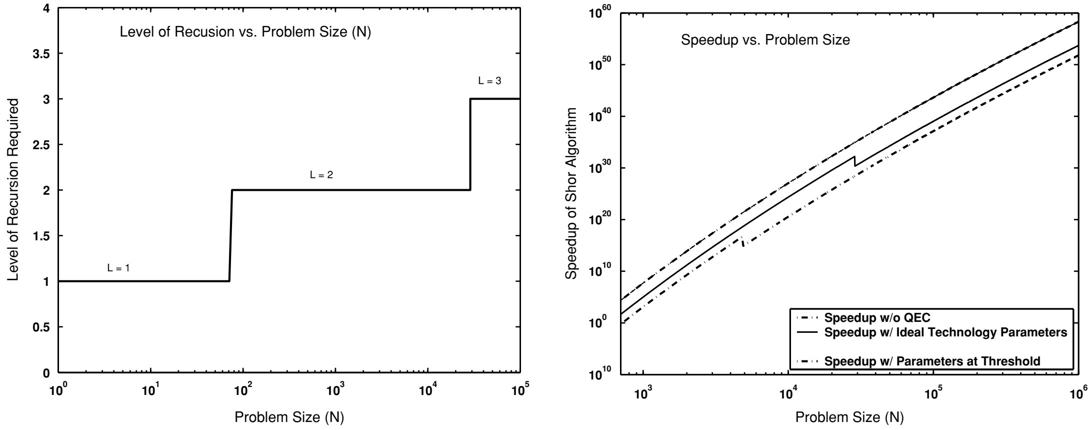
5.6.2 Error-Correction Slowdown

From a first look, it seems that the exponential slowdown due to error correction even with qubit tiles of only a few levels of recursion is prohibitive when the system size S becomes very large. For some applications, however, the exponential slowdown from error correction is balanced by the exponential speedup offered by the quantum algorithm structure versus its classical counterpart. One such application is Shor's quantum factoring algorithm, which is designed to break the widely used RSA public-key cryptosystem. RSA's security lies at the assumption that factoring large integers is very hard, and as the RSA system and cryptography in general have attracted much attention, so has the factoring problem. The efforts of many researchers have made factoring easier for numbers of any size, irrespective of the speed of the hardware. However, factoring is still a very difficult problem. The best classical algorithm known today [9] has complexity of

$$\exp\left((1.923 + o(1))(\log N)^{1/3}(\log\log N)^{2/3}\right)$$

for an N-bit integer. As a basis of comparison we use the most recent success at factoring a 663-bit number [129] classically for an estimated 121,000 MIPS years ($\approx 4 \times 10^{18}$ instructions). This is equivalent to a little over one year on a 100 GHz PC with a perfectly parallelized and distributed factoring implementation.

A plot of the required level of recursion versus the problem size N for factoring an N-bit integer using Shor's algorithm is shown in Fig. 5.9(a). The system parameters used are the Steane [[7, 1, 3]] code with ion-trap technology assumptions that are optimistic, but within the fundamental limits of the technology and not out of reach in the future. The details of the architecture are described in Chapter 9. We see that for factoring a 1024-bit (or even a



(a) (b)

FIGURE 5.9: (a) Required level of recursion for Shor's algorithm as a function of the problem size N defined in the context of an N-bit number that is being factored. (b) Speedup of Shor's algorithm as a function of the problem size N. The top-most line shows the speedup without error correction, the middle line shows the speedup with error correction, but at error parameters approximately three orders of magnitude below the accuracy threshold for the Steane [[7, 1, 3]] code, the bottom line the error parameters are at the threshold value of the [[7, 1, 3]] code. Each "glitch" in the two lower lines is an increase in the level of recursion.

2048-bit) number, level 2 recursion with the Steane $[\![7, 1, 3]\!]$ code may be sufficient given the provided architecture design. The optimistic error rates for the ion-trap technology are almost three orders of magnitude below the existing estimate for the accuracy threshold value of approximately 10^{-5} for the Steane $[\![7, 1, 3]\!]$ code [130].

The slowdown due to error correction can be seen in the logarithmic scale plot shown in Fig. 5.9(b), where the $\hat{y}$-axis marks the speedup of the quantum algorithm from its classical counterpart. The speedup is calculated as the number of days classically divided by the number of days quantum mechanically. The top line is the speedup without error correction. The middle line is the speedup with the optimistic ion-trap parameters, while the bottom line is the speedup with technology error rates at the threshold value of approximately 10^{-5}. Each "blip" on the speedup lines with error correction corresponds to increasing the level of recursion by one unit. The smallest problem size shown is $N = 700$, which requires level 2 encoding. The same problem size requires level 3 encoding if the technology parameters are at the threshold value. As we can see, even with error correction, the exponential speedup is preserved over classical computation. The asymptotic cost of Shor's algorithm is polynomial, and the polynomial cost incurred by the computation is responsible for the deviation of the speedup lines from being truly exponential (note the slight curvature). A physical operation in an ion-trap quantum computer is on the order of 10 μs thus at the physical level, the speedup calculated is based on a kHz quantum computer.

CHAPTER 6

Quantum Resource Distribution

Fundamentally, technologies that are well-suited for quantum computation are not ideal for quantum communication. This tension arises from the need to interact qubits with each other and the application of control signals on the qubits during computation, versus the need to insulate qubits from any interactions during information exchange from one location to another. For the execution of an arbitrary single-qubit operation the physical qubit carrier is usually exposed to an external field such as an ion shined on by a laser light. Similarly, a two-qubit operation such as the CNOT gate requires a specially focused external field to induce a coupling between the two qubits. On the other hand, the transport of a qubit requires the movement of the physical qubit carriers in such a way that they are isolated from external environment fields in order to preserve the qubit states during transport. Trapped atomic ions are well-suited carriers for computation since the qubit state lifetime is relatively long in trapped ions and laser light can be adjusted to induce quantum operations with a failure rate of as little as 10^{-7} [104]. The transport of ions requires the control of the trapping potentials and the movement of the ions through external electric and magnetic fields which contribute to increased failure rate in the quantum operations and the possibility of losing the qubit states during transport. Photons traveling through optical fibers are ideal for reliable communication since they do not interact well with the environment and with each other. The weak photon–photon interactions, however, make photon qubit carriers not desirable for extensive computation unless clever entangling methods are employed such as quantum gate implementations through collective photon measurements [93].

Consequently, communication is a significant challenge in scalable quantum computers. At the lowest level each qubit is a carrier of quantum information which cannot be cloned, and must be physically transported from a source to a destination. This makes each qubit either a physical transmitter of quantum information, where the qubit itself is physically moved, or operations are applied to transmit the information across a given distance. Both methods place great constraints on the reliability and speed of quantum data distribution. One method to protect the data from corruption is to repeatedly error correct along the channel at a cost of additional error correction resources. Another solution is to use the purely quantum concept of teleportation [26] to implement a long-range wire [54], which has been experimentally demonstrated on a very small scale [131, 78, 77]. As described in Section 2.4, teleportation transmits a quantum state between two points without actually sending any quantum data, but rather two bits of classical information for each qubit on both ends. In addition, the coupling of remote atomic qubits which are well suited for computation can be achieved through photon interactions [58, 95, 97, 98]. The design and optimization of a quantum architecture to support efficient data communication scalably to arbitrary large applications will be one of the key areas of contribution for computer architects.

6.1 PHYSICAL QUBIT MOVEMENT

Using the circuit model of computation with sufficient error correction, a CNOT gate between two qubits will be the most dominant operation that requires qubit–qubit interaction [130]. There is a large variety of physical qubit communication mechanisms employed by the available technologies to allow two qubits to interact. In fact, the classification of the qubit types heavily depends on the communication mechanisms available for interacting two or more qubits.

Qubits identified as *flying* qubits such as photons are constantly in motion, and gates are stationary physical devices that affect the photon qubits as they fly through the gate. Traditionally, photons are sent through fiber optic wires and the main source of decoherence in the wires is photon absorbtion. Photon qubits, however, are difficult to use in a quantum circuit model implementation for relevant computation, as it is very difficult to transfer the state of a "flying" qubit to a *stationary* qubit for computation.

Stationary qubits such as the solid-state qubit proposals occupy a specific physical space (or a fixed qubit container), where qubit-qubit interactions are limited to nearest neighbor only [56, 71, 66]. The construction of arbitrary one- and two-dimensional lattices for logical qubits using "stationary" qubits is perfectly possible through successfully *swapping* two neighboring qubit states until two specific qubit states reside in neighboring qubit containers. The nearest-neighbor communication channels are limited by the reliability of the SWAP operation, which is implemented by applying three successive CNOT gates. The physical gate mechanism for "stationary" qubits is an external system applied at the location of the qubit.

Trapped atomic ions that hold the qubit states offer a cross between "flying" and "stationary" qubits where the ions can be trapped between the segmented electrodes. Lasers can be applied to perform a logic gate at any previously defined interaction region. Two ions interact by ballistically shuttling the ions across the physical layout such that they occupy the same trap. An interesting proposal for Josephson junction qubits supports long-distance gates, where any two qubits are allowed to interact without the need to move them; however, the proposal limits the circuit execution to only one gate at a time [132, 79] on a single chip.

Nearest-neighbor and ballistic qubit communication mechanisms are best suited for implementation of the circuit model for quantum computation as they offer the most straightforward implementation of reconfigurable quantum logic [133]. From a system designer's perspective, the two communication models can be indistinguishable: the cost of successive SWAP operations across a swapping channel can be compared to physically moving ions through a sequence of unit distances in an empty ballistic channel. The challenge for system designers will be to map quantum circuits to physical layouts such that the latency of communication has minimal effect on the latency of the circuit execution. In addition, the physical layout designer must consider that an increased number of SWAP operations, or MOVE operations through a unit length channel is equivalent to performing random faulty operations on the transported qubit. Thus great care must be taken to create schedules that optimize not only for latency constraints, but reliability constraints.

The error correction procedures for qubits encoded at relatively low levels of concatenation may require an enormous amount of physical qubit movement, however, through clever optimization techniques it can be possible to limit the movement errors on the data. A significant problem arises when qubits encoded at a relatively high level of recursion must communicate with one another (for example, the execution of a transversal two-qubit gate between two logical qubits at level 3 concatenation). The exponential increase in the separation between the physical qubits at each additional level of recursion introduces distances that are impossible to traverse physically without a prohibiting loss of data. In the next section we describe the concept of quantum teleportation as the means for reliable long-distance communication in quantum architectures.

6.2 TELEPORTATION-BASED INTERCONNECT: QUANTUM REPEATERS

The concept of using teleportation as a long-distance communication channel is illustrated in Fig. 6.1 in three stages. The first stage involves the entangling of two qubits into an EPR pair through the network shown in Fig. 2.9. The two qubits are then transported through a physical channel where one is moved next to the source qubit and one to the location where we would like to transport the source qubit. Once the source qubit is interacted with the EPR qubit, the two are measured and the source can be recreated at the destination.

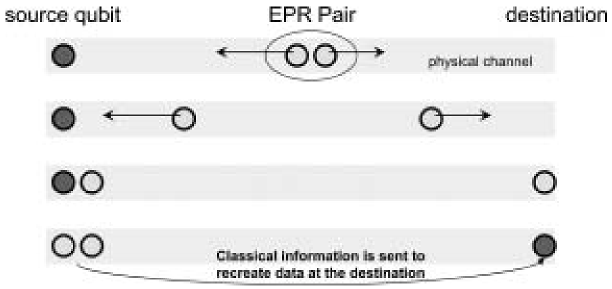

source qubit
EPR Pair
destination
physical channel
Classical information is sent to recreate data at the destination

FIGURE 6.1: Illustration of the stages of teleportation.

Note that we are still physically moving the entangled EPR qubits; however, unlike the source qubit, EPR qubits are replaceable. The damaged EPR pairs can be fixed by a process called *entanglement purification* [134, 18], which uses ancillary EPR pairs to distill the good pairs from the bad pairs. The caveat to purification is that the amount of resources increases exponentially with the EPR separation distance, along with the fact that if the EPR pair becomes too corrupted it may not even be purifiable. As the physical distance the EPR pairs must travel approaches the coherence length allowed by the implementation technology: (1) the number of additional EPR pairs required for purification of a single EPR pair increases exponentially; and (2) the fidelity of each of the qubits sent through the channel decreases exponentially. For large-scale quantum architectures we will need to send qubits at distances much larger than the coherence length of the physical channels [27, 28], and it would seem that an enormous amount of resources would be needed to achieve these distances.

Fortunately, entanglement is preserved through teleportation. For example, if one qubit is entangled with another qubit in the system, after it is teleported, the two qubits are still entangled in the same way. Thus, a very large channel may be divided into a number of smaller channels that are within the allowable physical coherence length and EPR pairs can be created and purified only within each segment of the channel. Through *entanglement swapping* we can

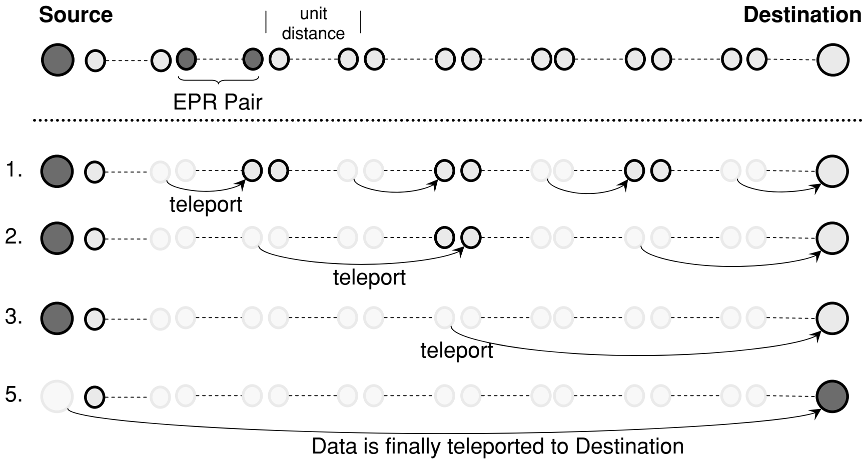


FIGURE 6.2: Illustration of Entanglement Swapping. A long-distance channel between the source qubit and the destination qubit is divided into a number of smaller segments connected with an EPR pair. EPR pairs only travel to two nearby islands, where they can be efficiently purified using the purification protocols with some additional ancillary EPR pairs. In stages 1 through 3 we teleport in parallel across the stations to reduce the number of connecting EPR pairs by half at each step, but still keep the connection between the source and the destination. Finally, we teleport the source qubit to its desired location when a single EPR pair spans the connection channel.

transfer the entanglement of the EPR pairs to create a single entangled pair that spans the two ends of the channel [135].

Fig. 6.2 demonstrates the stages of the entanglement swapping protocol. A long-distance channel between the source qubit and the destination qubit is divided into a number of smaller segments by *quantum repeater stations*, which are connected with a single EPR pair. The quantum repeaters can be implemented as islands that are strategically placed in the channels between the logical qubits to limit the distance traveled by each EPR pair. EPR pairs only travel to two nearby islands, where they can be efficiently purified using the purification protocols with some additional ancillary EPR pairs. To expand a single entangled EPR pair between the source and the destination over the entire channel we use a logarithmic algorithm similar to computing the transitive closure. In Fig. 6.2 there are four stages after the EPR pairs have been created to connect each neighboring repeater station. In stages 1 through 3 we teleport in parallel across the stations to reduce the number of connecting EPR pairs by half at each step, but still keep the connection between the source and the destination. Finally, we teleport the source qubit to its desired location when a single EPR pair spans the connection channel.

As we shall see further in Section 11, teleportation is a remarkable concept and can be used for much more than simply connecting two relatively distant locations on a chip. In fact, through teleportation it is even possible to avoid direct qubit-qubit interaction when executing two-qubit gates. Another remarkable property of quantum teleportaiton is the ability to error correct logical qubits as entire qubit blocks are being teleported. In general, the existence of the elegant long-distance quantum repeater protocol opens up many possibilities for the use of teleportation in large-scale quantum architectures.

CHAPTER 7

Simulation of Quantum Computation

As the technology for implementing QIP continues to advance one of the central challenges for system designers now and in the future will be the ability to accurately simulate the behavior of large-scale quantum computers. The main challange stems from the fact that quantum information processing can be described as an *extension* of the classical computational model when information is represented as a superposition of quantum bitstring states rather than as single classical bitstrings. This perspective helps us understand why the classical computational model is a subset of the larger quantum information processing scheme, and thus a classical system cannot efficiently simulate a quantum system.

In addition, as the general structure of large-scale quantum computers emerges clearer with each technological advancement, the need to accurately model such systems will increase in both urgency and importance.

A unitary operation on an n-qubit quantum register requires $O(2^n)$ operations to simulate and an $O(2^n)$ data bitstring entries to store the state of the register. Because of the limits imposed by destructive measurement, researchers are not convinced that quantum computation is necessarily more powerful than the classical model, and it is unclear where the boundary between the two models is. One fundamental boundary value is the accuracy threshold for fault-tolerant circuits. The state of an entangled quantum system, such as a logical qubit, can be sustained for an arbitrarily long period of time if the physical component failure rates are below the accuracy threshold of the encoding used. If the component failure rates are above the threshold value, then the entanglement will decohere exponentially quickly and the system will be forced to a single classical state [116].

Even worse for the simulation of quantum computers is the fact that no quantum computer system is completely isolated from its environment. In fact, to allow the implementation of a desired set of operations to be applied by some external system, quantum computers are inherently *open* to noise introduced by the environment. As our system evolves through time, it becomes entangled with the surrounding environment, and unknown forces that act on the environment cause decoherence directly to our system. This means that to accurately track the evolution of a quantum system, which is coupled, to the environment, we must store more information then necessary as opposed to tracking the superposition state of an isolated quantum register [38].

As system designers, however, we may not need to track the exact computations performed by quantum applications, but rather the application behavior in the architecture, such as latency, fault-tolerance, and effect on overall system size. As we shall see in this chapter, the challenges of efficiently simulating a general-purpose quantum computer with conventional techniques are far from prohibiting when we attempt to efficiently model the behavior of large-scale quantum applications. After all, if it were possible to simulate a general-purpose quantum computer efficiently, then there would be no need to build one. Luckily it is not possible.

Several general-purpose quantum simulators exist in the literature, including the QCE simulator specifically designed to simulate quantum computer hardware at the lowest level possible [136], the high-level language for quantum computation (also known as QCL) [137] is a functional-level general-purpose simulator with no knowledge of the hardware, and the quantum decision diagrams (QuiDD) package by Viamontes et al. [138] allows us to simulate arbitrary circuits. All general-purpose simulators incur exponential cost with each additional qubit, and thus simulating even several hundred qubits is completely unrealistic. Other simulators that impose limits on the entanglement of the system can simulate quantum circuits in polynomial time as long as the functionality of the circuits satisfies the imposed constraints [139, 140]. A restriction on entanglement is prohibitive for a systems designer who attempts to model error correction, which requires highly entangled qubits for a single logical codeword. There are two types of simulation methods that allow us to model the behavior of quantum circuits using methods that are polynomial in time, but do not impose any limits on the entanglement produced by the simulated circuit: simulation of error propagation and using the unique *stabilizer* representation of an n-qubit register. Both methods, however, require that the circuits are composed of only the Clifford group gates. This is, in fact, more than enough to simulate quantum error correction which is the bulk of the computational resources [53] during an application execution.

In Section 7.2 we describe in better detail the stabilizer formalism for quantum circuits, where we use it to simulate *stabilizer circuits* directly. Any n-qubit state $|\Psi\rangle$ that can be formed entirely with the Clifford group gates

$$\{H,\ \texttt{CNOT},\ X,\ Z,\ Y = -iZX,\ S\}, \tag{7.1}$$

where the qubits must start in the initial state $|0_1 0_2\ \ldots\ 0_n\rangle$, is known as a *stabilizer state*. The stabilizer circuit is the circuit composed of the Clifford group gates that form $|\Psi\rangle$. Any stabilizer state $|\Psi\rangle$ can be described uniquely using only $O(n^2)$ unitary one-qubit Pauli operators $\{I, X, Y, Z\}$. It is a powerful representation for quantum states first published by Gottesman in

1996 [115], where Gottesman provides a description for a very powerful class of error-correcting codes known as *stabilizer codes*. The class of CSS codes such as the Steane $[\![7, 1, 3]\!]$ code is a subset of the class of stabilizer codes.

7.1 SIMULATION OF ERROR PROPAGATION

If, as a system designer, one is not concerned with the precise state of the system at a given point in time, but rather is concerned with the failure rate of the system as a whole, or even each fault-tolerant component, one can use error propagation to accurately simulate the behavior of any active state stabilization mechanism in a logical qubit tile. In addition, inter-tile communication based on teleportation is also implemented using a stabilizer circuit (see Fig. 2.3). Thus, one can simulate the reliability and efficiency of the logical interconnect efficiently on a classical computer using error propagation. The key to simulation of error propagation is that an error on a qubit at any location of a circuit changes the state of the qubit, which causes any control gates based on that qubit to behave differently. Thus the error propagates through two-qubit gates and spreads to other qubits as the program progresses. This is why it is absolutely necessary to implement error-correcting circuits fault-tolerantly in such a way that an error on any 1 to t qubits will not spread to more than t qubits if the network is a recovery network, or a logical gate network for an $[\![n, k, d]\!]$ code correcting $t = (d - 1)/2$ errors.

Consider the simple circuit examples shown in Figs. 7.1 and 7.2, which demonstrate the propagation of X and Z errors, respectively. Both networks are carbon copies of the encoding network for the Steane $[\![7, 1, 3]\!]$ code shown in Fig. 5.4 but both start at the first measurement operation. Because the measurements measure in the computational basis, they will detect the states $|0\rangle$ or $|1\rangle$, and the X error will be detected by either measurement gate. Phase-flip errors on the other hand slip through the network and have the potential to multiply to more than one error as shown in Fig. 7.2. Should more than one Z error really does slip during

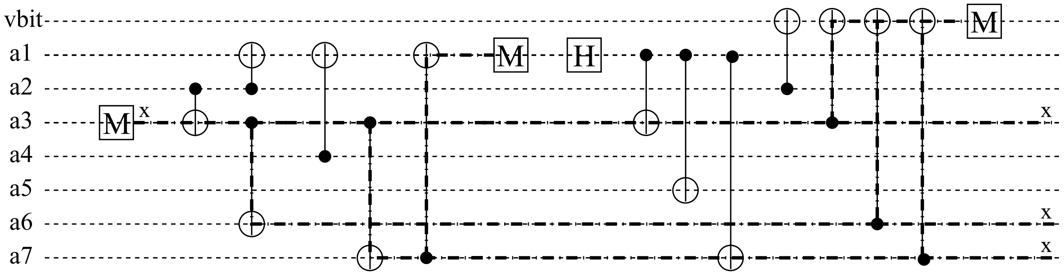


FIGURE 7.1: X-error propagation. The qubit lines affected by the error are shown in a thicker dot-dashed line. The measurement operations in the middle and the end of the network are designed to yield "1" if error is present and "0" otherwise.

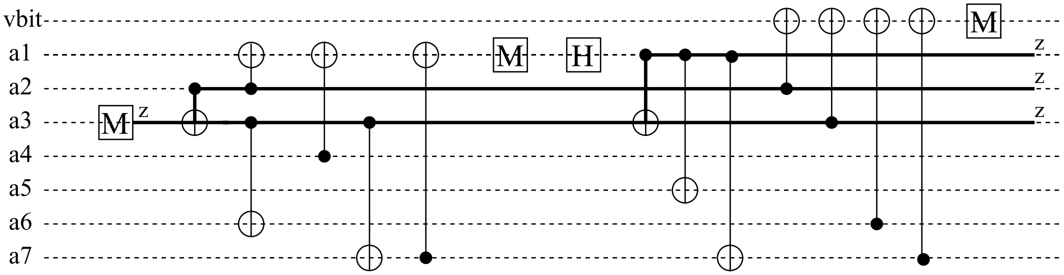


FIGURE 7.2: *Z*-error propagation. The qubit lines affected by the error are shown in a thicker solid line. Note that the Z errors are undetected by this network, and a single Z error occurring in the middle of the circuit has caused three Z errors in the output.

recovery procedure, the syndrome extraction will yield the wrong error location and thus we run the potential of correcting the wrong data bit (see Section 5.3). This is the reason why the syndrome extraction is repeated if a nontrivial syndrome is found, and we correct only upon matching consecutive syndrome measurements.

In reality, the effect of errors due to *all* gates can be traced through error-propagation simulations. Adding the T to the mix of gates whose errors we would like to track would complete the universal set for computation. The problem is that errors introduced by the T gate are a probabilistic superposition of the X and Z gates; thus we must follow both error paths. With each T gate in a quantum circuit, the number of paths doubles, and quickly the circuit becomes intractable to simulate. Applications such as Shor's algorithm rely heavily on the Toffoli gate described in Section 2.2.1, which is composed almost entirely of T gates.

Note that X and Z errors propagate differently through the two-qubit CNOT gates. Bit-flip errors propagate "forward" (i.e., control → target), while phase-flip errors propagate "backward" through a CNOT gate. This is easy to see for bit-flip errors since the state of the target bit is flipped depending on the state of the control bit. For phase flips we can see this more easily if we consider the states of both the input and the target qubit to be $(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$. After a Z error on the target qubit, the target qubit state will be $(|0\rangle - |1\rangle)$. The application of the CNOT gate puts the system in the state $(|00\rangle - |01\rangle + |11\rangle - |10\rangle)$, which can be written as $(|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle)$. Thus, we can see that the Z error has now been propagated to the control qubit as well as the target qubit after the CNOT gate.

The extensive quantum architecture tools known as QUALE by Balensiefer et al. [141, 142] use the error to verify the fault-tolerant properties of the error-correction networks they have modeled. QUALE uses traditional compiler techniques to map quantum circuits onto a realistic physical layout in order to enable the study of large-scale quantum applications and hardware. The intent of the software tool chain of QUALE is to simplify the development of large-scale quantum applications, where error correction—the most dominant application—is verified through simulating the propagation of errors. Since the noise model is stochastic and errors occur with associated probabilities, Monte Carlo simulation can be used to find the failure probability of any stabilizer circuit such as a logical operation as defined in Fig. 5.6. After the network is executed a sufficient number of times, the essential failure of the entire circuit is the number of registered failures divided by the number of registered successes per trial. A registered failure is any time more errors have propagated at the output of the circuit than the error correction code can correct. If a large-scale quantum computation is composed of a sequence of logical gates such as the gate in Fig. 5.6, the application is marked as "failed" and is restarted whenever one of two things happen: (1) more errors enter the recovery network than is possible to correct, which would completely change the meaning of the encoded codeword; and (2) either the recovery network or the logical gate circuitry are not fault tolerant, and cause a single fault at any location to propagate to more errors than the next recovery network can correct.

The drawback of simulating propagation of errors however, is that the failure probability results are pessimistic when compared to statistical data obtained from other simulation methods. Without knowing the state of the quantum register it is impossible to determine how and what type of fault will actually be a real fault. It is true that any of the Pauli operators are applied with equal probability ε, but the phase-flip operator Z, for example, does not affect the $|0\rangle$ state. Thus, simulating propagation of errors sometimes introduces faults on qubit states that are unaffected by the error operator, which makes it effectively a nonerror. A logical qubit in the encoded $|\overline{+}\rangle$ state is unaffected by a logical $\overline{X}$ operator; however, if enough X errors have propagated to that logical qubit such that they implement the $\overline{X}$ operator, this will be registered as a logical error and crash the entire application.

7.2 STABILIZER METHOD SIMULATION

Another method for efficiently simulating stabilizer networks is through the stabilizer formalism [115, 126]. Recall that any arbitrary n-qubit state $|\Psi\rangle$ which can be formed with gates in the Clifford group, provided that all qubits in the register have been initialized to $|0\rangle$, is a stabilizer state. An n-qubit operator U stabilizes the state $|\Psi\rangle$ if U does not change the state: $U|\Psi\rangle = |\Psi\rangle$. The key to the stabilizer formalism's use for the simulation of quantum circuits is the *Gottesman–Knill* theorem, which states that if the n-qubit state $|\Psi\rangle$ is a stabilizer state, then:

- $|\Psi\rangle$ is stabilized by a set of n-qubit operators composed of the Pauli group matrices given in Eq. 2.15.
- The stabilizer group can be generated by an $O(n)$ number of n-qubit Pauli operators (i.e., every stabilizer operator of the state $|\Psi\rangle$ can be written as a product of a small set of stabilizer operators for $|\Psi\rangle$).

- The state $|\Psi\rangle$ is uniquely described by the set of operators that generate all of its stabilizers. These operators are known as the *stabilizer generators* for $|\Psi\rangle$.

The last point states that it is exponentially cheaper to describe a stabilizer state $|\Psi\rangle$ using its stabilizer generators, rather than describing the state explicitly. Consider, for example, the stabilizer generators for some unknown three-qubit state $\{III, ZZI, IZZ, ZIZ\}$, where the ith Pauli operator in a stabilizer string is understood to act on the ith qubit only. The first operator III stabilizes anything, because it is just the identity on all three qubits. The second operator ZZI stabilizes the four states $|000\rangle$, $|001\rangle$, $|110\rangle$, and $|111\rangle$. The third operator IZZ stabilizes the states $|000\rangle$, $|100\rangle$, $|011\rangle$, and $|111\rangle$. Finally, the last operator stabilizes the states $|000\rangle$, $|101\rangle$, $|010\rangle$, and $|111\rangle$. Note that common to all four operators are the two states $|000\rangle$ and $|111\rangle$; thus the state stabilized by the generators $\{III, ZZI, IZZ, ZIZ\}$ is $|\Psi\rangle = \frac{1}{\sqrt{2}}(|000\rangle \pm |111\rangle)$. Usually, the stabilizer generator will include a sign.

The total number of classical bits needed to specify an n-qubit stabilizer state $|\Psi\rangle$ is $(2n + 1)$, where the "1" is due to the sign bit, and there are $2n$ Pauli operators to write down. Additionally, Gottesman and Knill showed that unitary operations on the qubits that are part of the Clifford group such as the CNOT, Hadamard, S-gate, and measurement take each stabilizer state to a different stabilizer state; thus the action of these gates can be modeled in only $O(n)$ time. Measurement is slightly more expensive if the outcome is deterministic, where the stabilizer generators can be updated in $O(n^3)$ time. Aaronson and Gottesman later demonstrated an implementation of a stabilizer-based simulator (known as CHP), where measurement can be updated in $O(n^2)$ time [143].

While we cannot simulate Shor's algorithm exclusively with stabilizer circuits, we can simulate efficiently the largest and most efficient class of error-correcting codes known: stabilizer, CSS codes such as the Steane [[7, 1, 3]] code, in addition to some of the most important quantum protocols such as teleportation, and superdense coding. In fact, the stabilizer formalism can be used to directly derive encoding and error-correcting procedures for stabilizer codes. For example, the set of n-qubit Pauli operators that generate the stabilizers for the encoded logical states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ for the Steane [[7, 1, 3]] is given by the six operators $\{g1, g2, g3, g4, g5, g6\}$ where

$$\begin{aligned}\{g1, g2, g3, g4, g5, g6\} = \{&XXXXIII,\ XXIIXXI,\ XIXIXIX,\\ &ZZZZIII,\ ZZIIZZI,\ ZIZIZIZ\}. \end{aligned} \tag{7.2}$$

The reader can verify that applying any of the above operators to the encoded $|\bar{0}\rangle$ and $|\bar{1}\rangle$ states for the [[7, 1, 3]] code (given in Eq. 5.11), will not change the two codewords. On the other hand, a Pauli error on any of the seven qubits will change the stabilizers for the two codewords. Thus, measuring each of the stabilizer generators to determine which ones still stabilize the codeword states is another way of obtaining an error syndrome for the ⟦7, 1, 3⟧ code. Two Pauli errors on any two qubits in the seven-qubit encoded states will transform the stabilizers to seven-qubit Pauli operators that are generated by the product of some of the six generators defined above, which will leave the stabilizer generators untouched. For this reason, two-qubit errors are undetectable with the Steane ⟦7, 1, 3⟧ code.

Stabilizer networks can be verified for fault tolerance and functionality using Monte Carlo simulations much the same way as simulating error propagation in networks. The interoperable tool chain QASM-TOOLS developed by Cross et al. [144] uses the assembly language QASM as an input language to represent and study fault-tolerant quantum circuits by estimating depolarizing noise thresholds using Monte Carlo simulation, and functionally verify stabilizer circuits using Aaronson's improved stabilizer simulator CHP. In addition, QASM-TOOLS can find lower bounds for the accuracy threshold of distance three codes such as the Steane ⟦7, 1, 3⟧ code using general malignant set counting [145], which counts all combinations of locations in a logical gate circuit that cause the network to fail.

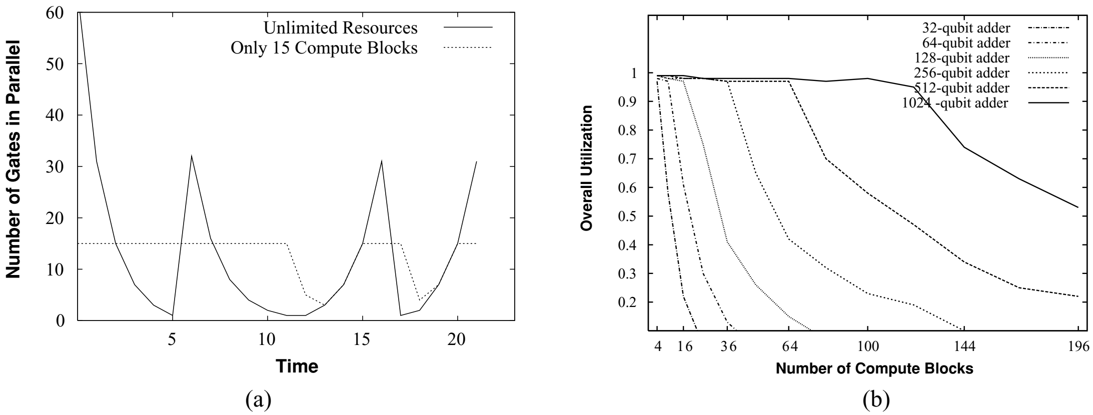
CHAPTER 8
Architectural Elements

Given our discussion at this stage of the book, we may deduce that a natural model for a large-scale quantum architecture is a homogeneous, tiled architecture with two main component categories:

1. Logical qubits implemented as self-contained computational tiles allowing gates to be performed directly on the encoded data while containing the necessary error-correction resources to correct data immediately following a logical gate execution.
2. Teleportation-based communication channels that may employ the concept of quantum repeaters to allow information transmission across arbitrary regions in the architecture.

One major drawback of such a homogeneous architecture is that by definition it allows the application of gates on encoded data blocks at any tile containing a logical qubit. In addition, active state stabilization (i.e. error correction) is required for each logical qubit after each logical gate (see Figure 5.6). This requirement makes independent lower-level qubit resources for adequate error correction an integral part of every logical qubit block, which in turn leads to forbidding area requirements when constructing a computationally relevant quantum chip. This is especially true when a spatially sparse technology is used as the trapped atomic ions, which could bring the computer area to as much as one square meter when factoring a 1024-bit number [27] and the underlying substrate is a piece of silicon dye. On the other hand, the homogeneous "sea-of-qubits" design for a quantum architecture makes sense, as every single logical qubit tile requires error correction, which is a process no different than executing a quantum circuit. Thus, memory and computation in quantum hardware use the same technology, and allowing logical operations at every tile while each tile is capable of active state stabilization is not limited by the physical implementation of the hardware.

Quantum applications, however, much like classical ones, exhibit natural serialization. By exploiting the limited parallelism at both the application and the physical microarchitecture level of a quantum computer, it is possible to reduce the area requirement while improving performance by limiting the wasted error-correction qubit resources [28]. In particular, a scalable quantum architecture design may employ specialization of the system into memory and



(a) (b)

FIGURE 8.1: Total communication and computation times for the two components of Shor's algorithm, (a) for a 64-qubit adder, the amount of parallelism that can be extracted when resources are unlimited, and when the number of gates per cycle are limited. This figure shows that if 15 gates, or an unlimited number of gates could be performed in each cycle, the total run time would remain the same. (b) Change in utilization as the number of compute blocks increases.

computational regions, each individually optimized to match hardware support to the available parallelism. A system designer for a quantum architecture may gain density increase by specializing components as blocks of memory and blocks of computation. As shown in the case study for a quantum architecture in Chapter 9, the area improvement over a homogeneous architecture can be as large as nine times. Shor's factoring algorithm, for example, is dominated by modular exponentiation, which is composed of adders. Fig. 8.1(a) plots the number of gates executed in parallel versus the run time of a 64-bit quantum adder routine. We see that the total run time remains the same when the logical qubit tiles that allow computation are limited to 15 instead of unlimited number at any given execution cycle. This is explained by the simple fact that the utilization of the available compute blocks drops as the number of compute blocks increases. The utilization of the compute blocks as a function of the number of compute blocks is plotted in the $\hat{y}$-axis of Fig. 8.1(b). Clearly, a more sophisticated scheduler may not need all 64 logical qubit tiles to allow gate execution, but distributing the execution cycles among lower number of tiles will allow us not only to reduce the error-correction resources in the tiles where computation is not allowed, but it may help with the classical resource distribution.

The high-level specialized architecture model we describe is shown in Fig. 8.2. The model is constructed from a collection of specialized architectural elements much like a classical architecture, independent of the physical implementation technology. Each element is composed of a number of "tiles" where each tile represents one or more logical qubits composed of a number of physical qubits encoded using a prespecified error-correcting code. The shaded tiles are taken to be logical data qubits and the clear tiles are logical ancilla blocks used for error

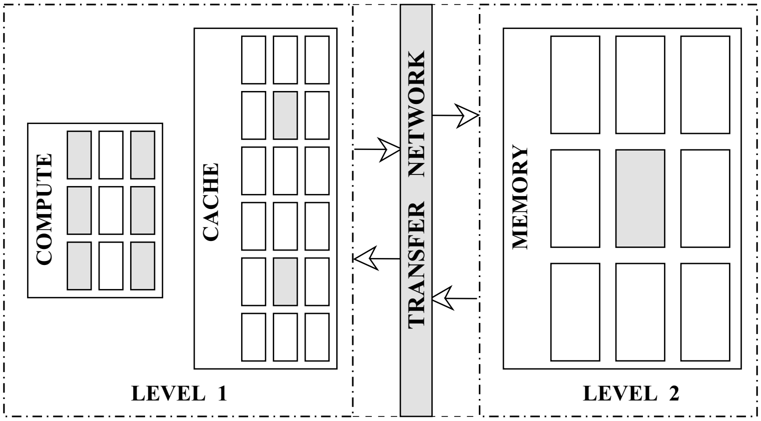


FIGURE 8.2: High-level general processor view.

correction at the highest level of recursion. Assuming that the implementation technology allows a WAIT gate to be considerably more reliable than other gates (a notion not completely unrealistic), we can reduce the overall computer area by changing the ratio of logical data blocks to logical ancilla blocks between memory and computational regions. For example, the memory tile shown on the right-hand-side of Fig. 8.2 is primarily concerned with storing the actively stabilizing state of encoded data qubits. The intervals between error-correction operations are increased by increasing the number of data qubits for each ancillary qubit that can be used for error correction. This cannot be done in the computational tiles, because error correction is needed after the execution of each gate, and a single computational tile may be used for the execution of both one- and two-qubit gates. In addition, we may be able to combine area savings with improved performance, by defining a specialized *compute code* (CC) used in the processing elements, and *memory code* (MC) used for storing data. The only logical operation employed by the memory code would be a WAIT gate, which is simply doing nothing.

The introduction of different encodings between tiles that allow computation and tiles that only store qubits will require a complex transfer network between the different encodings, where the data must not be decoded in the transfer process. As we will see further in this chapter, the transfer network is slow for it is composed of a number of gates on the encoded data and measurement operations, each followed by error correction. Fig. 8.2 shows an additional *cache* region used to buffer encoded data with the computational code after it is transferred from memory. In some ways, the memory hierarchy we describe in this chapter is a *code hierarchy*, where the hierarchical structure is needed to overcome the latency differences between state stabilization and code transfer from one encoding to another. The structure and optimization of the hierarchy is perhaps the most complex component of the architecture as it provides the transition operations necessary to take data encoded in the highest level of the hierarchy to the encoding needed for computation without delaying the algorithm execution.

8.1 QUANTUM PROCESSING ELEMENTS (PEs)

All logical quantum operations take place in the processing element (PE) tiles. A schematic of a hypothetical PE tile is shown in Fig. 8.3. When a logical qubit is teleported to an available PE it is stored in either of the two *accumulators* encoded with the compute code (CC), where the CC is chosen to be fast and relatively inexpensive in the number of physical qubits needed for encoding and error correcting a single logical qubit. The error correction is performed before and after the application of a single logical gate on the data stored in any of the two accumulators using the closer of the two ancillary blocks. The logical qubit needed for a single-qubit operation is loaded into one of the two accumulators of an available PE. The qubit can be found waiting in the *quantum cache* encoded with the same CC, or is teleported directly from the main memory if there is an available accumulator in some PE unit. A two-qubit gate requires both accumulators, where the physical qubits of each of the two participating logical qubits must interact with one another. There are enough CC ancilla provided to correct both logical qubits in each of the accumulators. The lines between the different regions in each PE are not as clear in reality as drawn in Fig. 8.3. For example, in the ion-trap technology the execution of a two-qubit gate with the Steane $[\![7, 1, 3]\!]$ code will require 49 pairs of ions to be placed in the same trap. Thus, both accumulators can be constructed by having 49 traps that allow physical two-qubit gates to be executed.

Gates acting on logical qubits must be implemented to preserve fault tolerance, where a single error on any of the lower level logical qubits will *not* spread to more lower level qubits than the CC can correct. The gates act on logical qubits without decoding the states; thus a compiler optimizing the fault-tolerant structure of each gate must have clearly defined transformation rules that preserve fault tolerance. The best CCs are the ones that (1) use very little physical qubit resource overhead and (2) allow "easy" fault-tolerant gate implementation. Good candidates for CC codes are the Steane $[\![7, 1, 3]\!]$ code, or the newly optimized Bacon–Shor $[\![9, 1, 3]\!]$ code [146, 147]. The Bacon–Shor $[\![9, 1, 3]\!]$ code is based on the well-known Shor nine-bit code

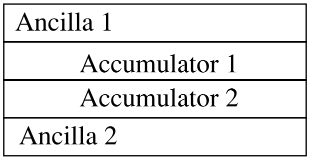


FIGURE 8.3: Hypothetical schematic of a processing element (PE) tile. Each high-level ancilla block is used to correct encoded data in the corresponding accumulator before and after the execution of a logical gate at the accumulator block. The PE tile requires two accumulator blocks to allow for two-qubit logical gates, where both qubits are teleported through the teleportation-based interconnect into either of the accumulator stations.

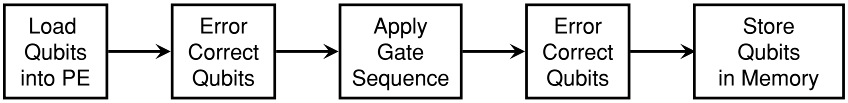


FIGURE 8.4: The five stages for an instruction execution from the perspective of a processing element.

[23] and allows very fast and efficient error-correction routines. The T gate (see Equation 2.14), however, is usually more difficult to implement as shown in Section 5.3. It requires the interaction of the logical qubit with a specially prepared encoded $A_{\pi/8}$ ancilla (also in CC), making the T gate essentially a two-qubit gate [125]. Many of the tiles in the processing region must be used to prepare the $A_{\pi/8}$ logical qubits used in the implementation of the T gate. Thus, when a T gate is executed the logical qubit and a ready $A_{\pi/8}$ qubit are teleported to two accumulators in an empty PE.

From the perspective of each PE, an instruction is executed through five stages shown in Fig. 8.4: (1) the logical qubits are loaded into an available PE; (2) the logical qubits are error corrected; (3) the gate implementation sequence is applied on the logical qubits; (4) again error correction is applied; and finally, (5) the logical qubits are sent to an available cache address.

8.2 QUANTUM MEMORY HIERARCHY

While classical memory hierarchies optimize for speed, given technologies of differing performance and cost (for example SRAM and DRAM), a *quantum memory hierarchy* optimizes for error-correction codes which can either facilitate computation or improve storage density. The memory hierarchy in quantum architectures exists because of error correction, it exists to provide the reliability necessary to fault-tolerantly encode and store quantum data for the duration of a given application. The lowest level structures of the hierarchy are designed to meet the speed and efficiency of the processing elements by accepting and storing encoded data residing in the higher levels of the hierarchy.

While the Steane $[[7, 1, 3]]$ and the Bacon–Shor $[[9, 1, 3]]$ codes seem to be best suited for computation, more efficient $[[n, k, d]]$ block codes (where $k > 1$) form multiple logical qubits together in a block of encoded physical qubits and can be used as a memory code (MC) for higher density storage. While block codes are very expensive for computation, their relatively large distance parameter d and compact n/k scale up with each level of encoding make them a promising candidate for memory codes. The caveat is that using different CC and MC codes calls for a considerably more complex transfer network when transporting a logical qubit resting in memory to an accumulator in the processing region. The transfer process of a logical qubit to a different location with a different encoding state code cannot be implemented only with the straightforward repeater-based interconnect used within each architecture region. The transfer network must provide for fast and efficient conversion between the CC and MC codes, as well

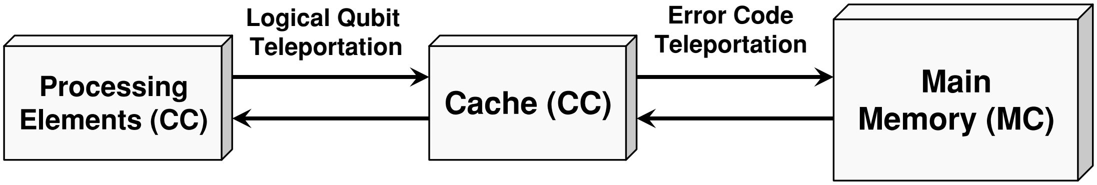


FIGURE 8.5: Memory hierarchy high-level concept. The concept is very similar to classical memory hierarchies; however, the separation between cache and processing elements is not spatial, but rather dependent of the encoding type. The error code employed by the cache is intended to match the speed and efficiency of the compute code employed by the processing elements.

as exploit temporal and spatial locality to effectively cache data in the CC code. The concept of the memory hierarchy is illustrated in Fig. 8.5.

When an instruction is ready for execution in an available PE, a check is made to determine if the logical qubits are in the cache. If so, the logical qubits are delivered to the processing element through the teleportation interconnect provided they have been successfully error corrected in the cache. If not, the logical qubit is transferred to the PE directly from the main memory.

Translating between the MC and the CC codes can be problematic—decoding and re-encoding leaves the data vulnerable and can produce correlated errors that our codes cannot correct. Fortunately, we can teleport from one code to another by encoding one of our maximally correlated pairs in one code and the second in the other [125]. The transfer region between memory and cache is one of the most interesting components of the memory hierarchy. This region transfers data encoded in code C1 to a second code C2 without the need to decode. Fig. 8.6 illustrates this concept. The transfer network teleports the data in C1 to C2, where C1 and C2 may be any two error-correcting codes. The code teleportation procedure works much the same way as standard data teleportation that is used for communication. A correlated ancillary pair is prepared first between C1 and C2 through the use of a multiqubit cat state (i.e., $(00\ldots0 + 11\ldots1)$). The data qubit interacts with the equivalently encoded ancillary qubit through a CNOT gate, and the two are measured. Following the measurement the state of the data is recreated at the C2 encoded ancillary qubit. This process is required every time we transfer a qubit from memory to the cache or vice-versa. The most important property of the transfer network is that C1 and C2 need not be two different codes such as a CC code and an MC code, but can be the same code at different levels of encoding between compute regions and memory regions.

Fig. 8.7 illustrates the steps taken when an operation needs to be applied to a number of logical qubits. The classical controllers identify an available PE and look for the logical qubits involved in the operation in the cache region, which stores logical qubits already converted to the

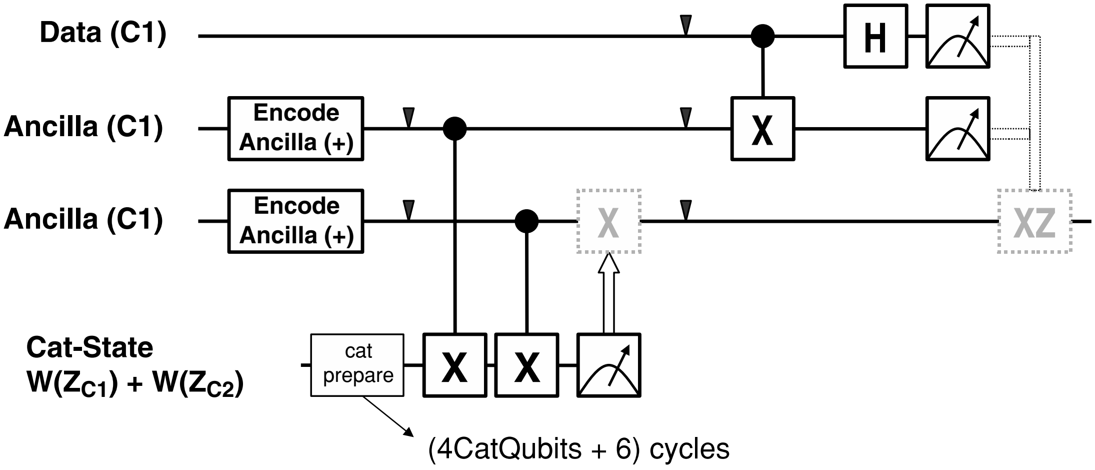


FIGURE 8.6: Code teleportation network from code C1 to code C2. C1 and C2 can even be the same error-correcting code, but different levels of encoding. The solid triangles denote an error-correction step. The cost of the "Cat-Prepare" gate in the bottom-most line is equal to the cost of preparing four cat qubits + six additional cycles. We refer to a cat qubit as a collection of n qubits prepared in an n-qubit cat state as described in Section 2.4. Note that this *is* the familiar teleportation network. The only difference is the creation of the EPR pair. Because C1 and C2 are different codes, we cannot create an encoded EPR pair by entangling them through a direct CNOT gate as shown in Fig. 2.9, but must measure their respective logical $\overline{X}$ operators and apply the corresponding gates (shown as the dashed X gate) for the EPR creation.

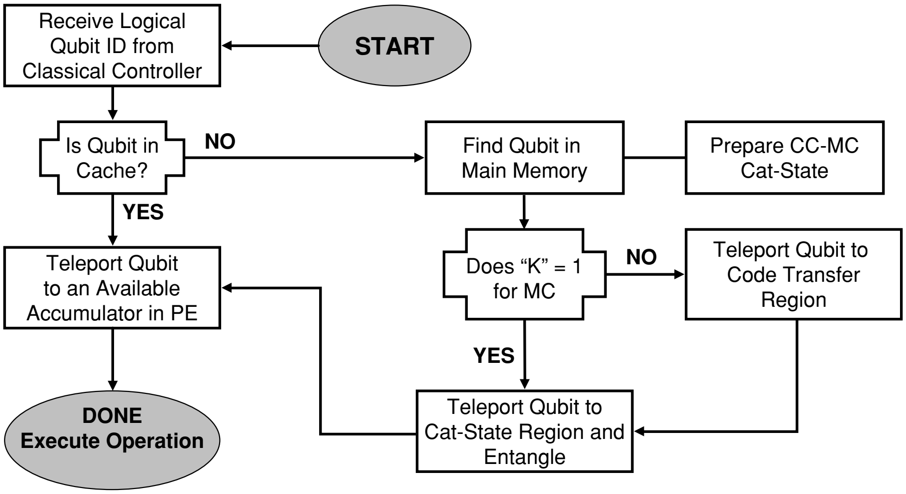

START
Receive Logical Qubit ID from Classical Controller
Is Qubit in Cache?
NO
YES
Find Qubit in Main Memory
Prepare CC-MC Cat-State
Teleport Qubit to an Available Accumulator in PE
Does "K" = 1 for MC
NO
YES
Teleport Qubit to Code Transfer Region
DONE Execute Operation
Teleport Qubit to Cat-State Region and Entangle

FIGURE 8.7: Cache read operation.

CC. If the qubits are there, they are teleported to the PE and the sequence shown in Fig. 8.4 is applied. If the qubits are not found in the cache region, they must be transferred from the main memory, where they are encoded with the memory code (MC) through the code teleportation procedure outlined in the steps of Fig. 8.7. In the architecture organization of Fig. 8.2 every region (the processing elements, the cache, the main memory, and the transfer region) is composed of logical qubit tiles interconnected by the programmable teleportation-based bus lines. When a cache hit occurs the classical control resources are focused on the teleportation channels that connect the cache and the PE, where second priority is given to transferring additional qubits to the cache. This, however, is a scheduling decision and currently no true schedulers exist for large-scale quantum computation.

The next stage of the architecture description is to implement efficient simulators that will allow us to fully exploit and parameterize the architectural design. Parameterizations of the cache read time, memory access time, operations times, and qubit failure rates are not only functions of our error-correction choices, but also functions of interconnect design and the structure of the architectural elements. Some important questions we need to answer involve the error-correction choices, cache replacement rules, and the availability of classical resources.

Error-Correction Choice: A key decision is to determine the error-correction codes for the main memory (MC) and the computation (CC). Much depends on the parameters and the properties of an error-correcting code: the time of execution for logical operations, the size of each tile, the time of failure of the application, and most importantly the coupling between communication and computation at the high level. In addition, at the physical level, the data communication patterns of different error-correcting codes differ wildly and may affect the efficiency of the code itself. For example, the Bacon–Shor $[[9, 1, 3]]$ code is a recently optimized version of the $[[9, 1, 3]]$ code described in Section 5.2 that allows almost no physical qubit movement between the two-qubit physical gates during encoded state preparation and error-correction procedure at the first level of encoding. Perhaps, the Bacon–Shor $[[9, 1, 3]]$ code coupled with a different smaller code at the next level of encoding may offer a much more efficient and reliable computational tile than using the same CC code from one level of encoding to the next.

Cache Replacement Rules: It is important to understand clearly the replacement rules when the cache is full. The application being executed is known in advance, so our compiler will be able to schedule the operations and the cache usage statically at compile time; however, preparation of the interconnect channels on demand and time to error correct can only be predicted with a limited accuracy. There is always a certain probability of failure which leads to stalling.

Code Conversion Choices: When is the best time to convert from CC to MC? We assume that qubits will not be sent back to the cache after usage, *unless* they are needed within the time of failure for a qubit resting in the cache. They will be teleported directly to the main memory in reverse of the operations outlined in Fig. 8.7. The cache can perform memory correction but with limited classical resources; thus qubits should not stay there for long.

Classical Resource Availability: What are the available classical resources? In our previous work [27] we have assumed unlimited classical control signals, which take the form of lasers for ion traps. If we have a very small number of lasers available, however, the replacement rules in the cache and the PE units will become extremely important. Currently the replacement rule is to send a qubit directly to memory when the computation is finished and only send it to the cache if it is needed before the cache storage failure rates. Sending it to memory may prove advantageous because it is designed to store qubits for long periods of time with very small number of laser resources. A careful balance must be reached, however, between the cost of code transfer from memory and memory storage.

8.3 QUANTUM SEARCH: QUANTUM ADDRESSING SCHEME FOR CLASSICAL MEMORY

The separation between memory and compute regions discussed so far is a system-level separation that provides a computer architect with various knobs to turn when optimizing a specific quantum application. A different and interesting separation between memory and computation is offered by the implementation of the quantum search algorithm known as *Grover's algorithm* for searching an unsorted database of N entries [10]. While classically the search would take $O(N)$ operations, quantum mechanically the cost is $O(\sqrt{N})$. A naive classical architecture for searching a database is to store all data entries into a long-term memory unit and perform a maximum of N LOAD operations from the memory to the processor for each entry in the database. The freshly loaded entry string is then compared to a solution string stored in the processor.

The main engine for the quantum searching algorithm is the *oracle* operator O whose action can be written as

$$|x\rangle \rightarrow (-1)^{f(x)}|x\rangle, \tag{8.1}$$

where x is the index register which points to the data entry in the database. The input x into the search function f returns 1 if x is a solution to the search problem and 0 otherwise. The index register $|x\rangle$ is composed of $\log N$ qubits, where each bitstring state $|x_i\rangle$ in the superposition indexes a data entry. Thus, the function of the oracle is to flip the sign of the index register if a solution is found. Along with the n-qubit index register (where $n = \log N$), the processing unit of the search algorithm involves an l-qubit register to hold an l-bit data entry initialized to $|0\rangle$ and an l-qubit register that stores the solution string.

What makes the architecture of the quantum search implementation interesting is that data can be stored classically, and thus reliable storage of the database is not a concern. For the

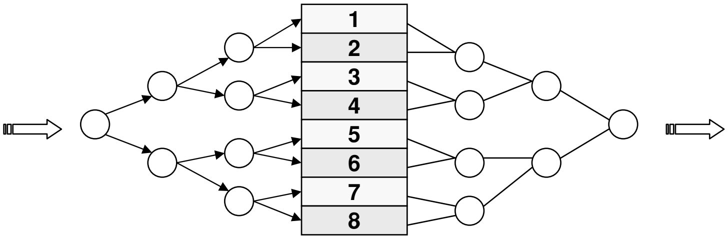


FIGURE 8.8: Schematic for classical memory that is addressed with qubits. The figure illustrates the concept with a three-qubit address memory of eight data entries. Each circle represents an ancillary qubit used as a switch to route the input index register to the correct data entry.

polynomial speedup to be achieved, an N-entry memory must be addressed quantum mechanically by $\log N$ qubits [38]. Fig. 8.8 illustrates the concept with a three-qubit address memory of eight data entries. Each circle represents an ancillary qubit used as a switch to route the input index register to the correct data entry. Each of the data register qubits is routed to the corresponding entries in the memory based on the state of each qubit switch, which is determined by the index register in the processor. The data register qubits enter at the left and exit at the right of Fig. 8.8. If a particular switch is in the superposition state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, then the data qubit is routed in both directions. In this manner the LOAD operation returns a superposition of data entries that can be compared with the l-qubit register that stores the solution string. To match one of the data strings from memory with the solution string stored by the l-qubit solution requires $O(\sqrt{N})$ LOAD operations.

In reality, the searching of an unsorted database quantum mechanically is not more efficient than storing and searching the database classically. The quantum addressed classical memory requires $O(N \log N)$ ancillary qubit switches, in addition to the operations overhead once the data is loaded into memory. Should error correction be needed for storing and searching through large databases, the modestly polynomial improvement over classical searching will be overwhelmed by the exponential slowdown due to error correction. For technology parameters, for example, that are three orders of magnitude below the accuracy threshold value of the $[\![7, 1, 3]\!]$ code, we would need to insert error correction for any database greater than half a million entries. The database size allowed before error correction is needed to be dropped to less than ten thousand if the technology parameters are at the threshold value.

Should qubits become, as easily and cheaply implementable as classical logic transistors are, then it may become interesting to implement quantum addressable memories. One use of a quantum addressing scheme is prefetching classical registers in the classical memory hierarchy. In a single clockstep the quantum address will be able to fetch an entire superposition of classical registers at any location of the memory.

CHAPTER 9

Case Study: The Quantum Logic Array Architecture

The authors of reference [27] describe a homogeneous, tile-based quantum architecture based on the ion-trap technology, that overcomes primary challenges of reliability, scalability and efficient quantum resource distribution. The quantum logic array (QLA) model integrates concepts for a large-scale quantum architecture design to enable substantial performance improvements critical to supporting full-scale applications such as Shor's factorization algorithm.

In this section we describe the QLA architecture as a case study for a large-scale quantum architecture design and extend the example further to include system parameters when the effects of specialization are introduced in the architecture design. The QLA quantum computing system, as shown in Fig. 9.1, is a homogeneous array of logical qubits implemented as self-contained computational tiles, and connected through the teleportation-based communication channels that utilize the concept of quantum repeaters as discussed in Chapter 6.

At the lowest level, the QLA is based on trapped-ion technology. Fig. 9.2 demonstrates the abstraction of the physical ion-trap layout in studying the QLA scheme. The layout can be represented as a collection of trapping regions connected together through shared junctions. A fundamental time step, or a clock cycle, in an ion-trap computer can be defined as any physical, operation (one-bit or two-bit) on a single ion-qubit, a basic move operation from one trapping region to another, and measurement. Table 9.1 summarizes current experimental parameters and corresponding optimistic parameters for ion traps. In our subsequent analysis we will assume that each *clock cycle* for a fundamental time step has a duration of 10 μs, failure rates are 10^{-8} for single-qubit operations and measurement, 10^{-7} for CNOT gates [104], and 10^{-6} per fundamental move operation. The movement failure rate is expected to improve from what it is now as trap sizes shrink and electrode surface integrity continues to improve. We assume trap sizes of 5 μm each [148], and on the order of 10 electrodes per trapping region [108], which gives us a trapping region dimension (including the junction) of 50 μm. The parameters chosen for the example are optimistic compared to [141] and [79]. Both of those papers assume more pessimistic near term parameters which are useful for building a 100-bit

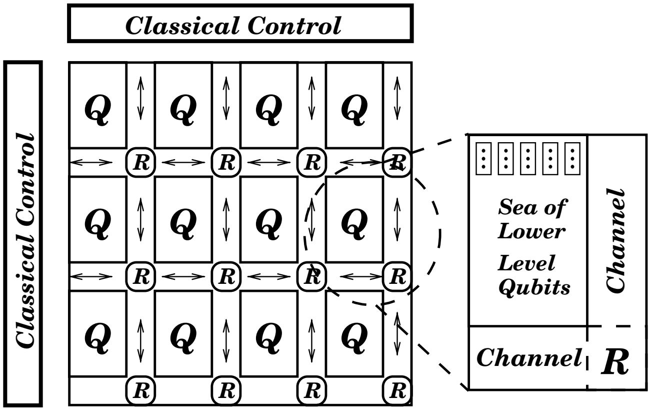


FIGURE 9.1: High-level view of the QLA architecture.

prototype, but probably not a scalable quantum computer that can factor 1024-bit numbers using Shor's algorithm. Based on the quantum computing ARDA roadmap [52], we feel justified in using aggressive parameters when looking 10–15 years into the future.

9.1 THE LOGICAL QUBIT DESIGN

The structure of the logical qubit in the QLA is driven by the ion-trap characteristics shown in Table 9.1, which place us significantly below the accuracy threshold value required by the threshold theorem. These parameters are optimistic, but not fundamentally impossible. Particularly important is the fact that the lifetime of an ion (measured in seconds and even minutes) is much larger than quantum operations which are on the order of tens of microseconds. These

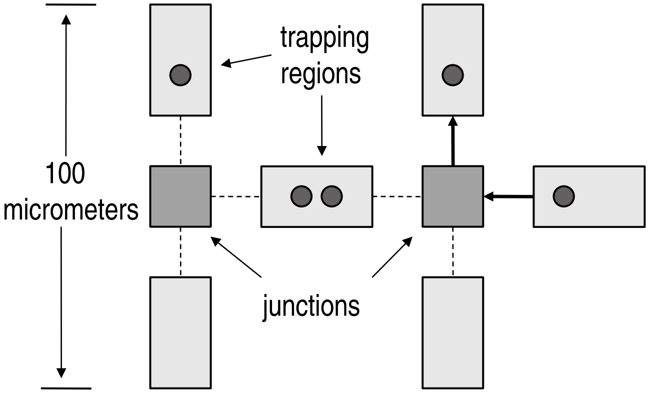


FIGURE 9.2: Our abstraction of the ion-trap layout. Each trapping region can hold up to two ions for two-qubit gates. The trapping regions are interconnected with the crossing junctions which are treated as a shared resource.

TABLE 9.1: Column 1 Gives Estimates for Execution Times for Basic Physical Operations Used in the QLA Model. Currently Achieved Component Failure Rates are Based on Experimental Measurements at NIST with $^{9}\mathrm{Be}^{+}$ Ions, and Using $^{24}\mathrm{Mg}^{+}$ Ions for Sympathetic Cooling [60, 100]. All Parameters are Followed by Their Projected Parameters in Parenthesis, Extrapolated Following Recent Literature [52, 105, 104], and Discussions with the NIST Researchers; These Estimates are Used in Modeling the Performance of Our Architecture.

	TIME μs	FAILURE RATE
OPERATION	NOW (FUTURE)	NOW (FUTURE)
Single gate	1 (1)	10^{-4} (10^{-8})
Double gate	10 (10)	0.03 (10^{-7})
Measure	200 (10)	0.01 (10^{-8})
Movement	20 (10)	0.005 (5×10^{-8})/μm
Split	200 (0.1)	
Cooling	200 (0.1)	
Memory time	10 to 100 s	
Trap size	$\sim$ 200 (1–5) μm	

relatively low memory error rates allow us to significantly reduce the area of a logical qubit by reducing the parallelism within a single error correction cycle, and the ancillary qubits required by the error-correction algorithm.

Fig. 9.3 shows the full implementation of a level 2 qubit tile. To reduce communication and complexity, we chose to model each logical qubit as a self-contained hardware structure that requires no external quantum resources to perform logical gates and state stabilization (i.e. error correction). This will allow an application level compiler to divide the quantum program into distinct data independent threads that are executed on separate computational units, which are simply the logical qubits in a homogeneous architecture such as the QLA. There are two high-level ancilla blocks in a level 2 qubit, which allows the error correction of two level 2 qubits when a two-qubit gate is executed inside a single-qubit tile. The two sets of high-level ancilla are necessary in computational tiles to ensure that both logical data qubits are error corrected immediately after the execution of a two-qubit gate without stalling the application execution.

A single data logical qubit at level 2 is built by encoding 7 level 1 qubit blocks with the Steane $[\![7,1,3]\!]$ code. A level 1 qubit block is shown at the top of Fig. 9.3. We choose the $[\![7,1,3]\!]$ code because it allows the implementation of a large set of logical gates *transversally*, with the exception of the T gate (see Section 5.3). This means that a logical quantum bit-flip gate on our qubit can be implemented by applying 49 physical bit-flip gates on the ions, in

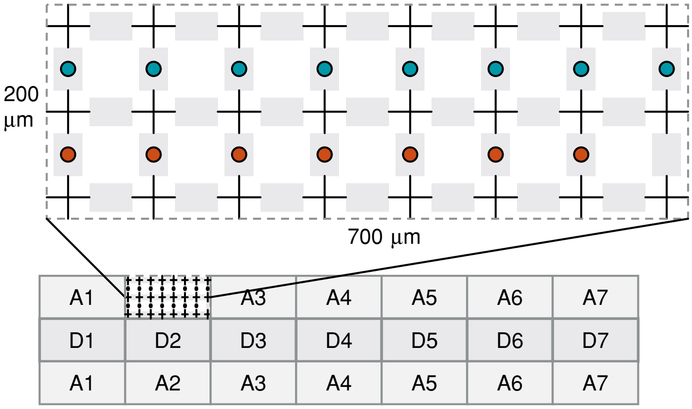


FIGURE 9.3: The logical qubit: seven groups of three level 1 blocks make a single level 2 logical qubit (middle). The two identical conglomerations on the sides are ancillary blocks used for error correction. The shaded boxes of the level 2 qubit are the encoded data level 1 blocks, which are supported by their respective level 1 ancilla blocks.

parallel. A logical CNOT gate is implemented by bringing 49 ions from some qubit tile A in the same trap as the 49 ions in qubit tile B. After 49 CNOT gates on the joined ions, the two sets are error corrected by the ancilla on both sides of the data region in a level 2 tile. The ancilla preparation network at level 2 does not require specially designated verification blocks, as the errors are detected during lower level syndrome extractions [128].

Considering communication, the level 1 error correction circuit shown in Fig. 2.17 will take 154 cycles, where each cycle is in the order of 10 μs, and can be as large as 0.003 s per error correction procedure at level 1. In our time estimates we choose to provide a single laser per level 1 block. The latency introduced by serializing the level 1 circuit is not significant since a maximally parallelized circuit would take approximately 127 cycles per error correction procedure. A fully serialized error correction at level 2 will last approximately 0.3 s, which is two orders of magnitude more than the time to error correct at level 1.

We have made the following assumptions when extracting the error syndromes for both level 1 and level 2 qubit blocks: (1) two syndromes are extracted in *serial* for both X and Z errors; and (2) we assume that in the case of a nontrivial syndrome the next extracted syndrome will match it, thus we can proceed with the error correction step. Since our logical qubit at level 2 is equipped with parallel syndrome extraction, assumption (a) makes Eq. 9.1 an overestimate of the final latency:

$$T_{L,\mathrm{ecc}} = \begin{cases} 2 \times T_{L,\mathrm{synd}}, & \text{trivial syndrome} \\ 2(2T_{L,\mathrm{synd}} + T_1 + T_{L-1,\mathrm{ecc}}), & \text{nontrivial} \end{cases} \tag{9.1}$$

where $T_{L,\text{synd}}$ is the time to extract a syndrome at level L, which is a function of the time to prepare the logical ancilla block. T_1 denotes the time of a logical one-qubit gate, and $T_{L-1,\text{ecc}}$ is the time for a lower level error-correction step that follows each level L logical gate. A syndrome is considered *trivial* if no errors are detected on the data, in which case no error correction is necessary and the syndrome is not repeated to reconfirm the location of any found error. In contrast a syndrome is considered *nontrivial* when one or more errors are detected in the data block.

Numerical simulations of a level 2 qubit showed that a nontrivial syndrome was measured for level one with a rate of $3.35 \times 10^{-4} \pm 0.41 \times 10^{-4}$, and for level two at a rate of $7.92 \times 10^{-4} \pm 0.81 \times 10^{-4}$. Our simulations did not yield a syndrome repetition of more than two times before the error correction step at the optimistic error rates for ion traps. Thus, it is a reasonable assumption that in the case of a nontrivial syndrome we require at most one more syndrome extraction before we are ready to apply the correcting gate. Taking a weighted average of the two cases in Eq. 9.1 we determine a level 2 error correction time of approximately 0.3 s. As shown in Fig. 5.9, using level 2 recursion with this qubit tile design is sufficient for factoring numbers as large as 2048-bit modulus.

We used QASM-TOOLS, formerly known as ARQ, to empirically compute p_{th} at level 2 for the QLA logical qubit. Our results, displayed in Fig. 9.4, show that the failure probability of a single one-qubit logical gate rapidly drops to zero at component failure rates lower than

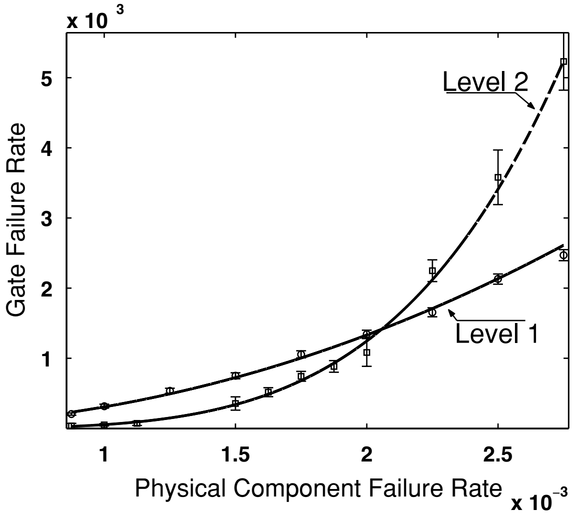


FIGURE 9.4: Estimate of the failure probability ($\hat{y}$ axis) of a single logical one-qubit gate followed by recursive error correction procedure at levels 1 and 2. The $\hat{x}$ axis denotes individual physical component failure rates.

$p_{\text{th}} = (2.1 \pm 1.8) \times 10^{-3}$. Above this value the rapid decrease in the reliability of our system as recursion increases can be attributed to the additional resource overhead of recursion.

The estimated threshold failure probability is much higher than the theoretical estimate of 7.5×10^{-5} computed in [149] for several reasons: (1) the structure of the qubit is optimized for the error correction circuit and may vary for different codes; (2) the high reliability of ion-trap memory has allowed us to significantly reduce the overall area and ancillary resources required; (3) the fixed, low movement error probability, and the fact that we made the design decision to never physically move the data, pushed our qubit's threshold closer to the 9×10^{-3} threshold value estimated by Reichardt [128]. We observed no failure at level 2 recursion as the physical component errors approached the expected ion-trap parameters from Table 9.1, which was expected. Reevaluating Eq. (5.13) with the empirical value for p_{th} we get an estimated level 2 reliability approaching the remarkably low value of 10^{-21}.

9.2 LOGICAL QUBIT INTERCONNECT

A logical two-qubit gate between level 2 qubits $Q1$ and $Q2$ is executed by moving all 49 physical ion qubits that encode qubit $Q1$ to the computational tile where qubit $Q2$ resides. If the application being executed is the factoring of a 1024-bit number using Shor's factoring algorithm, $Q1$ could be moving as far as 0.5 m (or 256 logical qubits) across the ion-trap chip. The long-distance communication channel employed by the QLA architecture is the repeater-based teleportation protocol described in Chapter 6, where a repeater station is placed between every logical qubit tile. The ultimate purpose of the repeater-based channel is to create a single EPR pair (i.e., two ions in the maximally entangled state $(|00\rangle + |11\rangle)/\sqrt{2}$) such that one of two qubits is at the location of qubit $Q1$ and the other one at the location of qubit $Q2$. An EPR pair distributed in such a way is required for each of the 49 ion qubits of qubit $Q1$ (not necessarily created in parallel) such that each of the 49 qubits can be teleported to the computational tile of qubit $Q2$.

Each EPR pair that connects to adjacent repeater stations is created in the middle where two ion qubits are entangled and separated to the two opposing ends. There are many ways to achieve entanglement between two ion qubits. In one scalable entanglement technique for ion traps [150], the ion qubits are initialized to the ground state $|00\rangle$ and placed in the same trap. An entangling controlled-phase gate adapted for coupling two ions together is used to implement a CNOT gate (also known as the Mollmer–Sorrensen entangling gate) placing the ions in the intermediate maximally entangled state $(|01\rangle + |10\rangle)/\sqrt{2}$ [99], which can be followed by single-qubit rotations to place the two-ion-qubit state in the desired EPR state. An alternative proposal [151] combines the features of optical lattices and ion traps, where individual ions are entangled through a common interaction with a pulsed, high-strength optical lattice. The

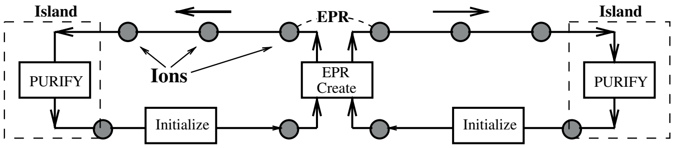


FIGURE 9.5: Detail of a channel between two repeater stations. The channel is a two-way ballistic transport region, where the EPR pairs are created in the middle and distributed in a pipeline fashion to the two Island/Reapeater stations.

benefits of this proposal are that the two ions do not need to be physically together for the entanglement operation; however, using this proposal would drastically change the underlying physical microarchitecture we have described so far. The Molmer–Sorrensen entangling gate has been used recently in two simultaneous, independent experiments that demonstrate the experimental realization of quantum teleportation using trapped ions [78, 77]. To model EPR creation we assume that two ions are brought together and the actual EPR generation routine is a resource that can be abstracted as a single box (as shown in Fig. 9.6) whose implementation can be modeled as the familiar entangling circuit shown in Fig. 2.9 from Section 2.4.

To optimize space and performance, we can model the channels between each island as a two-way ballistic transport region as shown in Fig. 9.5, which also illustrates the pipeline purification protocol employed by the QLA architecture for purifying a single EPR pair. The basic idea of purification [134] is to use several copies of lower fidelity EPR pairs to *distill* a single high fidelity EPR state that can be used for teleportation. Generally, it is not possible to create a perfect EPR state with unit fidelity mostly because of the usage of noisy gates in the process of creation and the transmission of the two qubits through the noisy physical channel between each repeater station.

If the initial preparation fidelity is high enough, by applying successive purification steps an EPR pair can be purified to an arbitrarily high fidelity. The pipeline purification sequence works by designating one EPR pair as the data pair which is continually purified in round-robin pipeline fashion by the additional ancillary EPR pairs. We assume to have enough ion resources in the pipeline to handle the maximum number of required purification steps without having to wait for the creation of new EPR pairs before each successive purification step. The original purification protocol was formulated by Bennett [134] where the efficiency of

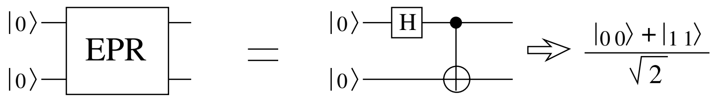
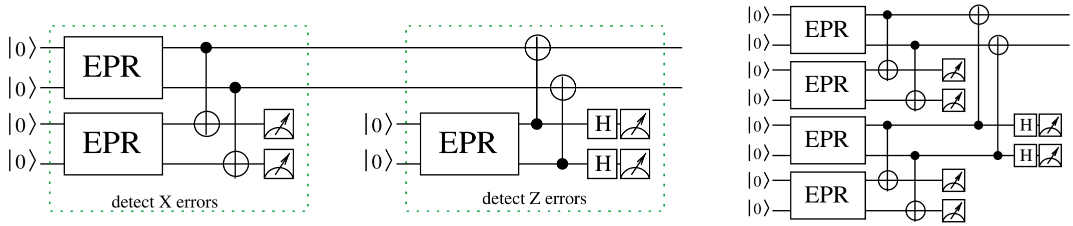


FIGURE 9.6: EPR generation can be abstracted as a box or modeled using a Hadamard gate followed by a CNOT gate between two qubits.



(a) (b)

FIGURE 9.7: (a) The data EPR pair (top) is created in parallel with an additional ancillary EPR pair used to detect bit-flip errors first. Phase-flip errors are detected with a third EPR pair or the previous ancillary pair reinitialized. (b) Four EPR pairs are created, two of which are used to check the other two for bit-flip errors. This is followed by the detection of phase-flip errors on the two EPR pairs remaining.

purification depends highly on reliability of the physical gates that make up the protocol (namely Hadamard and CNOT gates) and the inital fidelity of the EPR pair [135]. We use the recursive fidelity equations in [135] (where the first detailed analysis of quantum repeaters is performed) to study the implementation employed by our architecture for purification protocols whose efficiency depends as much on the gate reliability as it does on the type of errors that occurs and how the errors accumulate in the EPR states before and during purification. This allows us to distill higher fidelity EPR pairs with fewer purification steps. The purification circuit is shown in Fig. 9.7 where there are two possible network choices. Using the first network in Fig. 9.7(a) and limiting purification to be only between two adjacent islands we determine sufficient island distribution to be one island at every logical level 2 qubit.

After its creation, or even during the purification procedure the data EPR pair accumulates bit-flip of phase-flip errors that can place it in any of the four possible states known as the four *Bell states* $\{|\Psi_+\rangle,\ |\Psi_-\rangle,\ |\Phi_+\rangle,\ |\Phi_-\rangle\}$ [152]:

$$\begin{aligned}
|\Psi_+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \rightarrow \text{.............no errors}\\
|\Psi_-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \rightarrow Z \text{ error on } q1 \text{ or } q2\\
|\Phi_+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \rightarrow X \text{ error on } q1 \text{ or } q2\\
|\Phi_-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \rightarrow \text{both } X \text{ and } Z \text{ errors.}
\end{aligned} \tag{9.2}$$

The purification circuit shown in Fig. 9.7(a) uses one ancillary EPR pair to check the state $|\Psi_+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ first for bitflip errors and then uses another ancillary EPR pair to check the state for phase-flip (i.e., sign) errors. After interaction with the data EPR pair through the CNOT gates the two ancillary qubits are measured where odd parity for either X or Z error checks will indicate that there is an error in the data EPR pair. In the case of an error, the data EPR qubits are recycled in the pipeline and the next ancillary EPR pair resumes the function of the data, which is then purified. Each successful purification step increases the likelihood that the data EPR pair is free of errors, thus it increases its fidelity. The principle is the same as throwing a weighted coin with unknown weight for obtaining either heads or tails. Each time the coin lands heads given that it has landed heads the previous throw, the probability that the coin is weighted toward heads increases.

An alternate purification procedure is shown in Fig. 9.7(b), where four EPR pairs are prepared in parallel at the beginning. The data EPR pair is at the top and it is checked in parallel with an additional EPR pair for X errors. If both pass, the data EPR is checked for Z errors. Although we have not studied this protocol, it may have the potential to offer better purification efficiency by ensuring that the ancillary EPR pair used in the Z error detection is checked against X errors. The interaction between the two EPR states when checking Z errors will cause an X error in the ancilla to propagate to the data through the CNOT gates, which would remain undetected when the teleportation procedure is executed. The implementation of the network in Fig. 9.7(b) would require different EPR generation and island structure where each island would need to hold more than one data EPR pair at each node. In reality, any of the four Bell states can be used for teleportation, thus the purification efficiency can be further improved if we allow X or Z errors to remain and use the subsequent purification steps to ensure that indeed the X and Z errors detected in the previous step are present. In such cases we know which of four Bell states our EPR qubit is in, and modify the teleportation protocol accordingly, where the modification consists of different interpretations of the 2-bit bitstring that signifies how to apply the correcting X and Z gates on qubit $q3$ in Fig. 2.8 at the end of the teleportation protocol.

Suppose we define the *scope* of an EPR pair as the distance between each of the two EPR qubits as a function of the number of teleportation islands (i.e, repeater stations) between them. If the entire channel between logical qubits $Q1$ and $Q2$ is divided by K repeater stations, the ultimate goal is to create a number of single EPR pairs with a scope of K islands that connect the two logical qubits. EPR pairs that connect two adjacent repeater stations have a scope of zero.

The first tradeoff arises as the number of ion-qubit resources required to distill a single high fidelity EPR pair at any scope (note: we assume the network construction shown in Fig. 9.7(a)). Clearly, the minimum resources required are four ion qubits, two for the data EPR pair and two for the ancillary pair used for purification. The ancillary pair is continuously reprepared for each purification step. Alternately, the maximum number of resources used can be reached by creating all EPR pairs required for j purification steps which would require $\eta(2 \times 3)^j$ ion qubits, where η is some constant that takes into account the possibility of failure at some stage

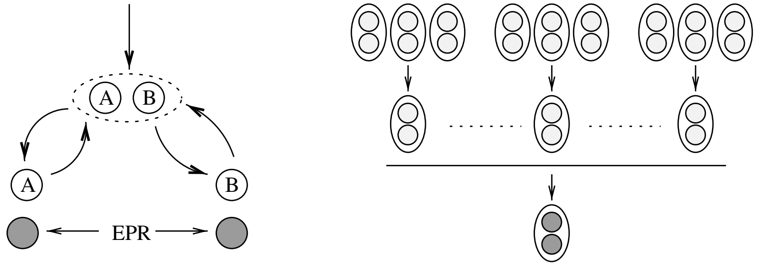


FIGURE 9.8: Minimum and maximum number of resources needed to purify a single EPR pair.

in the purification. The number of ion qubits needed is bounded by

$$4 \leq \text{resources} \leq \eta(2 \times 3)^j. \tag{9.3}$$

Both concepts are shown in Fig. 9.8, where the protocol that uses the minimum resources is shown on the left-hand side. Ion qubits A and B are continuously reprepared and interacted with the data EPR pair at each step of purification. In the scheme on the right-hand side, a two-step purification tree is shown, where 18 ion qubits are prepared into three groups of 3 EPR pairs used for the first purification step. After the first step, three purified EPR pairs are left and used to further distill a single EPR pair. While the first protocol uses far less resources, the final fidelity of the data EPR pair is severely limited by the fact that the ancillary EPR pair is continuously reprepared retains the same level of noise throughout the purification process. In the second protocol, on the other hand, the data and ancillary EPR pairs are equally purified at each step, and a much higher fidelity is achieved for the final EPR pair. This, however, is at the expense of high ion-qubit resources, and a complex microarchitecture that supports movement of all EPR pairs at each purification step. The pipeline approach we use as shown in Fig. 9.5 allows sequential purification without memory cycle delay between each purification step. By avoiding recursive purification and pipelining the ancillary EPR qubits, the QLA is able to utilize a significantly reduced bandwidth requirement for each distillation of EPR pairs between two adjacent repeater stations.

A second important trade-off arises when deciding the scope at which EPR pairs are purified. There are three possible ways to connect a source and a destination separated by K repeater islands, such that the final teleportation step of the data qubit between the source and the destination is teleported with the desired threshold fidelity required for error correction:

1. A *purely linear* approach, which distills high-fidelity EPR pairs only between adjacent islands to some fidelity F that will allow $O(\log K)$ teleportation hops (see Fig. 6.2) to be performed such that final fidelity of the data teleported is within the threshold value. The total time to achieve a given relatively large distance varies as the separation between repeater islands is changed. As the separation decreases, purification will be followed

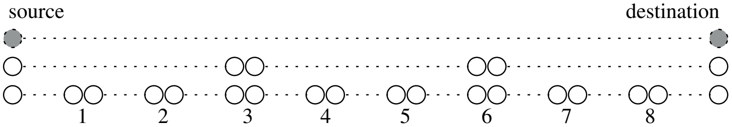


FIGURE 9.9: Nested purification protocol as described in [135]. We have three nesting levels, where the source and the destination are separated by eight repeater islands. At the most bottom level EPR pairs are created to connect three islands and are used to purify each other. The purified EPR pairs are further connected at the second level, to span the entire communication channel. EPR pairs at a given nesting level are constantly recreated to purify an EPR pair at the corresponding level.

by a greater number of teleportation hops between the source and the destination, thus more purification is needed to achieve a higher EPR starting fidelity. Alternately, as the separation increases, there are less teleportation hops, but the data and ancillary EPR pairs travel longer in the pipeline, thus more purification is needed to reduce the fidelity. It is an interesting tradeoff for a system designer to explore, and offers an opportunity to design a reconfigurable dynamic interconnect.

2. A *nested, semilinear* approach, which distills EPR pairs at different nesting levels with an increasing scope per level. This method was analysed in detail in [135]. At the lowest nesting level EPR pairs are created with a scope of m junctions, which are used to purify an EPR pair with the same scope at the second nesting level. The freshly purified scope m EPR pairs are connected to create an EPR pair with scope km for some other constant k, which are then used to distill a single EPR pair of scope km at the third nesting level. This process is repeated until we have a single EPR pair connecting the source and the destination as shown in Fig. 9.9.
3. Finally, we can create EPR pairs directly between the source and the destination without purifying at any intermediate scope. The purification is performed for an EPR pair that spans the source and the destination until a desired fidelity is reached.

The QLA architecture utilizes Approach 1, where we find that at the optimistic technology parameters for ion traps, the distances required for communication when factoring a 2048-bit number are attainable without the need to purify EPR pairs with scope higher than zero. Although Aproaches 2 and 3 offer much longer final distance, the pipelined linear approach offers a comparatively smaller bandwidth by providing only a single pipeline based channel from the source to the destination. In addition, the structure of the repeater islands is very simple when EPR pairs do not need to be purified at scopes higher than zero—all data EPR pairs are fixed in space until the entire purification procedure ends and the teleportation is complete. Approach 3 was studied in detail in a recent paper from Berkeley [153], where the creation of hundreds of EPR pairs is required between the source and the destination to purify EPR pairs that span the source and the destination. The authors achive a design that provides a potentially, very large communication distance, by utilizing the tree purification structure.

The latency cost of communication between logical qubits is critical for the success of the entire architecture during the execution of an application. We have made a design decision that ballistic transport must be used for moving ions within a logical qubit, and teleportation will be preferred when moving across larger distances in order to keep the failure rate due to movement below the threshold amount. Since EPR pairs are required for teleportation, we can reduce communication costs to a minimum if we have the required number of EPR pairs available at a logical qubit at the same time that it is ready to move. Fortunately, this is possible because of the high cost of error correcting the logical qubits. We can create, purify and transport the required EPR pairs to their respective qubits while they are undergoing error correction. But can this be done at a large scale?

To answer this question, we can use a tool to schedule the movement of EPR pairs in QLA [27]. One channel is assigned to carry the created EPR pairs to their destinations and another channel to return the used EPR pairs. Within each channel, the EPR pairs are pipelined. We define the bandwidth of QLA's communication channels as the number of physical channels in each direction—the channel shown in Fig. 9.5 has a bandwidth of 2. The goal of the scheduler is to find paths between logical qubits to transport all the required EPR pairs within the time it takes to perform a level 2 error correction.

The scheduler is heuristic, greedy scheduler that works by grabbing all available bandwidth whenever it can. However, if this means that the scheduler cannot find the necessary paths, it will back off and retry with a different set of start and end points. A simple approach to doing a two-qubit gate between logical qubits A and B would be as follows: teleport A to B's physical location, perform the gate and teleport it back. An optimization that the scheduler incorporates is that it only moves logical qubit A back if necessary. As a result, the logical qubits *drift* from one location to another. This adds a level of complexity to the scheduler, but at the same time reduces the amount of movement that the qubits are subjected to. With all of the above considerations in the scheduler, we found that given two channels in each direction (i.e., a single-pipeline structure), we could schedule communication such that it always overlapped with error correction of the logical qubits. The end result is reliable movement over arbitrarily large distances with minimal overhead. Table 9.2 summarizes the performance of the homogeneous QLA architecture when executing Shor's quantum factoring algorithm.

9.3 SPECIALIZED QLA ARCHITECTURE: CQLA

In the QLA computation can occur at any logical qubit tile, where each logical gate is followed by an error-correction procedure. To preserve homogeneity and maximum flexibility for large-scale applications each logical qubit is accompanied by the necessary error-correction auxiliary

TABLE 9.2: System Numbers for Shor's Algorithm for Factoring an N-bit Number Using the Circuit Descriptions of [154, 155] and the QLA Microarchitecture Model. The QLA Chip Area is Determined by the Number of Logical Qubits and Channels.

	$N = 128$	$N = 512$	$N = 1024$	$N = 2048$
Logical qubits	37,971	150,771	301,251	602,259
Toffoli gates	63,729	397,910	964,919	2,301,767
Total gates	115,033	1,016,295	3,270,582	11,148,214
Area (m^2)	0.11	0.45	0.90	1.80
Time (days)	0.9	5.5	13.4	32.1

qubit resources such that both accumulators in each tile can be error corrected in parallel. In this manner, each logical qubit tile has a ratio of (1 : 2) between the number of physical ions used to store encoded logical data and the number of physical ions used to store encoded high-level ancilla for error correction.

In Chapter 8, we discussed the possibility of specialized regions in the architecture to perform computation and storage in separately constructed logical qubit tiles. We even speculated that it may be beneficial to encode data differently between compute tiles and memory tiles, a design choice which may help us reduce the area introduced by the homogeneous architecture, and hopefully improve the time performance of the computer. Perhaps, the simplest way to reduce the area requirement is leaving the level of recursion and the chosen error-correcting code unchanged, but designate some qubit tiles for computation and some for data storage.

Such a separation between memory and computation introduces a very important concept which is counterintuitive to the classical architecture specialization model: the computational tiles that allow encoded gates to be applied on the data contain a greater amount of error correcting resources in order to allow faster error correction after each logical gate. Higher physical ion density in terms of number of ions that store data per unit area, can be achieved in the memory region by increasing the ratio between physical ions that store data and physical ions used to correct the encoded data as shown in Fig. 8.2. By surrounding a single logical ancilla block by eight logical data blocks to form one memory tile, we greatly reduce the turn-around efficiency of error correction per logical data, but we increase the ratio of data per ancilla from (1 : 2) to (8 : 1) [28].

The underlying assumption is that memory errors are a second-order event and the probability that an ion qubit will fail while waiting for the next error-correction cycle is within the accuracy threshold value of the $[[7, 1, 3]]$ code. Additionally, when a logical data residing in a memory tile is needed for gate execution while waiting for the next error-correction procedure, the teleportation of the logical qubit combined with the error the data has accumulated while waiting may introduce too many physical errors for the computing tile to recover the logical qubit once the data is teleported there. A system scheduler must issue a "stall" for the operation and wait for the logical data to complete an error-correction cycle in memory before it can be teleported to the compute region, where it must be immediately error corrected upon arrival.

To test the specialization model as compared to the homogeneous QLA architecture, we scheduled the quantum modular exponentiation component of Shor's factoring algorithm for several problem sizes. Quantum modular exponentiation is the most time consuming part of Shor's algorithm, and the Draper carry-lookahead adder is its most efficient implementation [154, 155]. This adder comprises single-qubit gates, two-qubit CNOT gates and three-qubit Toffoli gates and is heavily dominated by Toffoli gates. The time to perform a single fault-tolerant Toffoli is equal to the time for fifteen two-qubit gates, each of which is followed by an error-correction step. Table 9.3 shows the area savings that can be achieved when using

TABLE 9.3: For Various Size Inputs, This Table Shows How the QLA Performs for Modular Exponentiation. The Space Saved Due to Compressing The Memory Blocks and Separating Memory and Compute Regions is Shown as Compared to Prior Work [27]. The Gain Product is Compared With the Homogenous Architecture, the QLA, Which has a Gain Product of 1.0 and a SpeedUp of 1.0. A SpeedUp of 0.54 is Actually a Loss in Performance, Since the Original Performance is Multiplied by the SpeedUp Number.

INPUT SIZE	COMPUTE BLOCKS	AREA REDUCED (FACTOR OF)	SPEEDUP	GAIN PRODUCT
32-bit	4	6.69	0.54	3.61
	9	3.22	0.97	3.14
64-bit	9	6.36	0.70	4.45
	16	3.79	0.98	3.71
128-bit	16	7.24	0.72	5.24
	25	4.90	0.96	4.70
256-bit	36	6.65	0.92	6.12
	49	5.07	0.98	4.96
512-bit	64	7.42	0.92	6.80
	81	6.06	0.98	5.94
1024-bit	100	9.14	0.80	2.19
	121	7.81	0.97	2.65

denser memory for various adder sizes. Note that performance is, as expected impacted for the [[7, 1, 3]] error-correcting code as we exploit the limited parallelism in the negatively. To limit the performance degradation we have addressed the parallelism available within the application itself and determined the number of compute blocks to maximally exploit this parallelism with changes in the problem size N (i.e., factoring an N-bit number). For a fixed problem size, utilization of each compute block decreases with an increase in the number of compute blocks as shown in Fig. 5.9(b). Clearly, the decrease in utilization is offset by the increase in overall performance. Thus the challenge in this case is to find the *balance* between utilization and performance.

9.3.1 The Gain Product: Architecture Performanc Metric

When the overall performance of the new specialized architecture is compared to the homogeneous QLA architecture [27], we see that area is reduced by a factor of 9 when factoring a 1024-bit number using only 100 compute blocks. The performance reduction is almost 20%, where the underlying error-correcting code remains the [[7, 1, 3]] code. Since one of the most feasible large-scale ion-trap schemes requires the electrodes to be etched into a Silicon substrate [25, 61], we place equal importance on reducing the area requirements of the architecture as improving the time performance of the application execution.

A good metric for comparing the merit of our design choices taken that affect both area and computational speed is the *gain product* (GP) [28] defined as

$$GP = \frac{(Area_{old} \times ExecutionTime_{old})}{(Area_{new} \times ExecutionTime_{new})}, \tag{9.4}$$

where *ExecutionTime* is the execution time per application procedure. In the case of Table 9.3, *ExecutionTime* is defined as the average time per adder for modular exponentiation. The GP indicates the improvement in system parameters relative to a well defined base architecture which is assumed to have a GP value of 1. The higher the gain product, the better the collective improvement in area and time of the system. As can be seen in Table 9.3, when factoring a 1024-bit number the GP is nearly 2, which is a Gain Product improvement of nearly 100%.

9.3.2 Communication Issues: Executing the Toffoli Gate

The communication constraints in quantum adders, which are the building blocks of the modular exponentiation component of Shor's algorithm, deserve some attention. The most intensive data communication pattern forms during the execution of the Toffoli gate. As described in Section 2.2.1, Toffoli gates cannot be directly implemented on encoded data and must be broken down into multiple one and two-qubit gates. The one-qubit gates include the T gate, which requires

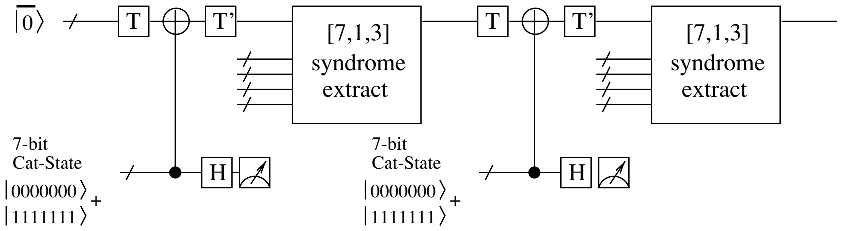


FIGURE 9.10: The circuit for the creation of a single $A_{\pi/8}$ ancillary state. Not shown is the preparation and verification process for the 7-bit Cat States.

the specially prepared encoded $A_{\pi/8}$ ancillary state to execute the T gate implementation shown in Fig. 5.5 in Section 5.3. The circuit used for preparation of the $A_{\pi/8}$ ancilla state is described in detail in [145] and is given in Fig. 9.10, where we see that the preparation requires two error-correction steps and measurement of an additional two seven-qubit cat-states. Separate *specialized ancilla* tiles must be designated to ensure a fresh supply of prepared $A_{\pi/8}$ ancilla states is available when needed. The execution of a T gate requires both accumulators in a single computational tile, where one accumulator is occupied by the data qubit and the other by the $A_{\pi/8}$ ancilla.

The flow of data between the three qubits to complete a single Toffoli forms the most intense communication pattern during the entire addition operation. To study the bandwidth requirements during the Toffoli gates, a scheduler is created that would have all the requirements for communication (creating EPR pairs, transporting EPR repairs, and purifying them) in place while the logical qubit to be transported is undergoing error correction after completion of the previous gate [28]. As it turns out, with the bandwidth of a single channel as shown in Fig. 9.5, it may be possible to completely overlap communication and computation when using the Steane [[7, 1, 3]] code.

Superblocks: To avoid the mismatch between the long error correction cycle of logical qubits stored in memory and logical qubits stored in the compute blocks, we execute highly localized routines such as the Toffoli gate implementation in specially defined superblock regions. The notion of a *superblock* is defined to mean a collection of one or more closely grouped computational tiles. The computational tiles are grouped together to exploit the principle of *locality* inherent in quantum applications, which (much like the classical definition) means that data is most likely to be reused soon after each usage. Larger superblock regions have the advantage of an increased perimeter bandwidth between the compute and memory regions of the specialized architecture. This increase in bandwidth of a larger superblock is offset by the much greater increase in communication required by having to move data from the computing region to the memory region. Intuition suggests that at a certain point, it may be more efficient to

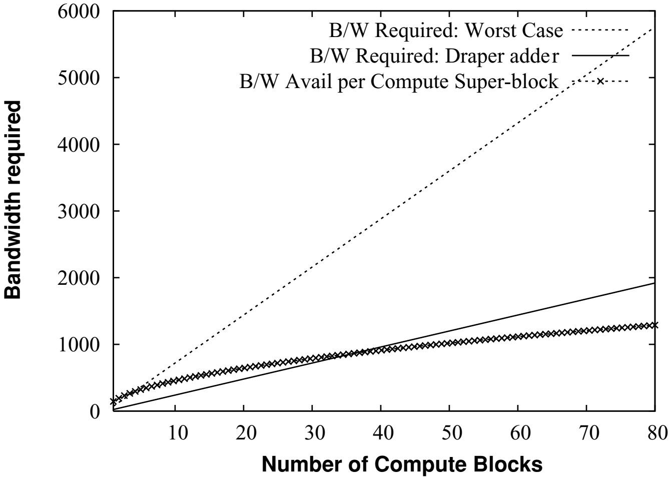


FIGURE 9.11: The point of intersection of the two bottom curves is the optimal size of a compute superblock. These two curves are bandwidth required (at the perimeter of the compute superblock) in modular exponentiation and bandwidth available. The third steep curve is the worst case bandwidth required.

have multiple small superblocks instead of one large superblock. The authors in [28] explore this notion and determine this number concretely. Plotted is the change in bandwidth required against change in bandwidth available as the number of compute blocks increases in Fig. 9.11. The cross-over point is 36 compute blocks per superblock, immaterial of what error-correction code is used. Thereafter it is no longer beneficial to increase the size of an individual compute superblock.

9.3.3 Memory Hierarchy in the QLA Architecture

At this stage we have not yet discussed the notion of cache introduced in Section 8.2. In fact, the specialized QLA discussed in the previous section does not even have a memory hierarchy to discuss, since both the computational tiles and the memory tiles were constructed using the same error-correcting code at the same level of recursion. We saw how the mere separation between memory and computation when decoherence of idle ion qubits is a second-order event, can dramatically reduce the area of the quantum processor, but we also witnessed some performance degradation. Consider, for example, that we do not use level 2 encoding in the computational tiles, but rather remain at level 1 for the Steane $[[7, 1, 3]]$ code. This introduces two substantial concerns:

- The computational tiles are exponentially faster than memory, and thus a single memory cycle introduces significant overhead during the application execution;

- Increasing the number of level 1 computational tiles to reduce processor-memory communication, is tricky because the doubly exponential loss of logical gate reliability at level 1 compared to level 2 is prohibitive for the execution for large-scale applications.

The benefits, however, are clear, logical gates become exponentially faster at level 1 than they are at level 2, and area reduction from using 7 instead of 49 ion qubits per logical data qubit is significant. To make the scheme work, the cache is introduced to utilize the familiar principle of locality and serve the purpose of a buffer between the encoding in the processing elements and the encoding in memory. The data that resides in the cache is placed there either from the processing elements, or has been teleported there through the transfer network shown in Fig. 8.2. Recall that the transfer network is a tile-based computational structure that implements the process of code teleportation to prevent decoding the data between its transfer from one encoding to another. Once again, the cache is at level 1, and we must account for the doubly exponential loss in reliability of the logical data stored there.

To see why the loss in reliability cannot be ignored: recall that the failure per component for the entire system of size $S = KQ$ must be at most $1/KQ$ where K is defined as the number of logical timesteps in the application and Q is the number of logical qubits. Suppose that all operations required by a given quantum application and performed by the QLA architecture are divided between level 1 and level 2 operations. An extremely important observation here is that error-correction cycles on a logical data residing in the main memory are considered a logical operation on the data with an associated level of reliability. Even error correction is performed using noisy lower-level gates and can introduce errors on the data. The fact that it must be implemented fault-tolerantly, is to ensure that errors do not spread to more lower level qubits than the $[\![7, 1, 3]\!]$ code can correct.

The QLA architecture now consists of a memory at level 2, a compute region also at level 2, a cache and a compute region at level 1 and transfer networks for changing the qubit encoding levels. A revised estimate of the required failure per component is needed to account for the loss of reliability due to the level 1 encoding.

An intuitive interpretation of the KQ system size parameter is that it is the area of a 2-D rectangle, where one side is the number of logical qubits and the other side is the length of the computation as a function of the number of time steps K. The area of the rectangle can be divided into several regions: (1) the region of operations that take place at level 1; (2) the region of operations that take place at level 2; (3) the region where qubits are "dead," whose states have not yet been initialized; and (4) the transfer regions between levels 1 and 2. The transfer region is actually divided into logical operations between levels 1 nd 2, so there is no need to distinguish between operations performed during the transfer and operations performed in the computational region. In addition, the third region of "dead" qubits is insignificant for the overall system KQ value while executing Shor's algorithm, because after only the first few time steps all qubits have been utilized in the evenly distributed adders. Given the KQ rectangle for different regions of encoding levels, the modified desired crash failure probability per component is equal to:

$$\epsilon_{\text{fail}} = \frac{1}{f_{L1} K Q_{L1} + f_{L2} K Q_{L2}}, \tag{9.5}$$

where f_{L1} is the fraction of the time spent computing at level 1 and f_{L2} is the fraction of the time spent computing at level 2. The total *KQ* parameters for pure systems at level 1 and level 2 are denoted by KQ_{L1} and KQ_{L2} respectively. The crash failure rate at level 2 is doubly exponentially smaller than the one at level 1, thus we cannot divide the total operations evenly between the two regions. It is also incorrect to assume that the longer the data stays at level 2, the more reliable it becomes, and thus, the more operations we can have at level 1 before they fail. The moment the encoding of a logical qubit is teleported to a different level of error correction, the first error-correction cycle or logical operation must ensure that the qubit does not accumulate more errors than the new level of encoding can handle. The reliability of the encoded qubit immediately becomes controlled by the new encoding, and the time we can compute or store the qubit at that encoding state is controlled by the *KQ* parameter of the subprocedure executed with the data in question.

TABLE 9.4: This Table Shows the Results of Incorporating a Memory Hierarchy and Two Separate Encoding Levels. Depending on the Number of Parallel Transfers Possible Between Memory and Cache, we Can Expect Different Speedup Values for the Adder at Level 1. This Combined with Results From Table 9.3 Give us the Final Gain Product. Comparatively, the Homogeneous Architecture has a Gain Product Number of 1.0. (Note: All Numbers have been Rounded to First Significant Digit After the Decimal.)

PAR XFER	ADDER SIZE	L1 SPEEDUP	L2 SPEEDUP	ADDER SPEEDUP	AREA REDUCED	GAIN PRODUCT
			STEANE [[7, 1, 3]] CODE			
10	256	17.4	1.0	6.2	5.1	31.7
	512	17.4	1.0	6.3	6.1	38.4
	1024	18.2	0.9	5.0	9.1	45.1
5	256	10.4	1.0	4.1	5.1	25.0
	512	10.4	1.0	4.0	6.1	24.5
	1024	11.0	0.9	2.9	9.1	26.9

In fact, the doubly exponential decrease in reliability means that to sustain scalability for Shor's algorithm in the QLA architecture, we can perform just 1% of the total operations at level 1 and the rest must be performed at level 2. One can imagine this to mean that we can only spend 1% of the total logical cycles over all qubits at level 1 recursion, including error-correction cycles. Since quantum modular exponentiation is performed by repeated quantum additions, we find that to comfortably maintain the fidelity of the system, we can perform one level 1 addition for every two level 2 additions. All the operations performed at level 1 this way constitute only 1% of the total system *KQ* parameter, should we have performed everything at level 2. The resulting increase in performance measured as the Gain Product is shown in Table 9.4. Over the entire system KQ rectangle the additions performed at level 1 have constituted less than 1% of the entire computation.

The only communication between the QLA architecture and the classical control processors is the results of measurement and commands for executing classically scheduled quantum instructions. All communication patterns and instruction order execution is orchestrated through software tools that run in the classical processors. To study the behavior of a specialized architecture into software-managed caches, or scratchpad memory Thaker et al. [28]

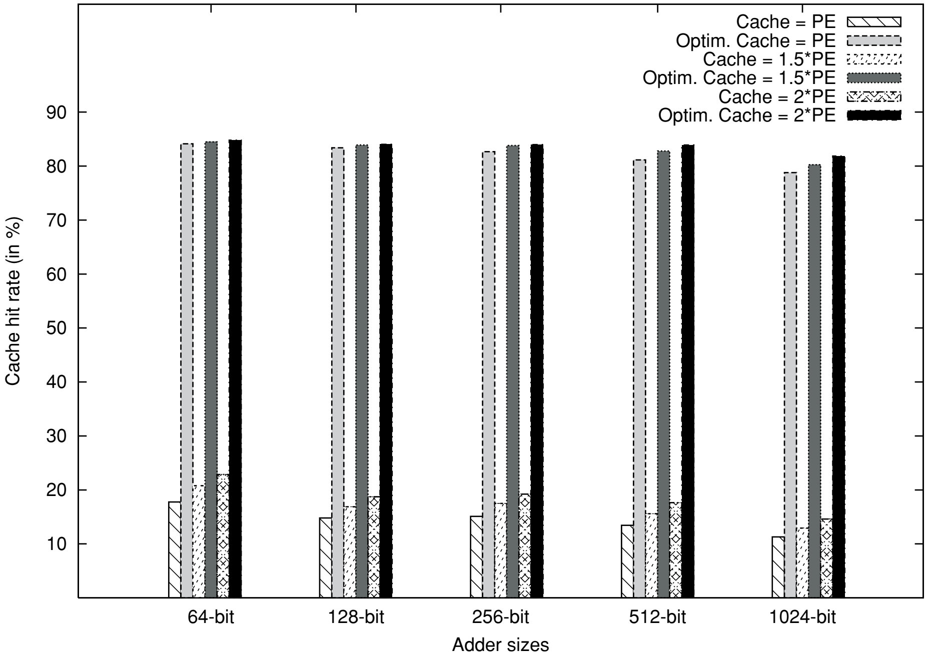


FIGURE 9.12: Shows the cache hitrate for different adders when both cache and compute region are at level 1 recursion. Largest cache considered holds twice the number of logical qubits as the compute block. Results for both the nonoptimized version and the optimized version are shown.

have created a simulator that models a cache as described in this section. The simulator takes into account the computation cost in both encoding levels and also the cost of transferring logical qubits between encoding levels. The application under consideration is still the Draper carry-lookahead adder [154]. Input to the simulator is a sequence of instructions where each instruction is similar to assembly language for quantum computation and describes a logical gate between a number of qubits.

When the simulator runs this code in the sequence intended by the Draper carry-lookahead adder, the cache hit-rate is limited to 20%. To improve the hit-rate, the authors in [28] utilize the following optimized approach. Since the scheduling is static (i.e., run-time instruction scheduling is not assumed at this stage of development), the instruction fetch window for the simulator can be the entire program being executed. The simulator takes advantage of this by first creating a dependency list of all input instructions. Then it carefully selects the next instruction such that probability of finding all required operands in the cache is maximized. This *optimized fetch* yields a cache hit-rate of almost 85% immaterial of adder size and cache size. The replacement policy in the cache is *least-recently-used*. Fig. 9.12 shows the cache hit-rates for different sized adders for the nonoptimized and optimized instruction fetch approaches. If n is the number of logical qubits in the compute region, the cache sizes studied are n, $1.5n$, and $2n$. As the graph shows, the increase in hit-rate is more pronounced due to the optimized fetch than due to increasing cache size. A sensible choice for the QLA architecture is to employ a cach cache size of twice the number of qubits in the compute region. The high hit-rate means the complex transfer network of Fig. 8.2 will not be overwhelmed.

CHAPTER 10

Programming the Quantum Architecture

We begin this Chapter with a description of instruction set architecture (ISA), which captures the interface between a quantum compiler and the architecture. At the application level we have logical instructions acting on logical qubits where measurement operations give the control processors knowledge about the algorithm execution. Below the application level is the physical layout where basic logical gates are decomposed into a fault-tolerant sequence of elementary, technology-dependent, assembly-like instructions. At both levels of execution, the instruction-set environment should provide easy separation of quantum computations from classical data interpreted by the classical control processors. Our discussion is focussed on the description of the high-level instruction set, which is independent of physical implementation technology and allows the compiler to orchestrate the architectural resources available.

The machine instructions we describe operate on both quantum data (logical qubits) and classical data (such as logical qubit addresses, measurement results, and classical control bits). All classical data is stored and manipulated by the classical control processors. The only access the classical processors have with the quantum hardware is through the execution of measurement instructions, which contain both classical and quantum arguments. If an instruction argument is a logical qubit, an address is not explicitly provided. This is because each qubit is a physical entity, and quantum data cannot be cloned, so the control processors are able to keep track of the qubit locations.

A summary of some of the suggested instruction types is shown in Table 10.1. Most instructions at the architecture level of a quantum computer can be classified as *procedure call* instructions. For example, a basic quantum gate instruction could be "`gate_cnot Q1,Q2`," where Q1 is the control logical qubit and Q2 is the target is implemented in the hardware using a fault-tolerant sequence of operations on lower level qubits. It is therefore a self-contained computer program in itself that is incorporated into the larger application and is separately optimized. The control processors are instructed to execute the entire procedure of lower level operations necessary for the completion of a CNOT gate. Error-correction procedures are also

TABLE 10.1: The Set of Possible Instructions That the High-level Compiler Can Use to Represent the Progression of a Quantum Program on Our Architecture Model. FETCH/SEND and LOAD/STORE May Seem Redundant, but They are Not. LOAD/STORE is Only Intended for Communication Between the Cache and Memory.

INSTRUCTION	ARGUMENTS	TYPE	FUNCTION
GATE_A	Q1, [Q2]	Quantum	Single or two-qubit operations
MEASURE	Q1, cbit	Quantum	Measurement, result stored in the classical cbit
PREPARE	Q1	Quantum	Prepare an initial encoded logical qubit state
FETCH	Q1, PE_i, AC_j	Classical	Fetch qubit Q1 into PE_i and accumulator j
SEND	Q1, memory_address	Classical	Send Q1 from a PE into memory (cache or main)
LOAD/STORE	Q1, memory_address	Classical	Load/store Q1 to the specified address
REFRESH	Q1	Classical	Error correct qubit Q1

implemented as a single instruction with a logical qubit as an argument. Branching instructions serve the same purpose as classical branches, though the decision of whether to branch or not is always dictated by a classical bit set by the result of a quantum measurement.

10.1 PHYSICAL INSTRUCTION SCHEDULER

A similar ISA is described in practice by the low-level quantum assembly language (QASM) first proposed and implemented by Balensiefer et al. [142, 141]. QASM consists of a sequence of declarations and commands for physical qubits similar to the logical procedures shown in Table 10.1. Qubits, classical bits, gate names, and classical functions are initially declared. The preparation procedure is classified with the two physical gates XPREPARE and ZPREPARE, which place a qubit in either eigenstate of the X or Z operator, respectively, and can be decomposed into a measurement operation followed by a corresponding single-qubit gate. The ZPREPARE operation is implemented by applying a measurement gate followed by a bit-flip gate if the measurement result yields a “1.” The qubit is this way initialized to the $|0\rangle$ state. The XPREPARE gate places the qubit in either the $|+\rangle$ or $|-\rangle$ state by applying a Hadamard operation on a ZPREPARE’d qubit.

Irrespective of the underlying technology used to implement a circuit-model-based quantum system, one of the central challenges of accurately modeling the physical components of a large-scale architecture is the ability to map and schedule a quantum application onto a physical layout by taking into account the cost of communication, the classical resources, and the maximum parallelism that can be exploited. Error correction is by far the most dominant application and the driving application for optimizing quantum architectures; however, what is missing to being able to simulate the fault tolerance and functionality of error-correction networks, is the automatic generation of communication commands and ILP representation of an arbitrary quantum circuit—a need that a physical scheduler based on traditional classical scheduling techniques may meet. Just as in classical superblock schedulers, the output of a quantum physical operations scheduler can also be a QASM file, but one that is fully parallelized and communication instructions have been inserted.

In addition to generating two-dimensional information about the communication paths for a given quantum circuit, a physical operations scheduler allows us to determine the exploitable instruction-level parallelism (ILP) in quantum circuits. Studying the limits of ILP can be used by hardware designers to avoid spending resources on classical control features that will remain unutilized throughout the computation. Furthermore, massive ILP is an underlying requirement for achieving the best possible schedule in quantum error correction [116]. Even though it has been shown that a threshold value exists when movement is considered [118, 149], the ability to *precisely* predict the amount of communication during error correction is crucial for determining how high the threshold value really can be. In addition, knowledge of the communication requirements and available ILP will provide us with better understanding of the exact hardware resources needed for error correction.

The QUALE tool-chain from the University of Washington [142] uses the classical Path-Finder package [156] to map the instruction of a quantum circuit onto a physical layout, provided that it is known ahead of time which qubits are supposed to move. Alternatively, QPOS is a quantum physical operations scheduler based on traditional classical instruction scheduling heuristics [157–159] through careful priority calculation at both the circuit level and the physical layout level that does not place any physical constraints on the qubits. QPOS is described in detail in [160]. At the circuit level, instruction priorities are based on the number of instructions that depend on each operation. The priorities are used to choose the desired communication paths after the source qubits and the destination qubits have been disambiguated. If instructions have the same priorities the paths are prioritized based on least path interference and shortest path. This amounts to maximally parallelize the movement operations. In our case study in Chapter 9 we employ QPOS to schedule the fault-tolerant qubit tiles of the example architecture provided.

While recent breakthroughs in error-correction algorithms [146, 147] combined with clever large-scale quantum architecture design [135, 28] allow us to be optimistic about the realization of applications such as Shor's quantum factoring algorithm [7], the precise orchestration of millions of interacting physical qubits at the cycle level, will undoubtedly prove to be necessary for realistic implementations. Clever cycle-level schedulers that build on existing knowledge of efficient classical scheduling algorithms to provide us with a starting base for developing sophisticated scheduling techniques tailored for quantum circuits.

10.2 HIGH-LEVEL COMPILER DESIGN

We can identify several levels of reordering rules that a quantum compiler may employ at any stage of the compilation. On one level are network optimizations independent of the underlying architecture (i.e., communication cost is not considered and each gate has a unit cost). The next level of algorithm optimization/compilation is the *coupling* of computation and communication, where a given high-level network is mapped to a specific set of logical hardware resources. A third level of optimization is when the specific logical gate implementation is parameterized and considered not only in the architectural resources, but also in the high-level network synthesis. For a fixed set of universal gates, individual gate costs and size vary wildly depending on the error-correction procedure. In addition, the teleportation-based communication mechanism at the high level may be used by the compiler to allow the execution of single-qubit gates during the logical qubit movement [125], provided sufficient communication channels' bandwidth.

Fig. 10.1 is an example of the elements for a possible quantum architecture compiler. A description of the program in a very general high-level manner such as a large unitary matrix or a high-level C-like language such as QCL [137] that is technology and layout independent

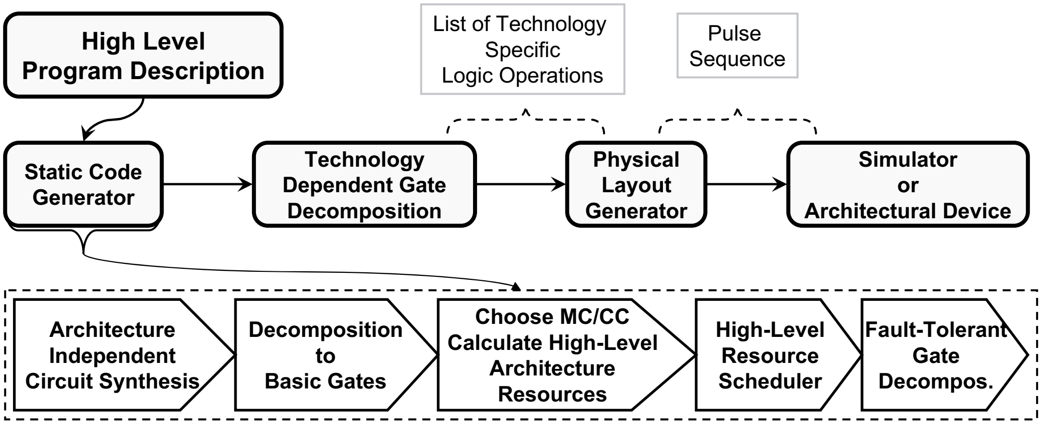

High Level Program Description
List of Technology Specific Logic Operations
Pulse Sequence
Static Code Generator
Technology Dependent Gate Decomposition
Physical Layout Generator
Simulator or Architectural Device
Architecture Independent Circuit Synthesis
Decomposition to Basic Gates
Choose MC/CC Calculate High-Level Architecture Resources
High-Level Resource Scheduler
Fault-Tolerant Gate Decompos.

FIGURE 10.1: Compilation layers.

serves as an input to the *static code generator* (SCG). The bottom of Fig. 10.1 shows the steps of the SCG component.

The first SCG stage is to perform technology- and architecture-independent circuit synthesis that breaks the algorithm into a useful identifiable set of quantum operations. Once a universal set of basic gates is determined the SCG further decomposes the network into those gates exposing all high-level qubit resources needed for the application. Next, the SCG determines useful error-correcting codes for the memory and the computation together with the architectural resource constraints and high-level structures. At this point the program is composed of assembly language-like instructions as in Table 8.1 fully exposing the hardware resources and ready to be scheduled by the high-level scheduler. Finally, the SCG implements fault tolerance into each operation by decomposing each logical gate and LOAD/STORE operation into a fault-tolerant list of lower level quantum/classical instruction based on the choices for MC and CC.

The output of the SCG is a high-level quantum assembly language that will describe a fault-tolerant, error-correction enabled quantum algorithm with a clear description of the available quantum and classical resources at the system level. The next stage is the technology dependent compiler (TDC), which knows nothing about the geometrical layout of individual tiles, but decomposes the quantum operations into the equivalent elementary operations available for the particular technology. In the case of ion traps the output is a giant list of ion-trap logic gates consisting of single qubit rotations and controlled-Z gates.

Last is the physical layout generator (LG), which takes in the available resources and allocates physical locations and data paths for each physical qubit in the system. The LG has full knowledge of the physical layout of each tile and schedules the elementary qubit operations accordingly, even if this changes the original sequence provided by the SCG. Assuming that maximum parallelism has been implemented at previous stages the LG attempts to (1) minimize the communication costs for multiqubit gates at the physical level and (2) optimize the resource distribution and minimize the cost due to resource constrains on the achieved parallelism already in the network description for each logical gate. The output of the LG is a sequence of fine grained control pulses fed into the physical device.

10.3 ARCHITECTURE-INDEPENDENT CIRCUIT SYNTHESIS

Architecture-independent circuit synthesis is analogous to the design and optimization of classical integrated networks, where technology-independent synthesis is performed using abstract logic gates. After this, the network is mapped to the technology by converting the gates to the gates best suited to the specified technology as described in Section 10.2. At any stage of the compilation flow (except perhaps the layout generator), given a general logical network, a compiler may identify various subnetwork structures which lend themselves to different optimization techniques.

At the highest possible level of a quantum algorithm, the action of the algorithm on n logical qubits is described as a $2^n \times 2^n$ unitary matrix. This is analogous to a given boolean function in classical computation. Just as it is relatively easy to translate a boolean function into a corresponding network of operations, it is similarly possible to decompose an arbitrary n-qubit unitary operator into basic quantum logic gates. Also, exactly as in classical computation, the optimization of the resulting network is a nontrivial task. What may be truly different are the transformation rules for a quantum network.

There is a significant amount of ongoing work in architecture-independent quantum network synthesis. The first subnetworks that may be identified by a compiler are networks composed entirely of controlled-NOT gates. Considerable work has been done for the synthesis of controlled-NOT networks and general classical reversible networks [161, 162]. Provided are local transformation rules for arbitrary controlled^k-NOT networks (interconnected NOT gates controlled by the AND of k bits). The transformation rules take any controlled^k-NOT network to an equivalent network in its canonical form, which can then be optimized using a heuristic whose cost is to minimize the average number of control qubits. Quantum modular exponentiation—the most expensive component of Shor's factoring algorithm—can be written entirely as a controlled^k-NOT network. Additionally, a technique for restructuring stabilizer networks which are used in every error-correction routine is given. Aaronson and Gottesman [143] prove that any stabilizer network has an equivalent network in canonical form with only $O(n^2/\log n)$ gates, leaving open the question whether an optimal construction exists. Much research has been done with arbitrary two-qubit operators. It is desirable to decompose any two-qubit operator into a number of controlled-NOT gates (i.e., CNOT) since the universal gate library [33] consists of CNOT gates and one-qubit gates. Song and Klappenecker [163] contribute a method for optimizing arbitrary controlled two-qubit operators, Shende, Bullock, and Markov [164–166] propose tests and implement an algorithm that gives quantum networks that simulate arbitrary two-qubit unitary operators. More specifically, they provide a method to determine which two-qubit operators are CNOT optimal, with the worst case being three CNOT gates. Moreover, Shende and Markov [167] have studied the problem of finding optimal networks implementing incompletely specified two-qubit operators usually used for state preparation when the input is known or after measurement operations. Other network synthesis work has been with arbitrary n-qubit diagonal operators [168, 169]. All the above-mentioned network optimizations are implemented during the first of the quantum compiler: the static code generator before the architectural resources are considered.

Looking ahead: Although extensive groundwork has been done in architecture-independent circuit synthesis, a carefully designed quantum compiler can provide a framework from which to unify, refine, and expand these optimizations. In particular:

- A set of concrete transformation rules for stabilizer networks can be given to provide a better canonical form than the one in [143], which can be used to create heuristics that optimize error-correction networks.
- Creation of fault-tolerance preserving transformation rules that allows us to change the universal set of basic gates while maintaining maximum time to failure in a high-level algorithm.
- The combination of all different transformation rules has never been explored before. For example, it is unknown how the rules affect each other once they are implemented in common optimization tool.
- Finally, there is much promise in the exploration of incompletely specified n-qubit operators when decomposing high-level quantum algorithms.

10.4 MAPPING CIRCUITS TO ARCHITECTURE

While circuit synthesis is an important step, mapping these idealized circuits to a physical machine is perhaps the greatest opportunity for optimization. A custom hardware implementation of each circuit is not only impractical due to machine size constraints, but also technology-dependent elementary operations, large fault-tolerance overheads, and the use of teleportation all make the classical approach of direct circuit synthesis to hardware unappealing in the quantum domain. Creating schedulers that map quantum circuits onto equivalent fault-tolerant procedures that utilize the tradeoffs associated with the quantum hardware and possibilities at the systems level will be one of the key contributions of computer architects.

Let us consider an example which describes part of the process of mapping and optimizing a controlled^k-NOT network. This is illustrated in Fig. 10.2, which shows three equivalent controlled^k-NOT-based networks. It is clearer to describe the network schematically rather than showing the instructions explicitly as in Table 10.1. When shown schematically one can "see" the needed communication from qubit to qubit when executing multiqubit gates such as controlled operations.

The network in Fig. 10.2(a) is the derived canonical form using the transformation rules provided in [161] of an initial unoptimized CNOT-based network. The canonical form is a useful starting point for the optimization of any boolean CNOT-based network since it allows all NOT operations to be concentrated on the last qubit ($q7$). This means that the gates can be

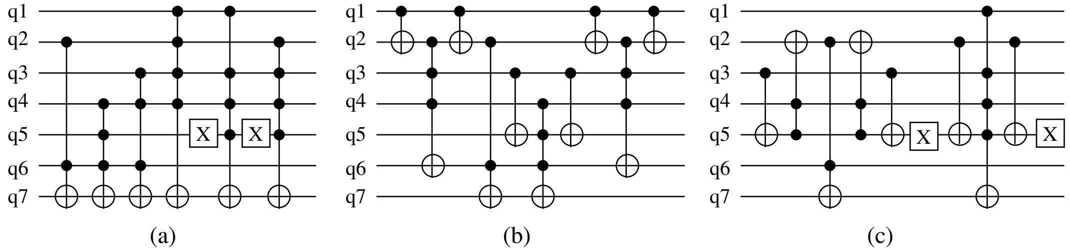


FIGURE 10.2: Optimizing networks for fewer control bits. The circles are NOT gates controlled by the AND of the qubits connected by solid dots on the vertical lines. The boxes marked with an "X" are bit-flip gates. We want to minimize the number of solid dots per gate without dramatically increasing the number of gates. (a) An unoptimized controlled-NOT-based network in its canonical form [161]. (b) Using the transformation rules in [161] to reduce the number of control qubit per gate (1.7 per gate). (c) Using boolean algebra simplification of the network in (a) to reach 1.4 control qubits per gate.

executed in any order and it would be straightforward to apply boolean algebra simplification. A good CNOT-based network cost metric is the minimization of the number of control qubits per gate, which is what the authors of [161] strive for. Their result is shown in Fig. 10.2(b). This is a reasonable assumption since any controlled^k-NOT gate with $k > 2$ must be divided into $(2 * \lfloor \log k \rfloor + 1)$ *Toffoli* gates (a three-input, three-output reversible NAND gate; implemented as a NOT with two control bits) using $\lfloor \log k \rfloor$ additional logical ancilla qubits adding to the overall resources needed from the architecture. In addition, the synthesis of each Toffoli gate into one- and two-qubit operations is relatively expensive: a Toffoli gate divides into two Hadamard gates, one S gate, six CNOT gates, and seven T gates [38] as in Fig. 2.5 gates are essentially two-qubit gates since they require interaction with the specialized ancilla qubits and need both accumulators in a PE. An even better network in terms of the control-qubit cost is Fig. 10.2(c) which we derived using simplification rules derived from boolean algebra (i.e., $A \oplus BA = A\overline{B}$).

Calculating the CNOT *circuit cost:* Ideally, quantum researchers would like to create a compiler that can recognize the optimal network structure of Fig. 10.2 for the specified architectural constraints. The compiler will choose the MC and CC encodings, decompose the basic network gates into fault-tolerant procedures, each individually optimized over the architecture such that the cost of communication, computation, and classical resource overhead throughout the high-level network execution is minimized. In a network of one- and two-qubit gates, the most expensive operation is a transfer of a logical qubit from the MC encoded memory to the CC encoded cache or PE. The *memory read* time (MR_t) roughly estimated from Fig. 8.7 is equal to two MC teleportation steps, five MC error corrections, five logical gate times over MC whose sequence diagram is shown in Fig. 8.4, and one error correction step over CC. A cache miss is equal to MR_t; however, a cache hit is just a single teleportation step from the cache to the

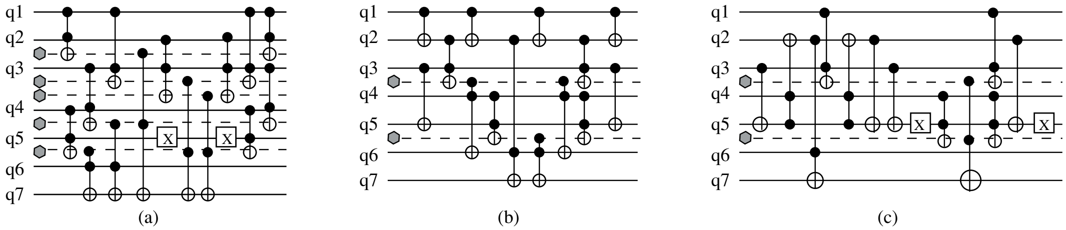


FIGURE 10.3: Equivalent networks to the ones in Fig. 10.2, but with all gates decomposed into Toffoli gates and full exposure of the logical qubit resources required. The dotted horizontal lines represent ancilla qubits, which are added to reduce overall communication costs once mapped to the architecture. Network (a) is least desirable, using the most logical qubits. Network (c) is the most desirable as it decomposes into fewer elementary operations.

processing element over the CC encoding. Thus, the cache read time (CR_t) is therefore

$$CR_t = X(t_{\text{tel,CC}}) + Y(MR_t),$$

which is a weighted average of the expected cache-hit rate versus the expected cache-miss rate. The CC is chosen such that the time for a logical operation after the retrieval of the data should be much less than the time for an operation over an MC encoded qubit, where the tradeoff is that the logical qubits stored over MC are much more reliable.

Fig. 10.3 shows the networks from Fig. 10.2 with all controlled$^{k>2}$-NOT gates broken into CNOT and Toffoli's only using a number of auxiliary logical qubits initialized at the "0" encoded state (dashed lines). The gates are reordered to expose the available parallelism at this level; however, each Toffoli gate is not yet decomposed into one- and two-qubit gates as shown in Fig. 2.5. Each gate is a fault-tolerant logical gate composed of a number of physical operations over the PE tile whose physical network depth and dimensions are determined by the choice of CC. One can imagine the magnitude of the computation even for such a small network. Without explicitly calculating the schedule for each network over basic single and two-qubit operations, we see that the network in Fig. 10.3(a) uses three more logical qubits than both other networks. In addition it requires 10 Toffoli gate time steps over 15 total Toffoli gates. Figs. 10.3(b) and 10.3(c) require only 7 and 6 Toffoli time steps and 8 and 7 total Toffoli gates respectively. The number of Toffoli time steps limits the network's overall performance making the network in Fig. 10.3(a) least desirable, even with infinite resources and parallelism. This, however, does not provide us with much information about the network's communication costs without more careful consideration. To avoid any cache misses a Toffoli gate will require two PE units and two logical qubit cache tiles. The time for a Toffoli gate will be roughly equal to the time of 14 logical operations over CC, plus the starting cost of loading the three data qubits from memory. Note that in addition to being very time consuming, loading a qubit from memory exposes it to greater risks of failure over the network. We must also consider that in reality, classical control resources are equally as expensive as quantum ones in addition to a need to control the overall area explosion due to recursive error correction, thus limiting the available parallelism.

Looking ahead: The controlled^k-NOT circuit example provides only an overview of the complexity involved in implementing just one portion of the compiler design flow shown in Fig. 10.1—the static code generator. We have presented this example without a specific theory of the reordering rules for architecture-dependent quantum networks. The most important first stages of a workable compiler implementation must be to identify the unique elements of quantum computing networks and properties of quantum computation that will allow us to create the corresponding intermediate compiler data structures and representations. Apart from tracing the necessary intermediate steps of a quantum compiler we can identify several important challenges for system designers when designing a compiler for the development of large-scale quantum applications. The first challenge is the development of simulation and modeling techniques for the quantum circuits involved in the implementation of the high-level architectural elements; secondly, we must find suitable cost metrics for compiler optimization that will allow us to generate and evaluate efficient fault-tolerant networks at both the architecture level and the physical level of execution for a given quantum application; thirdly, it is desirable to identify algorithms to insert, preserve, and optimize low-level, fault-tolerant networks that implement high-level computations; finally, it is important to identify architectural strategies that can exploit uniquely quantum computation and communication resources, such as teleportation-based error correction and varying logically universal sets of quantum operations. In the next Chapter we describe the possibility of teleportation-based quantum operations and error correction.

CHAPTER 11
Teleportation-Based Quantum Architectures

It would be ideal to treat the physical implementation of quantum logic gates with a specific technology in mind such as the QLA's treatment of ion traps. The problem is that there is an enormous amount of available choices for physical gate implementation, which is equally matched by an enormous amount of available possibilities for constructing logically universal circuits.

In Chapter 2 we described the circuit model for quantum computation which implements universal quantum logic as a sequence of unitary matrices that act on the probability amplitude vector describing a collection of units of quantum data known as qubits. Furthermore, in Chapter 2 we introduced the Clifford group operations (see Eq. 5.10) combined with the single-qubit T gate as an elementary basis for universal quantum computation. The chosen basis of gates offers relatively straight forward, fault-tolerant constructions for implementing quantum logic on encoded qubits. Steane [170] summarizes several proposals for constructing a fault-tolerant universal set of quantum operations that includes the Clifford group. Some proposals include the three-qubit Toffoli gate as an elementary operation, and some the controlled-S gate [171, 172]. When designing a quantum architecture, or modeling software for quantum architectures, a system designer may need to allow flexibility in the software to choose the appropriate universal set of gates that allows the generalization to all $[[n, k, d]]$ error-correcting codes.

Perhaps, even more interesting is the fact that we may not even need the direct application of gates to perform universal quantum computation. All we need is a circuit structure (or a mechanism) that implements the functionality of universal gates. More specifically, a mechanism that allows the unitary transformation of a quantum state $|\Psi\rangle$ without the physical application of the unitary operation itself. Such a mechanism is *teleportation*. In 1998, Gottesman and Chuang published a paper [125] that showed teleportation as a universal quantum logic primitive that can be used to perform any quantum computation. The teleportation gate scheme is used to allow two-qubit operations in optical quantum computers (see Section 4.1). We can extend this further by looking at the possible tradeoffs when designing an architecture that utilizes universal quantum logic on encoded data through teleportation. Such investigations

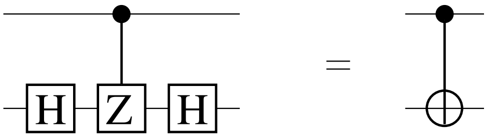


FIGURE 11.1: A CNOT gate can be build by using a controlled-Z gate and two Hadamard gates.

may drastically change the structure of the entire quantum system as defined by the QLA architecture case study.

One of DiVincenzo's principal requirements for quantum technologies is the ability to orchestrate universal quantum logic, which is generally composed of single-qubit gates, two-qubit gates, and measurement in the circuit model of quantum computation. Except for superconducting qubits, most technologies allow a relatively easy arbitrary single-qubit rotations. Therefore, the ability to perform qubit–qubit interactions, or two-qubit gates is the most critical requirement for a given technology, particularly, since an implicit assumption in any qubit–qubit interaction is the ability to communicate quantum information between the two qubits.

Many of the circuit synthesis papers mentioned in Section 10.3 assume the CNOT gate to be the standard elementary two-qubit gate and synthesize circuits to be CNOT optimal. Perhaps incorrectly, a general assumption is that DiVincenzo's criteria demand the ability for a technology to demonstrate a reliable CNOT gate; however, the direct application of a CNOT gate is not necessarily required. The elementary two-qubit gate in the ion-trap technology, for example, is the controlled-Z rotation [77, 78], which can be used to functionally construct a CNOT gate as shown in Fig. 11.1.

Any two-qubit gate used to implement a CNOT operation requires the interaction of two qubits, either through *direct* qubit–qubit interaction, which implies that the quantum data for both qubits is placed at the same spatial location through teleportation, or through direct physical movement of the qubit carriers; or *indirect* qubit–qubit interaction, which is done through some shared medium that allows the two-qubit states to be coupled without the need to bring the data spatially close together.

Both types of interactions have their respective drawbacks. Qubits that interact directly require either information swapping between nearest neighbors, or shuttling qubits through empty channels: introducing errors proportional to the length of the channels. Transferring quantum information creates the need for complex low-level schedulers such as QPOS [160], or the inner workings of QUALE [142], both of which assume technologies that require direct physical qubit communication. In addition, bringing the states of two qubits together creates difficulties for distinguishing the qubits from one another and opportunities for correlated errors.

The indirect qubit–qubit interaction may seem more efficient on the outset by leaving the qubits in place, but it still requires a common medium that is used to couple the qubits and can potentially introduce correlated errors. There are several techniques to achieve this: one technique uses single photons to implement multi qubit gates between trapped atoms [43, 95, 96], another technique couples qubits through a common quantum field mode, which can be thought of as a shared quantum "bus" and can be realized with a laser beam [97, 98].

11.1 THE CNOT GATE AND SINGLE-QUBIT GATES THROUGH TELEPORTATION

The simplest way to consider the implementation of a CNOT gate using teleportation is already utilized by the QLA architecture we described in Chapter 9. A schematic is shown in Fig. 11.2. Qubits Q_1 and Q_2 residing in the memory region are teleported to a processing element (PE) through EPR pairs created between each respective memory address and the PE.

The gates in the dashed boxes in the circuit of Fig. 11.2 implement a *Bell measurement* between any two qubits. Recall the four two-qubit Bell states $\{|\Psi_+\rangle, |\Psi_-\rangle, |\Phi_+\rangle, |\Phi_-\rangle\}$ given in Eq. 9.2, where the state $|\Psi_+\rangle$ is the familiar two-qubit EPR state. Just as a single-qubit state can be written as a superposition of two basis states such as $|+\rangle$ and $|-\rangle$ or $|0\rangle$ and $|1\rangle$, a two-qubit state can be written as a superposition of the four Bell states:

$$|q_1, q_2\rangle = c_0|\Psi_+\rangle + c_1|\Psi_-\rangle + c_2|\Phi_+\rangle + c_3|\Phi_-\rangle. \tag{11.1}$$

A Bell measurement such as the circuit in the dashed boxes of Fig. 11.2 determines which of the four Bell states the two qubits are in. As a result of the measurement the two qubits are collapsed into one of the four Bell states. In addition, the Bell measurement also serves as an entangling operation between the two qubits if they are originally unentangled. It helps to abstract the Bell measurement procedure as a box (see Fig. 11.3) much like we did with the EPR pair creation because each technology has a very specific method for implementing a Bell measurement on two qubits.

In Chapter 9 we described an architecture which required a direct CNOT gate between two logical qubits encoded with 49 physical qubits at level 2 recursion. We presented the

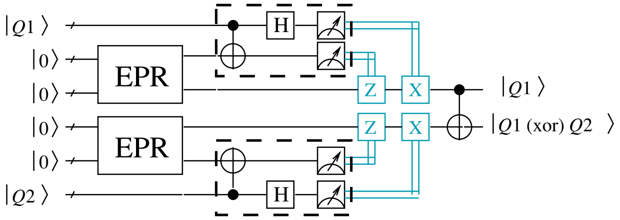


FIGURE 11.2: Standard CNOT operation between two logical qubits in remote locations. The qubits are teleported to a common destination such as two adjacent accumulators in a processing element and interacted with a CNOT gate.

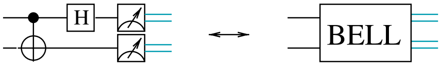


FIGURE 11.3: We abstract the Bell measurement circuit as a box with the inscription "Bell."

long-distance communication channel between the two logical qubits as a repeater-based interconnect which creates 49 purified elementary EPR pairs that span the entire channel between qubits Q_1 and Q_2. The teleportation procedure then is used to teleport each of the 49 physical qubits from Q_1 and Q_2 so that the two logical qubits are directly next to each other in the two destination accumulators. The direct application of a transversal CNOT gate follows the logical qubit transfer once they are located in adjacent accumulators as shown in Fig. 11.2.

To avoid the direct interaction between the two logical qubits shown on the right-hand-side of Fig. 11.2, we can move the CNOT gate through the preceeding single-qubit X and Z operations by changing their order without affecting the functionality of the circuit. The result is shown in Fig. 11.4. In this new (but equivalent) construction, there is no direct interaction between qubits Q_1 and Q_2, but there is a direct CNOT gate between the EPR blocks. The interaction between the EPR blocks is only possible if the four blocks themselves are encoded logical qubits initially in the logical $|\overline{0}\rangle$ states as shown in the Figure 11.4. The creation of the two logical EPR blocks followed by a CNOT gate between the two blocks is enclosed with a dashed line in Figure 11.4, to enforce the notion that this procedure can be done in place without any interaction with the data blocks Q_1 and Q_2. In this manner, the implementation of the CNOT gate decomposes into encoding four qubits initialized to $|\overline{0}\rangle$ into some prespecified four-qubit state, denoted as the state $|\overline{M}\rangle$, where

$$|\overline{M}\rangle = \frac{(|\overline{00}\rangle + |\overline{11}\rangle)|\overline{00}\rangle + (|\overline{01}\rangle + |\overline{10}\rangle)|\overline{11}\rangle}{\sqrt{2}}. \tag{11.2}$$

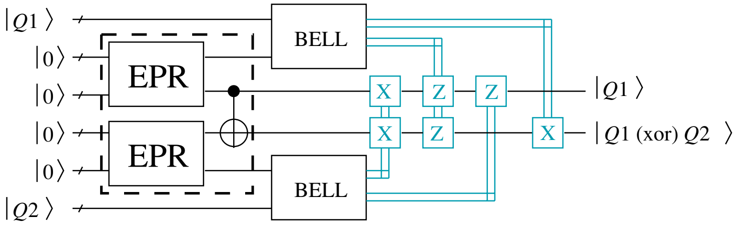


FIGURE 11.4: Teleporting two-qubits through a controlled-NOT gate by using only single-qubit rotations, Bell measurements, and a special four-qubit state $|\overline{M}\rangle$ which can be composed of two EPR pairs, or two GHZ states. If two GHZ states are used, the CNOT gate between the two EPR pairs will be replaced by a Bell measurement between the two GHZ states. The circuit shown and the procedure is given in [125].

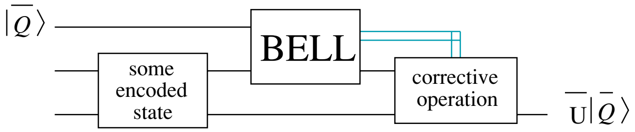


FIGURE 11.5: An implementation of a single-qubit operator U through teleportation. The implementation requires the preparation of an encoded state using two logical ancillary qubits and a Bell measurement followed by the corrective operation.

Moreover, the $|\overline{M}\rangle$ state can be prepared using two three-qubit cat states ($|000\rangle + |111\rangle$), also known as GHZ states [173], by applying a Bell measurement between two of the qubits in each GHZ state followed by single-qubit gates controlled on the result of the Bell measurement [125]. The four qubits not involved in the Bell measurement will retain the $|\overline{M}\rangle$ state and can be used for the implementation of the CNOT gate. This gives us a CNOT gate mechanism that requires only classically controlled single-qubit gates, a specially created four-qubit entangled state, and two Bell basis measurements. Given that the creation of the $|\overline{M}\rangle$ state can be performed offline, and deterministically much like the creation of EPR qubits, then the CNOT gate between two logical qubit may be implemented without any direct qubit–qubit interaction.

Remotely entangling two qubits to form an EPR pair is possible [43, 95, 96]. In addition, it may be possible to remotely entangle three qubits into a GHZ state, or even create a black box mechanism that creates GHZ states of encoded qubits and distributes them in the architecture through a repeater-based channel as used in the QLA model. Even if the black box consists of traditional data shuttling to create qubit–qubit interactions for the encoded special states, it can be localized to a special state "factory" region where the distances are short relative to the application level system.

Gottesman and Chuang [125] further show that the same methodology can be used to construct a teleportation-based mechanism for any encoded single-qubit logical operation. A schematic for implementing an arbitrary single-qubit operator $\overline{U}$ is shown in Fig. 11.5.

The universal gate set we are considering only requires a teleportation implementation for the T gate for any other single-qubit gate is transversal and can be applied locally. The T gate circuit shown in Fig. 5.5 of Section 5.3 is much simpler than the network of Fig. 11.5 and utilizes the concept of *one-bit teleportation* [174]; however, it requires a CNOT gate between the data state and the specially prepared $|A_{\pi/8}\rangle$ ancilla state. To avoid direct qubit–qubit interaction, the required CNOT gate in Fig. 5.5 can be implemented using the teleportation circuit shown in Fig. 11.2 with the resource cost of four additional qubits for the creation of the $|M\rangle$ state.

11.2 THE ARCHITECTURE

We are faced with two gate implementation choices. The first one is to teleport data to a processing region using the repeater-based interconnect, and the second choice is to teleport

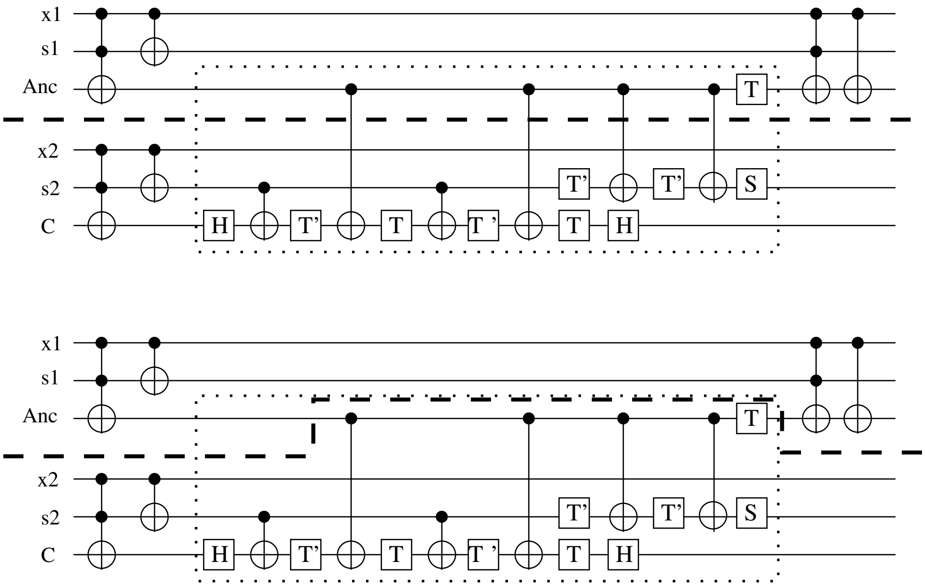


FIGURE 11.6: The model used by in [175] to distinguish between teleporting data and teleporting gates in a distributed quantum computer. The upper half of the figure shows a 2-bit adder of six qubits where the middle Toffoli gate has been expanded into its one- and two-qubit gate decomposition. The thin dashed line separates the two processing regions evenly as it is intended to illustrate that gates are teleported from one region to the other, while the data remains in place. The lower part of the figure shows data teleportation as described by the QLA architecture.

gates through specially created ancillary states. Fig. 11.6 shows the distinction between the two choices of distributing quantum computation in the two-bit adder from Section 2.2.1. The adder is divided into two main processing regions that initially perform computation in parallel through the first two time steps. The third time steps requires a Toffoli gate between the ancilla qubit in the first region and two qubits from the lower (second) region. The Toffoli gate has been decomposed into elementary one- and two-qubit gates in the dashed box of each half of Fig. 11.6. If the two three-qubit regions are significantly far apart, we have a choice to teleport the data as described in Section 9.2, or to teleport the qubits through the gates involved in the decomposition of the Toffoli gate (see Fig. 2.5 in Section 2.2.1).

The trade-offs associated with teleporting gates as we discussed so far have been given in more detail in [175], and teleporting data on a distributed quantum computer, where the schematic distinction shown in Fig. 11.6, is introduced. The authors base their study on several implementations of the adders used for Shor's factoring algorithm and find that, at the large-scale, it is more expensive to teleport gates than it is to teleport data in terms of the number of elementary operations competing for shared resources. The authors of [175] use a clever

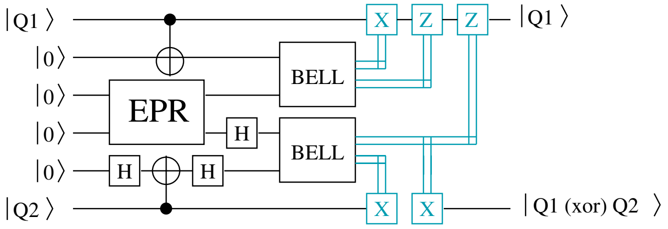


FIGURE 11.7: Teleporting two-qubits through a controlled-NOT gate that requires four ancillary qubits, but only two need to be encoded as an EPR pair. The other two are used to couple with the data before the Bell measurement operation.

construction of the teleported CNOT gate that does not require the four ancillary qubits to be placed in the $|\overline{M}\rangle$ state and leaves the CNOT gate implementation in the encoded data qubits, rather than in the EPR qubits. Their construction is shown in Fig. 11.7.

The underlying architecture is based on solid-state qubits coupled indirectly through a universal quantum bus [97, 98]. Similar distributed architecture can be realized with the ion-trap technology by coupling two ions through photo detector stations and beam splitters [43, 95, 96]. The quantum bus connects the distributed pieces of the application level system, where each piece uses transceiver qubits to connect to the bus. In this manner, data or gates can be transferred between multiple distributed regions by using the transceiver qubits as EPR pairs for teleportation. Two transceiver qubits in different regions can be remotely entangled through the quantum bus.

Intuitively, the observation that teleporting gates are less efficient than teleporting data is reasonable when looking at Fig. 11.6. Once the data is teleported to a specific region, it becomes *local* to that region and the sequence of gates can be executed directly to complete the Toffoli operation without much communication overhead. On the other hand, the teleportation of gates requires repeated usage of the quantum bus and the contention for the transceiver qubits increases [175].

But what about encoded gates on fault-tolerantly constructed logical qubit states, which will undoubtedly be required for large-scale applications? The CNOT gate construction in Fig. 11.7 is not a truly teleported CNOT gate because it requires two local CNOT operations between the data qubits and the ancillary qubits before the Bell measurement procedure. If locally executed CNOT gates are allowed where data is transferred between the control qubit and the target qubit, then we can use much simpler teleportation-based CNOT gate construction given in [174]. Fig. 11.8 shows two versions of a teleported CNOT gate where only two ancillary qubits are required as an encoded EPR pair between the control and target qubits.

An interesting trade-off would be to study the optimal logical distances at which traditional direct interaction CNOT gates are allowed, and larger distances where CNOT gates are

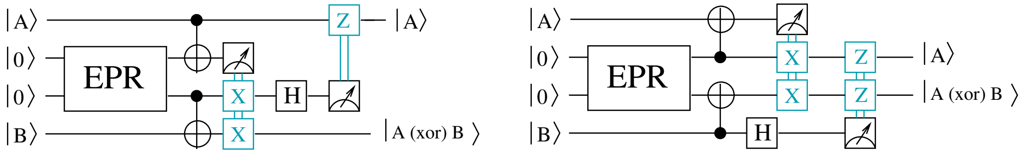


FIGURE 11.8: Two versions of a simplified "remote" CNOT gate.

sent through teleportation procedures through remote data coupling. Our architecture can be a distributed logical architecture, where there are N regions labeled $\{R_1, R_2, \ldots, R_N\}$, which contain both logical data qubits and logical ancillary qubits used for the creation of specialized states for gate teleportation. This has the potential to significantly improve the reliability of the application. Standard direct interaction logical CNOT gates are executed within each region. The logical data never leaves to another region, but inter-region CNOT gates are implemented through the specialized ancillary qubits in each region.

This scheme has the potential to significantly improve the reliability of the architecture, as logical gate distances between regions are relatively short, and inter-region gates are teleported. The specialized states between regions can be prepared independently of the execution of the application and verified for correctness. The coupling of the individual qubits can be done remotely through entangling trapped ions through fiberoptic wires or using the shared quantum bus. Only specialized states that pass the verification procedures will be used for gate teleportation where Bell measurements are performed. Of course, this scheme would require sufficient amount of resources invested in the preparation of the specialized ancillary states for gate teleportation. The logical data qubits and specialized ancillary qubits would necessarily be equipped with the error-correction mechanisms needed for each logical qubit tile, further increasing the amount of error-correction resources.

An alternative construction would be to use gate teleportation to speed up quantum applications. For example, if qubits Q_1 and Q_2 are required for a certain sequence of operations and the two qubits reside in the memory region, the first operation in the program can be performed while teleporting the qubits to the processing region.

11.3 ERROR CORRECTION THROUGH TELEPORTATION

Even more amazing are the potential system level error-correction advantages gained when allowing sufficient interconnect bandwidth such that *encoded* EPR pairs are communicated instead of elementary EPR pairs. Note the relationship between the control data qubit and the nearest EPR qubit in Fig. 11.8 (lines 1 and 2). If the qubits are encoded using some CSS $[\![n, k, d]\!]$ code such as the Steane $[\![7, 1, 3]\!]$ code, then the sequence of operations between lines 1 and 2 is strikingly similar to the Steane method for error correction. As a matter of fact it is the Steane method, and while we are teleporting the gate, the measurement performed is equivalent to extracting the error syndrome of the data.

In fact, teleportation itself is error correction. Let's take a step back and consider the standard teleportation procedure that simply teleports quantum data as in the original circuit of Fig. 2.8 in Section 2.3.1, or the CNOT gate from Fig. 11.2. If less than $t = (d - 1)/2$ errors have occurred on the logical data by the time the Bell measurement is complete, then the encoded state of the qubit will be *correctly* identified through the logical measurement operation. The correct state will then be recreated at the destination EPR logical qubit. If the EPR qubit is sufficiently well distilled, teleportation is another method for correcting errors on encoded data. The only difference is that the data is not corrected in place given some error syndrome but recreated at some other location marked by the logical destination EPR qubit. A large-scale system designer can explore the potential trade-offs that may arise in the fault-tolerant properties of the architecture vs. the required communication bandwidth as encoded EPR pairs are used to connect distant logical qubits. One of the most intuitive potential advantages offered by teleportation-based error correction is the possibility to reduce the number of error correction procedures required to perform on the logical data after each computational step.

Knill [64, 93, 94] has studied the fault tolerant properties of using teleportation as error-correction protocol applied on linear optical architectures. He has devised extensive *error-detecting code* procedures and has demonstrated that the accuracy threshold for scalable quantum computation can be as high as 1% error rate per physical gate. His method of *postselected quantum computation* uses the property that logical states used for computation are accepted only if no errors are detected with sufficiently high probability. He uses simple two-and six-qubit concatenated quantum error-detecting codes to show that, by postselecting the output of the logical operations, the probability of error in his architecture can be reduced arbitrarily. All quantum logic in knill's architecture models is performed through teleportation of gates rather than direct qubit-qubit interactions.

CHAPTER 12

Concluding Remarks

In this book, we have explored the design of large-scale quantum architectures in the context of system-level balance between fault-tolerant, logical qubit structures and communication mechanisms that protect quantum data while in transmission. Logical qubit structures include the number of ancillary qubits necessary for the required rate of error correction. The bandwidth of the interconnect channels is balanced with the size and speed of the computational blocks that work on these logical qubits. The distribution of the quantum computational resources is matched to the application's support for gate teleportation or data teleportation, and thus allowing for the creation of logical teleportation resources. The amount of usage for different error-correcting codes or levels of encoding is matched to the size of the application and the needed reliability to finish the application with a high enough success rate. In general, the inherently high decoherence rate of quantum information places the issue of fault tolerance at the heart of a balanced system design.

Design of large-scale quantum systems is in its infancy. As quantum technologies continue to improve, however, the opportunities for system designers will dramatically increase. There are already several groups exploiting these opportunities:

- Emanuel Knill at the National Institute for Standards and Technology (NIST) is the leading architect behind fault-tolerant optical systems with teleportation-based error correction and gate implementations [64, 94]. His leading work in teleportation-based error detecting and correcting schemes offers one of the most viable alternatives to the architectures based on the Steane method of error correction.
- Mark Oskin and David Bacon at the University of Washington are working to design tools to study and model quantum architectures based on some of the most efficient error-correcting codes known [53, 54, 146, 147, 176].
- Mircea Vladutiu from the Politehnica University of Timisoara, Romania has published extensively about modeling quantum algorithms on reconfigurable circuit structures that use reconfigurability to improve the scalability of quantum error-correcting codes [177, 178]
- Researchers at the Quantum Architectures Research Center (QARC) led by Isaac Chuang at MIT, Frederic T. Chong at UC Santa Barbara, Mark Oskin at the University

of Washington, and John Kubiatowicz at UC Berkeley, have made a significant impact on the studying of the implementation and control of classical control structures for emerging quantum technologies [153, 54, 179].

- Teleportation-based distributed quantum systems for large-scale applications are being studied at Keio University, Japan guided by Kohei Itoh [175, 176].
- The quantum circuits led by Igor Markov and Columbia (Alfred Aho) have provided significant contributions to quantum logic circuit synthesis and testing, including the development of fault-tolerant software architecture for quantum computers that maps a high-level program into fault-tolerant machine-level instructions [127, 130, 162, 166, 167].
- The QLA Model has recently provided a base architecture for system designers to work with and improve as evidenced from the work led by Prof. T. N. Vijaykumar at Purdue University.
- In general, the numerous theoretical and experimental research projects that are ongoing, make the field of quantum computing one of the fastest growing fields of science.

We have focused on the QLA architecture as a case study from which to develop a framework of architectural abstractions. To model the QLA architecture we have made some very strict design assumptions such as the fault-tolerant structure of the long-distance interconnect, the error-correcting code of the encoded qubits, and finally the low-level microarchitecture model based on the ion-trap technology. While the assumptions made are sufficient to demonstrate that, within existing technological boundaries, scalable quantum computation is feasible, there are still many possibilities for constructing the basic fault-tolerant elements of an architecture. Our hope is that this book will help provide the necessary background and abstractions for system designers to explore this space of technologies and potential designs. Leveraging our collective experience in computer design will be instrumental in making practical quantum computing a reality.

ACKNOWLEDGMENTS

The authors would like to thank their research collaborators, Darshan Thaker, Jedidiah Crandall, John Oliver (all from UC Davic), Andrew Cross, and Professor Isaac Chuang from MIT for discussing much of the material with us and most importantly providing us with useful criticism for the content of this book. We would also like to thank MIT ion trappers Ken Brown, Jaroslaw Labaziewicz, and Rob Clark for answering technical questions about the physical implementations of quantum computers. Rodney Van Meter (Keio University, Japan), Marc de Kruijf (University of Madison, Wisconsin), Jack Lewin (UCI) provided invaluable line-by-line comments in the later stages of writing. And finally, we thank Mark Hill from the University of Wisconsin, Madison for making this book possible and subsequently providing us with the necessary guidance for the content of the book.

Appendix Timeline of Quantum Computers

The timeline for quantum computation is largely taken from [180] with some needed additions such as references to the relevant articles and additional technological and theoretical contributions we found important to include.

1973 • Alexander Holevo publishes a paper showing that n qubits cannot carry more than n classical bits of information [29].

1975 • R. P. Poplavskii shows that simulating quantum systems on classical computers is computationally infeasible due to the superposition principle [181].

1976 • Polish mathematical physicist Roman Ingarden [182] shows that Shannon information theory cannot directly be generalized to the quantum case, but rather that it is possible to construct a quantum information theory which is a generalization of Shannon's theory.

1980 • Yuri Manin discusses the need for a theory of quantum computation that captures the fundamental principles of computation without committing to a physical realization [183].

1981 • Richard Feynman in his talk at the First Conference on the Physics of Computation, held at MIT, observed that the act of setting up a multiparticle interference experiment and measuring the outcome is equivalent to performing quantum computation exponentially more powerful than the classical simulation of the experiment. • Tommaso Toffoli introduced the reversible Toffoli gate [34], which provides a universal set for reversible classical computation.

1984 • Charles Bennett and Gilles Brassard employ Wiesner's conjugate coding for distribution of cryptographic keys [40, 16].

1985 • David Deutsch describes the first universal quantum computer based on a universal quantum Turing machine [5, 4].

1991 • Artur Ekert invents entanglement-based secure communication [184].

1993 • Dan Simon invents an oracle problem for period finding, where a quantum computer would be exponentially faster than conventional computer [39].

1994 • Peter Shor extends Simon's work to create an algorithm that allows a quantum computer to factor large integers quickly [7]. The algorithm solves both the factoring problem and the discrete log problem becoming the first discovery that threatens the security of some of the most important cryptographic schemes such as the RSA [8] public key encryption. Additionally, physical realization of Shor's algorithm quickly becomes the driving force behind realizing scalable and reliable quantum computation.

1995 • Benjamin Schumacher discovers a way to interpret quantum states as information and coins the term *qubit* [30]. • Ignacio Cirac, at University of Castilla-La Mancha at Ciudad Real, and Peter Zoller and the University of Innsbruck proposed an experimental realization of the controlled-NOT gate with trapped ions [24]. • Peter Shor and Andrew Steane simultaneously proposed the first schemes for quantum error correction. This is recognized as the key technology for building large-scale quantum computers that work and the first step toward eliminating the prohibitive nature of decoherence. • Christopher Monroe and David Wineland at NIST (Boulder, Colorado) experimentally realize the first quantum logic gate with trapped ions, according to Cirac and Zoller's proposal.

1996 • Lov Grover invents the quantum database search algorithm [10], allowing the potential to solve in quadratic time any brute-force random search problem. • Daniel Gottesman publishes the first paper [115] that classifies the stabilizer class of quantum error-correcting codes and defines the stabilizer formalism.

1997 • David Cory, Amr Fahmy, and Timothy Havel [56], and at the same time Neil Gershenfeld and Isaac L. Chuang at MIT [57], publish the first papers on quantum computers based on bulk spin resonance. Qubits are stored in the spin of the protons and neutrons of small molecules and placed in MRI machines.

1998 • Chuang, Gershenfeld, and Kubinec demonstrate the first execution of Grover's algorithm [185]. • Daniel Gottesman formulates the Heisenberg representation for quantum codes [186], which is responsible for the Gottesman–Knill theorem allowing efficient simulation of stabilizer quantum circuits.

1999 • Sympathetic cooling is used to cool trapped ions for quantum computation [103]. • Distant atoms are entangled indirectly by coupling them with photon-based qubits [58]. • Daniel Gottesman and Isaac Chuang demonstrate that quantum teleportation can be used as a logically universal computational primitive [125].

2000 • Researches at the Technical University of Munich demonstrate the first working five-qubit NMR quantum computer [187]. • Briegel and Raussendorf formulate the cluster-state model for quantum computation which allows universal computation through single-qubit measurements [46]. • David DiVincenzo formulates the five plus two requirements for quantum technologies proposals that aim to demonstrate scalable, general-purpose quantum computation [32].

2001 • First execution of Shor's algorithm at IBM's Almaden Research Center led by Isaac Chuang and researchers at Stanford University [1]. The number 15 was factored using 1018 identical molecules, each containing seven active nuclear spins. • Experimental implementation of the order-finding algorithm is demonstrated by the same research team [188]. • Emanuel Knill develops an efficient scheme of scalable quantum computation using linear optical components and measurements [64].

2002 • The Quantum Information Science and Technology Roadmapping Project, involving some of the main participants in the field, laid out the quantum computation roadmap [52]. • Mark Oskin, Fred Chong, and Isaac Chuang publish the first work on comprehensive scalable quantum architecture design [53]. • Kielpinski, Wineland, and Monroe propose the CCD-based architecture as the first truly scalable scheme for large-scale quantum computation based on the trapped-ions technology [25].

2003 • Shi-Biao Zheng and colleagues experimentally demonstrate quantum teleportation using the cluster state model for quantum computing [47]. • Michael Freedman and colleagues from Caltech formulate the topological model for quantum computation [51].

2004 • A collaboration of researchers proves that adiabatic quantum computation is equivalent to the circuit model of quantum computing [45]. • Michael Nielsen and C. Dawson publish the first work on scalable fault-tolerant computation using cluster states [48]. • Independent experiments at the National Institute for Standard and Technology (NIST) [77] led by David Wineland and a team in Austria [78] led by Rainer Blatt successfully realize quantum teleportation with trapped atomic ions. Their experiments demonstrate all necessary components in practice for a scalable quantum architecture.

2005 • Researchers at the Georgia Institute of Technology led by Alex Kuzmich demonstrate storage and retrieval of single photonic qubit states between remote quantum memories [189] by transferring the state from photons to atoms and back again, marking an important step toward distributed quantum computation. • A scalable quantum computer chip for atomic qubits was built for the first time by researchers at the University of Michigan led by Christopher Monroe [108], offering hopes for making a practical quantum computer using conventional semiconductor manufacturing technology. • David Bacon from the University of Washington develops the idea of self-correcting quantum memories using operator quantum error correction [1], which leads David Poulin from Caltech to formulate the highly efficient structure of the Bacon-Shor codes [2].

2006 • HP Labs' Quantum Information Processing Group begins finding ways to use photons, or light particles, for information processing, rather than the electrons used in digital electronic computers today [190]. Their work holds promise for someday developing faster, more powerful, and more secure computer networks. • Peter Zoller, from the University of Innsbruck in Austria, discovers method of using cryogenic polar molecules to make stable quantum memories [191]. • Researchers at Cambridge University and Toshiba announce a new quantum device that produces entangled photons [192]. • Ameenah Al-Ahmadi and Sergio Ulloa from Ohio University discover how to make coherent light travel between quantum dots, facilitating communication in optical quantum computers [193]. • Sam Braunstein at the University of York along with the University of Tokyo, and the Japan Science and Technology Agency gave the first experimental demonstration of quantum telecloning [194]. Researches led by David Wineland at NIST are able to trap atomic ions on a silicon-based chip paving the way for smaller and more reliable ion-trap quantum computers [61]. • Researchers at the University of California Santa Barbara led by J. Martinis and University of California Riverside led by A. Korotkov experimentally demonstrate measurement of superconducting qubits [1] • Researchers at the University of Southern California led by Min-Hsiu Hsieh develop an alternative theory of quantum error correction using entanglement [2].

References

[1] L. M. Vandersypen et al., "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, Vol. 414, p. 883, 2001. doi:10.1038/414883a

[2] J. Kim, S. Pau, Z. Ma, H. R. McLellan, J. V. Gates, A. Kornblit, and R. E. Slusher, "System design for a large-scale ion-trap quantum information processor," *Quantum Inf. Comput.*, Vol. 5, No. 7, pp. 515, 2005.

[3] P. Benioff, "Quantum mechanical models of Turing machines that dissipate no energy," *Phys. Rev. Lett.*, Vol. 48, pp. 1581–1585, 1982. doi:10.1103/PhysRevLett.48.1581

[4] D. Deutsch, "Quantum theory, the Church-turing principle and the universal quantum computer," In *Proc. R. Soc. Lond.*, Vol. A 400, pp. 97–117, 1985.

[5] D. Deutsch, "Quantum computational networks," *Proc. R. Soc. Lond.*, Vol. A 400, pp. 97–117, 1985.

[6] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM J. Comput.*, Vol. 26, No. 5, pp. 1411–1473, 1997. doi:10.1137/S0097539796300921

[7] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," in *35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[8] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, Vol. 21, No. 2, pp. 120–126, 1978. doi:10.1145/359340.359342

[9] J. P. Buhler, H. W. Lenstra, and C. Pomerance, "Factoring integers with the number field sieve," *The Development of the Number Field Sieve, Lecture Notes in Mathematics*, Vol. 1554, Berlin: Springer-Verlag, 1994, pp. 50–94.

[10] L. Grover, "A fast quantum mechanical algorithm for database search," in *Symposium on Theory of Computing (STOC 1996)*, pp. 212–219.

[11] A. M. Childs, E. Farhi, and J. Preskill, "Robustness of adiabatic quantum computation," *Phys. Rev. A*, Vol. 65, 2002, quant-ph/0108048.

[12] I. L. Chuang, "Quantum algorithm for clock synchronization," *Phys. Rev. Lett.*, Vol. 85, 2006, 2000, quant-ph/0005092.

[13] C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," in *IEEE International Conference on Computers, Systems, and Signal Processing*, 1984, pp. 175–179.

[14] W. van Dam and G. Seroussi, "Efficient quantum algorithms for estimating gauss sums," 2002, e-print: quant-ph/0207131. doi:10.1103/PhysRevLett.77.2818

[15] S. Hallgren, "Polynomial time quantum algorithms or Pell's equation and the principal ideal problem," in *Symposium on Theory of Computing (STOC 2002)*, May 2002, pp. 653–658.

[16] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *J. Cryptogr.*, Vol. 5, No. 1, 1992.

[17] S. M. Barnett and S. J. D. Phoenix, "Information-theoretic limits to quantum cryptography," *Phys. Rev. A*, Vol. 48, No. 1, pp. R5–R8, 1993.

[18] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Phys. Rev. Lett.*, Vol. 77, 1996, pp. 2818–2821. doi:10.1103/PhysRevLett.77.2818

[19] H. K. Lo and H. F. Chau, "Unconditional security of quantum key distribution," *Science*, Vol. 283, pp. 2050–2056, 1999, quant-ph/9803006. doi:10.1126/science.283.5410.2050

[20] T. Nakassis, J. C. Beinfang, P. Johnson, A. Mink, D. Rogers, X. Tang, and C. J. Williams, "Has quantum cryptography been proven secure," in *Proc. SPIE Defense and Security Symposium,* Orlando, FL, 2006.

[21] G. Gelfond, MagiQ technologies. www.magiqtech.com, 2002. doi:10.1103/PhysRevLett.74.4091

[22] G. Ribordy, O. Guinnard, and H. Zbinden, "id Quantique," www.idquantique.com, 2004.

[23] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, Vol. 54, pp. 24–93, 1995.

[24] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett*, Vol. 74, pp. 4091–4094, 1995. doi:10.1103/PhysRevLett.74.4091

[25] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, Vol. 417, pp. 709–711, 2002. doi:10.1038/nature00784

[26] C. H. Bennett et al., "Teleporting an unknown quantum state via dual classical and EPR channels," *Phys. Rev. Lett.*, Vol. 70, pp. 1895–1899, 1993. doi:10.1103/PhysRevLett.70.1895

[27] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "A quantum logic array microarchitecture: scalable quantum data movement and computation," in *Proc. 38th International Symposium on Microarchitecture*, MICRO-38, 2005.

[28] D. D. Thaker, T. S. Metodi, A. W. Cross, F. T. Chong, and I. L. Chuang, "Quantum memory hierarchies: efficient designs to match available parallelism in quantum computing," in *International Symposium of Computer Architecture (ISCA-33),* Boston, MA, 2006.

[29] A. S. Holevo, "Bounds for the quantity of information transmitted by a quantum communication channel," *Prob. Pered. Inf*, Vol. 9, No. 3, pp. 3–11, 1973.

[30] B. Schumacher, "Quantum coding," *Phys. Rev. A*, Vol. 51, pp. 2738–2747, 1995. doi:10.1103/PhysRevA.51.2738

[31] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proc. R. Soc. Lond.*, Vol. A 439, pp. 553–558, 1992.

[32] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschr. Phys.*, Vol. 48, pp. 771–783, 2000, quant-ph/0002077. doi:10.1038/299802a0

[33] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A.*, Vol. 52, p. 3457, 1995, quant-ph/9503016.

[34] T. Toffoli, *Reversible Computing*, New York: Springer, 2000.

[35] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum Algorithms Revisited," *In Proceedings of the Royal Society of London*, Vol. A, No. 454, pp. 339–354, 1997, quant-ph/9708016.

[36] G. Song and A. Klappenecker, "Optimal realizations of controlled unitary gates," *J. Quantum Inform. Comput.*, Vol. 3, No. 2, pp. 139–155, 2003, quant-ph/0207157.

[37] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, Vol. 299, pp. 802–803, 1982. doi:10.1038/299802a0

[38] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge, UK: Cambridge University Press, 2000.

[39] D. Simon, "On the power of quantum computation," in *Proc. 35th Annual Symposium on Foundations of Computer Science IEEE Computer Society Press,* Los Alamitos, CA, 1994, pp. 116–123.

[40] C. Bennett and S. J. Wiesner, "Communication via one- and two-particle operators on Einstein–Podolsky–Rosen states," *Phys. Rev. Lett.*, Vol. 69, No. 2881, 1992.

[41] A. Harrow, P. Hayden, and D. Leung, "Superdense coding of quantum states," *Phys. Rev. Lett.*, Vol. 92, No. 187901, 2004, quant-ph/0307221.

[42] F. A. Wolf, *Taking the Quantum Leap*, San Francisco, CA: Harper and Snow, 1981.

[43] C. Monroe, "Quantum information processing with atoms and photons," *Nature*, Vol. 416, pp. 238, 2002. doi:10.1038/416238a

[44] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution," 2000, quant-ph/0001106. doi:10.1103/PhysRevLett.86.910

[45] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, “Adiabatic quantum computation is equivalent to standard quantum computation,” 2004.

[46] H. J. Briegel and R. Raussendorf, “Persistent entanglement in arrays of interacting particles,” *Phys. Rev. Lett*, Vol. 86, pp. 910–913, 2001, quant-ph/0004051. doi:10.1103/PhysRevLett.86.910

[47] B. Zeng, D. L. Zhou, Z. Xu, and C. P. Sun, “Quantum teleportation using cluster states,” *Quantum Physics,* 2003, e-print: quant-ph/0304165.

[48] M. A. Nielsen and C. M. Dawson, “Fault-Tolerant Quantum Computation with Cluster States,” 2004. doi:10.1038/35002528

[49] M. Nielsen, “Optical quantum computation using cluster states,” *Phys. Rev. Lett*, Vol. 93 (040503), 2004, quant-ph/0402005.

[50] J. A. Jones, V. Vedral, A. Ekert, and G. Castagnoli, “Geometric quantum computation using nuclear magnetic resonance,” *Nature*, Vol. 403, pp. 869–871, 2000. doi:10.1038/35002528

[51] M. H. Freedman, A. Kitaev, M. J. Larsen, and Z. Wang, “Topological quantum computation,” *Bull. Am. Math. Soc.*, Vol. 40, pp. 31–38, 2003, quant-ph/0101025. doi:10.1090/S0273-0979-02-00964-3

[52] D. Wineland and T. Heinrichs, “Ion trap approaches to quantum information processing and quantum computing,” in *A Quantum Information Science and Technology Roadmap*, 2004. URL: http://quist.lanl.gov.

[53] M. Oskin, F. Chong, and I. Chuang, “A Practical Architecture for Reliable Quantum Computers,” *IEEE Comput.*, Vol. 35, No. 1, p. 79–87, January 2002.

[54] M. Oskin, F. T. Chong, J. Kubiatowicz, and I. L. Chuang, “Building quantum wires: the long and the short of it,” in *ISCA-30,* San Diego, CA, 2003.

[55] D. Copsey et al., “Toward a Scalable, Silicon-Based Quantum Computing Architecture,” *Selected Topics, J. Quantum Electron.*, Vol. 9, No. 6, pp. 1552–1569, 2003. doi:10.1109/JSTQE.2003.820922

[56] D. Cory, A. Fahmy, and T. Havel, “Nuclear magnetic resonance spectroscopy: an experimentally accessible paradigm for quantum computing,” in *Proc. 4th Workshop on Physics and Computation,* New England Complex Systems Institute, 1996.

[57] N. A. Gershenfeld and I. L. Chuang, “Bulk spin-resonance quantum computation,” *Science*, Vol. 275, No. 350, pp. 350–356, 1997. doi:10.1126/science.275.5298.350

[58] C. Cabrillo et al., “Creation of entangled states of distant atoms by interference,” *Phys. Rev. A*, Vol. 59, pp. 1025–1033, 1999. doi:10.1103/PhysRevA.59.1025

[59] B. B. Blinov, L. Deslauriers, P. Lee, M. J. Madsen, R. Miller, and C. Monroe, “Sympathetic cooling of trapped Cd^+ isotopes,” *Phys. Rev. A.*, Vol. 65, pp. 040–304, 2002.

[60] D. J. Wineland et al., "Experimental issues in coherent quantum-state manipulation of trapped atomic ions," *J. Res. NIST*, Vol. 103, pp. 259–328, 1998, quant-ph/9710025.

[61] J. Britton, D. Leibfried, J. Beall, R. B. Blakestad, J. J. Bollinger, J. Chiaverini, R. J. Epstein, J. D. Jost, D. Kielpinski, C. Langer, R. Ozeri, R. Reichle, S. Seidelin, N. Shiga, J. H. Wesenberg, and D. J. Wineland, "A microfabricated surface-electrode ion trap in silicon," 2006, quant-ph/0605170. doi:10.1103/PhysRevLett.62.2124

[62] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble, "Measurement of conditional phase shifts for quantum logic," *Phys. Rev. Lett.*, Vol. 75, p. 4710, 1995.

[63] G. J. Milburn, "Quantum optical fredkin gate," *Phys. Rev. Lett.*, Vol. 62, pp. 2124–2127, 1989. doi:10.1103/PhysRevLett.62.2124

[64] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *Nature*, Vol. 409, pp. 46–52, 2001. doi:10.1038/35051009

[65] T. D. Ladd, J. R. Goldman, F. Yamaguchi, Y. Yamamoto, E. Abe, and K. M. Itoh, "An All Silicon Quantum Computer," *Physics Online Archive*, 2001.

[66] B. Kane, "A silicon-based nuclear spin quantum computer," *Nature*, Vol. 393, pp. 133–137, 1998. doi:10.1038/30156

[67] B. E. Kane, "Silicon based quantum computation," *Prog. Phys.*, Vol. 48, pp. 1023–1041, 2000.

[68] A. J. Skinner, M. E. Davenport, and B. E. Kane, "Hydrogenic spin quantum computing in silicon: a digital approach," *Phys. Rev. L*, Vol. 90, No. 087901, p. 1–4, 2003.

[69] G. Burkard, D. Loss, and D. P. DiVincenzo, "Coupled quantum dots as quantum gates," *Phys. Rev. B*, Vol. 59, pp. 20–70, 1999, cond-map/9808026. doi:10.1103/PhysRevB.59.20

[70] S. Hellberg, "Robust quantum computation with quantum dots," 2003, quant-ph/0304150. doi:10.1109/77.622206

[71] Y. Makhlin, G. Schoen, and A. Shnirman, "Josephson-junction qubits with controlled couplings," *Nature*, Vol. 398, p. 305, 1999.

[72] M. Bocko, A. Herr, and M. Feldman, "Prospects for quantum coherent computation using superconducting electronics," *IEEE Trans. Appl. Supercond.*, Vol. 7, pp. 3638–3641, 1997. doi:10.1109/77.622206

[73] P. M. Platzman and M. I. Dykman, "Quantum computing with electrons floating on liquid helium," *Science*, Vol. 284, pp. 1967–1969, 1999. doi:10.1126/science.284.5422.1967

[74] V. Privman, I. D. Vagner, and G. Kventsel, "Quantum computation in quantum-hall systems," *Phys. Lett. A*, Vol. 239, p. 141, 1998.

[75] L. C. L. Hollenberg, A. S. Dzurak, C. Wellard, A. R. Hamilton, D. J. Reilly, G. J.

Milburn, and R. G. Clark, "Charge-based quantum computing using single donors in semiconductors," *Phys. Rev. B*, Vol. 69, No. 113301, p. 1–4, 2003, cond-mat/ 0306235.

[76] S. Amesha, K. MacLean, D. Zimbühl, I. Radu, M. A. Kastner, M. P. Hanson, and A. C. Gossard, "Toward the manipulation of a single spin in an algaas/gaas single-electron transistor," in *Proc. SPIE Defense and Security Symposium,* Orlando, FL, 2006.

[77] M. Barrett, J. Chiaverini, T. Schaetz, J. Britton, et al., "Deterministic quantum teleportation of atomic qubits," *Nature*, Vol. 429, 2004.

[78] M. Riebe, H. Haffner, C. F. Roos, et al., "Deterministic quantum teleportation with atoms," *Nature*, Vol. 429, No. 6993, pp. 734–737, 2004. doi:10.1038/nature02570

[79] R. Van Meter and M. Oskin, "Architectural implications of quantum computing technologies," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, Vol. 2, No. 1, pp. 31–63, 2006. doi:10.1145/1126257.1126259

[80] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, Vol. 74, pp. 145–195, 2002. doi:10.1103/RevModPhys.74.145

[81] Z. Y. Ou and L. Mandel, "Violation of bell's inequality and classical probability in a twophoton correlation experiment," *Phys. Rev. Lett.*, Vol. 61, pp. 50–53, 1988. doi:10.1103/PhysRevLett.61.50

[82] P. Kwiat, K. Mattle, H. Weinfurter, A. Zeilinger, A. V. Sergienko, and Y. H. Shih, "New high-intensity source of polarization-entangled photon pairs," *Phys. Rev. Lett.*, Vol. 75, pp. 4337–4341, 1995. doi:10.1103/PhysRevLett.75.4337

[83] P. Kwiat, E. Waks, A. G. White, I. Appelbaum, and P. H. Eberhard, "Ultrabright source of polarization-entangled photons," *Phys. Rev. A*, Vol. 60, pp. 773–776, 1999. doi:10.1103/PhysRevA.60.R773

[84] D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, "Experimental quantum teleportation," *Nature*, Vol. 390, pp. 575–579, 1997. doi:10.1038/37539

[85] J.-W. Pan, D. Bouwmeester, H. Weinfurter, and A. Zeilinger, "Experimental entanglement swapping: entangling photons that never interacted," *Phys. Rev. Lett.*, Vol. 80, pp. 3891–3894, 1998. doi:10.1103/PhysRevLett.80.3891

[86] D. Boschi, S. Branca, F. De Martini, L. Hardy, and S. Popescu, "Experimental realization of teleporting an unknown pure quantum state via dual classical and Einstein–Podolsky–Rosen channels," *Phys. Rev. Lett.*, Vol. 80, pp. 1121–1125, 1998. doi:10.1103/PhysRevLett.80.1121

[87] A. Furusawa, J. Sorensen, S. L. Braunstein, C. Fuchs, H. J. Kimble, and E. S. Polzik, "Unconditional quantum teleportation," *Science*, Vol. 282, pp. 706–709, 1998. doi:10.1126/science.282.5389.706

[88] Y.-H. Kim, S. P. Kulik, and Y. Shih, "Quantum teleportation of a polarization state with a complete bell state measurement," *Phys. Rev. Lett.*, Vol. 86, pp. 1370–1373, 2001. doi:10.1103/PhysRevLett.86.1370

[89] P. A. Kwiat, J. R. Mitchell, P. D. D. Schwindt, and A. G. White, "Grovers search algorithm: an optical approach," *J. Mod. Opt.*, Vol. 47, pp. 257–266, 2000. doi:10.1080/095003400148187

[90] S. Takeuchi, "Analysis of errors in linear-optics quantum computation," *Phys. Rev. A*, Vol. 61, No. 052302, 2000.

[91] J. C. Howell, J. A. Yeazell, and D. Ventura, "Optically simulating a quantum associative memory," *Phys. Rev. A*, Vol. 62, No. 042303, 2000.

[92] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Phys. Rev. Lett.*, Vol. 73, pp. 58–61, 1998. doi:10.1103/PhysRevLett.73.58

[93] E. Knill, "Quantum gates using linear optics and postselection," *Phys. Rev. A*, Vol. 66, 2002.

[94] E. Knill, "Quantum computing with very noisy devices," 2004, quant-ph/0410199. doi:10.1038/nature02377

[95] L. M. Duan, B. B. Blinov, D. L. Moehring, and C. Monroe, "Scalable trapped ion quantum computation with a probabilistic ion-photon mapping," 2004, e-print: quant-ph/0401020.

[96] B. B. Blinov, D. L. Moehring, L. M. Duan, and C. Monroe, "Observation of entanglement between a single trapped atom and a single photon," *Nature*, Vol. 428, pp. 153–157, 2004. doi:10.1038/nature02377

[97] D. N. Matsukevich and A. Kuzmich, "Quantum state transfer between matter and light," *Science*, Vol. 306, No. 5696, pp. 663–666, 2004. doi:10.1126/science.1103346

[98] T. P. Spiller, K. Nemoto, S. L. Braunstein, W. J. Munro, P. van Loock, and G. J. Milburn, "Quantum computation by communication," 2005, quant-ph/050902. doi:10.1038/nature01492

[99] A. Sorensen and K. Molmer, "Entanglement and quantum computation with ions in thermal motion," *Phys. Lett. A*, Vol. 62, p. 02231, 2000.

[100] D. Leibfried et al., "Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate," *Nature*, Vol. 422, pp. 412–415, 2003. doi:10.1038/nature01492

[101] E. L. Hahn, "Spin echoes," *Phys. Rev.*, Vol. 80, pp. 580–594, 1950. doi:10.1103/PhysRev.80.580

[102] J. V. Porto, S. Rolston, T. B. Laburthe, C. J. Williams, and W. D. Phillips, "Quantum information with neutral atoms as qubits," *Phil. Trans. R. Soc. Lond.*, Vol. A361, pp. 1417–1427, 2003.

[103] B. B. Blinov, L. Deslauriers, P. Lee, M. J. Madsen, R. Miller, and C. Monroe, "Sympathetic cooling of trapped ions for quantum logic," *Phys. Rev. A.*, Vol. 61, p. 032310, 2000, quant-ph/9909035.

[104] R. Ozeri et al., "Hyperfine coherence in the presence of spontaneous photon scattering," 2005, ph/0502063. doi:10.1016/S0924-4247(99)00381-7

[105] A. M. Steane, "How to build a 300 bit, 1 gop quantum computer," 2004, quant-ph/0412165.

[106] M. A. Rowe et al., "Transport of quantum states and separation of ions in a dual rf ion trap," *Quantum Inf. Comput.*, Vol. 2, pp. 257–271, 2002.

[107] J. Chiaverini, R. B. Blakestad J. Britton, J. D. Jost, C. Langer, D. Leibfried, R. Ozeri, and D. J. Wineland, "Surface-electrode architecture for ion-trap quantum information processing," 2004, e-print: quant-ph/0501147.

[108] W. K. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, J. Rabchuk, and C. Monroe, "T-junction ion trap array for two-dimensional ion shuttling, storage and manipulation," 2005, quant-ph/0508097.

[109] S. Seidelin, J. Chiaverini, R. Reichle, J. J. Bollinger, et al., "A microfabricated surface-electrode ion trap for scalable quantum information processing," 2006, e-print: quant-ph/0601173.

[110] E. M. Chow, H. T. Soh, H. C. Lee, J. D. Adams, S. C. Minne, G. Yaralioglu, A. Atalar, C. F. Quate, and T. W. Kenny, "Integration of through-wafer interconnects with a two-dimensional cantilever array," *Sensors Actuators*, Vol. 83, pp. 118–123, 2000. doi:10.1016/S0924-4247(99)00381-7

[111] D. Rosn, J. Olsson, and C. Hedlund, "Membrane covered electrically isolated through-wafer via holes," *J Micromech. Microeng.*, Vol. 11, p. 344, 2001.

[112] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach* (The Morgan Kaufmann Series in Computer Architecture and Design). San Francisco, CA: Morgan Kaufman, 2003.

[113] A. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett*, Vol. 77, pp. 793–797, 1996. doi:10.1103/PhysRevLett.77.793

[114] E. Knill and R. Laflamme, "A theory of quantum error-correcting codes," *Phys. Rev. A*, Vol. 55, pp. 900–911, 1997, quant-ph/9604034. doi:10.1103/PhysRevA.55.900

[115] D. Gottesman, "A class of quantum error-correcting codes saturating the quantum hamming bound," *Phys. Rev. A*, Vol. 54, pp. 18–62, 1996, quant-ph/9604038.

[116] D. Aharonov and M. Ben-Or, "Fault tolerant computation with constant error," pp. 176–188, quant-ph/9906129. doi:10.1080/095003400148240

[117] A. Y. Kitaev, "Quantum error correction with imperfect gates," in *3rd Int. Conf. of Quantum Communication and Measurement*, 1997, pp. 181–188.

[118] D. Gottesman, "Fault tolerant quantum computation with local gates," *J. Mod. opt.*, Vol. 47, pp. 333–345, 2000, quant-ph/9903099. doi:10.1080/095003400148240

[119] A. M. Steane, "Overhead and noise threshold of fault-tolerant quantum error correction," 2002, e-print: quant-ph/0207119. doi:10.1109/18.796388

[120] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, Vol. 54, p. 1098, 1996.

[121] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Enlargement of Calderbank-Shor-Steane quantum codes," *IEEE Trans. Inform. Theor.*, Vol. 45, pp. 2492–2495, 1999, quant-ph/9802061. doi:10.1109/18.796388

[122] D. A. Lidar and L.-A. Wu, "Encoded recoupling and decoupling: an alternative to quantum error correcting codes, applied to trapped ion quantum computation," *Phys. Rev. A.*, Vol. 67, No. 032313, 2003.

[123] J. Von Neuman, "Probabilistic logic and the synthesis of reliable organisms from unreliable components," in *Automata Series,* C. Shannon and J. McCarthy, Eds., Princeton, NJ: Princeton Univ. Press, 1956, pp. 43–98.

[124] A. M. Steane, "Space, time, parallelism and noise requirements for reliable quantum computing," *Fortsch. Phys.*, Vol. 46, pp. 443–458, 1998, quant-ph/9708021. doi:10.1038/46503

[125] D. K. Gottesman and I. L. Chuang, "Quantum teleportation is a universal computational primitive," *Nature*, Vol. 402, pp. 390–392, 1999, quant-ph/9908010. doi:10.1038/46503

[126] D. Gottesman, "Theory of fault-tolerant quantum computation," *Phys. Rev. A*, Vol. 57, pp. 127–137, 1998, quant-ph/9702029. doi:10.1103/PhysRevA.57.127

[127] K. M. Svore, A. W. Cross, A. V. Aho, I. L. Chuang, and I. L. Markov, "Toward a software architecture for quantum computing design tools," in *Workshop on Quantum Programming Languages (QPL)*, 2004.

[128] B. W. Reichardt, "Improved ancilla preparation scheme increases fault-tolearant threshold," 2004, e-print: quant-ph/0406025. doi:10.1038/37539

[129] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung, "Rsa-640 is factored!," 2005.

[130] K. M. Svore, A. W. Cross, I. L. Chuang, and A. Aho, "Pseudothreshold or threshold?—more realistic threshold estimates for fault-tolerant quantum computing," 2005, e-print: quant-ph/0508176.

[131] D. Bouwmeester et al., "Experimental quantum teleportation," *Nature*, Vol. 390, pp. 575–579, 1997. doi:10.1038/37539

[132] J. Lantz, M. Wallquist, V. S. Shumeiko, and G. Wendin, "Josephson Junction Qubit Network with Current-Controlled Interaction," *Phys. Rev. B*, Vol. 70, No. 140507(R), pp. 60–70, 2004.

[133] A. Yao, "Quantum circuit complexity," in *Proc. 34th Annual Symposium on Foundations of Computer Science*, 1993, pp. 352–361.

[134] C. H. Bennett et al., "Purification of noisy entanglement and faithful teleportation via noisy channels," *Phys. Rev. Lett.*, Vol. 76, p. 722, 1996.

[135] W. Dur, H. J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev.*, A59, p. 169, 1999.

[136] K. Michielsen and H. De Raedt. QCE: a simulator for quantum computer hardware," *Turk. J. Phys.*, Vol. 27, p. 343, 2003.

[137] B. Omer, "A procedural formalism for quantum computing: Qcl," *Master thesis technical physics,* TU, Vienna, 1998.

[138] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Graph-based simulation of quantum computation in the density matrix representation," *Quantum Inform. Comput.*, Vol. 5, No. 2, pp. 113–130, 2005, quant-ph/0403114.

[139] L. G. Valiant, "Quantum circuits that can be simulated classically in polynomial time," in *Proc. ACM Symposium on Theory of Computing (STOC)*, 2001, p. 114.

[140] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, Vol. 91, No. 147902, 2003.

[141] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures," in *ISCA-32,* Madison, WI, 2005.

[142] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "Quantum architecture tools: Quale," online at: http://www.cs.washington.edu/homes/lucasks/tools.html. doi:10.1038/35007021

[143] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A*, Vol. 70, No. 052328, 2004, quant-ph/0406196.

[144] Andrew Cross. qasm-tools: An interoperable open-source software tool chain for studying fault-tolerant quantum circuits," Available for download online at: http://web.mit.edu/awcross/www/qasm-tools/.

[145] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for distance 3 codes," 2005, e-print: quant-ph/0504218.

[146] D. Bacon, "Operator quantum error correcting subsystems for self-correcting quantum memories," 2005, e-print: quant-ph/0506023.

[147] D. Poulin, "Stabilizer formalism for operator quantum error correction," 2005, e-print: quant-ph/0508131.

[148] D. J. Wineland, D. Leibfried, M. D. Barrett, A. Ben-Kish, et al., "Quantum control, quantum information processing, and quantum-limited metrology with trapped ions," in *Proc. Int. Conf. on Laser Spectroscopy (ICOLS)*, 2005, quant-ph/0508025.

[149] K. M. Svore, B. Terhal, and D. P. DiVincenzo, "Local fault-tolerant quantum computation," 2004, e-print: quant-ph/0410047.

[150] C. Monroe, C. A. Sackett, D. Kielpinski, B. E. King, C. Langer, V. Meyer, C. J. Myatt, M. Rowe, Q. A. Turchette, W. M. Itano, and D. J. Wineland, "Scalable entanglement of trapped ions," in *Workshop on Trapped Ion Quantum Computing*, NIST, Boulder, CO, 2000.

[151] I. Cirac and P. Zoller, "A scalable quantum computer with ions in an array of microtraps," *Nature*, Vol. 404, pp. 579–581, 2000. doi:10.1038/35007021

[152] J. S. Bell, "On the Einstein–Podolsky-Rosen paradox," *Physics*, Vol. 1, pp. 195–200, 1964.

[153] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection networks for scalable quantum computers," in *International Symposium of Computer Architecture (ISCA-33),* Boston, MA, 2006.

[154] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," 2004, e-print: quant-ph/0406142. doi:10.1109/71.372778

[155] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," 2004, e-print: quant-ph/0408006.

[156] L. McMurchie and C. Eberling, "Pathfinder: a negotiation based performance-driven router for fpgas," in *Proc. ACM Symp. on Field Programmable Gate Arrays*, 1995, pp. 111–117.

[157] H. Chou and C. Chung, "An optimal instruction scheduler for superscalar processor," *IEEE Trans. Parallel Distrib. Syst.*, Vol. 6, No. 3, pp. 303–313, 1995. doi:10.1109/71.372778

[158] B. L. Deitrich and Wen mei W. Hwu, "Speculative hedge: regulating compile-time speculation against profile variations," in *Proc. 29th International Symposium on Microarchitecture*, 1996, p. 29.

[159] C. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B. Rau, and M. Schlansker, "Profile-driven instruction level parallel scheduling with applications to superblocks," in *Proc. 29th Int. Symp. on Microarchitecture*, Vol. 29, 1996, pp. 58–67.

[160] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "Physical operations scheduler in a quantum information processor," in *Proc. SPIE Defense and Security Symposium,* Orlando, FL, 2006.

[161] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing cnot-based quantum circuits," in *Proc. Design Automation Conference (DAC)*, 2002, pp. 419–424, quant-ph/0401162.

[162] K. N. Patel, I. L. Markov, and J. P. Hayes, "Efficient synthesis on linear reversible circuits," 2003, e-print: quant-ph/0302002. doi:10.1103/PhysRevLett.78.2252

[163] G. Song and A. Klappenecker, “Optimal realizations of controlled unitary gates,” *J. Quantum Inform. Comput.*, Vol. 3, No. 2, pp. 139–155, 2003, quant-ph/0207157.

[164] V. V. Shende, S. S. Bullock, and I. L. Markov, “Recognizing small-circuit structure in two-qubit operators and timing Hamiltonians to compute controlled-not gates,” *Phys. Rev. A*, (012310), 2003, quant-ph/0308045.

[165] V. V. Shende, I. L. Markov, and S. S. Bullock, “Minimal universal two-qubit quantum circuits,” *Phys. Rev. A*, Vol. 69, No. 062321, pp. 1–7, 2003, quant-ph/0308033.

[166] Vivek V. Shende, Igor L. Markov, and Stephen S. Bullock, “Finding small two-qubit circuits,” *Proc. SPIE*, No. 5436, pp. 348–359, 2004.

[167] V. V. Shende and I. L. Markov, “Quantum circuits for incompletely specified two-qubit operators,” *Quantum Inform. Comput.*, Vol. 5, No. 5, pp. 048–056, 2005.

[168] T. Hogg, C. Mochon, W. Polak, and E. Rieffel, “Tools for quantum algorithms,” *Int. J. Mod. Phys.*, Vol. C10, pp. 1347–1362, 1999, quant-ph/9811073.

[169] S. S. Bullock and I. L. Markov, “Asymptotically optimal circuits for arbitrary *n*-qubit diagonal computations,” *Quantum Inform. Comput.*, Vol. 4, No. 1, pp. 027–047, 2004, quant-ph/0303039.

[170] A. M. Steane, “Efficient fault-tolerant quantum computing,” *Phys. Rev. Lett.*, Vol. 78, pp. 2252–2255, 1997, quant-ph/9809054. doi:10.1103/PhysRevLett.78.2252

[171] P. W. Shor, “Fault-tolerant quantum computation,” in *Proc. 37th Symp. on Foundations of Computer Science*, 1996.

[172] E. Knill, R. Laflamme, and W. Zurek, “Threshold accuracy for quantum computation,” 1996. doi:10.1119/1.16243

[173] D. Greenberger, M. Horne, A. Shimony, and Zeilinger, “Bell’s theorem without the inequalities,” *Am. J. Phys.*, Vol. 58, pp. 1131–1143, 1990. The GHZ state inventors. doi:10.1119/1.16243

[174] X. Zhou, D. Leung, and I. L. Chuang, “Methodology for quantum logic gate construction,” e-print: quant-ph/0002039. doi:10.1145/1126257.1126259

[175] R. Van Meter et al., “Distributed arithmetic on a quantum multi-computer,” in *International Symposium of Computer Architecture (ISCA-33),* Boston, MA, 2006.

[176] R. Van Meter and M. Oskin, “Architectural implications of quantum computing technologies,” *ACM J. Emerging Technol. Comput. Syst. (JETC)*, Vol. 2, No. 1, pp. 31–68, 2006. doi:10.1145/1126257.1126259

[177] M. Udrescu, L. Prodan, and M. Vladutiu, “Using hdls for describing quantum circuits: a framework for efficient quantum algorithm simulation,” in *2nd ACM International Conference on Computing Frontiers (CF’05),* Ischia, Italy, 2004, pp. 96–110.

[178] M. Udrescu, L. Prodan, and M. Vladutiu, "Improving quantum circuit dependability with reconfigurable quantum gate arrays," in *2nd ACM International Conference on Computing Frontiers (CF'05),* Ischia, Italy, 2005, pp. 133–144.

[179] N. Isailovic, M. Whitney, Y. Patel, J. Kubiatowicz, D. Copsey, F. T. Chong, I. L. Chuang, and M. Oskin, "Datapath and control for quantum wires," *Trans. Archit. Code Optim. (TACO),* Vol. 1, No. 1, pp. 34–61, 2004. doi:10.1145/980152.980155

[180] Timeline for quantum computing. http://en.wikipedia.org/wiki/Timeline-of-quantum-computing, 2006. doi:10.1016/0034-4877(76)90005-7

[181] R. P. Poplavskii, "Thermodynamical models of information processing," *Usp. Fiz. Nauk*, Vol. 115, No. 3, pp. 465–501, 1975.

[182] R. S. Ingarden, "Quantum information theory," *Rep. Math. Phys.*, Vol. 10, pp. 43–72, 1976. doi:10.1016/0034-4877(76)90005-7

[183] Y. Manin, "Computable and uncomputable," Moscow, Sovetskoye Radio, 1980. doi:10.1103/PhysRevLett.69.1293

[184] A. Ekert et al., "Practical quantum cryptography based on two-photon interferometry," *Phys. Rev. Lett.*, Vol. 69, pp. 1293–1295, 1992. doi:10.1103/PhysRevLett.69.1293

[185] I. L. Chuang, N. Gershenfeld, and M. Kubinec, "Experimental implementation of fast quantum searching," *Phys. Rev. Lett.*, Vol. 18, No. 15, pp. 3408–3411, 1998. doi:10.1103/PhysRevLett.80.3408

[186] D. Gottesman, "The heisenberg representation of quantum computers," *Hobart, Group theoretical methods in physics*, 1998, pp. 32–43, e-print: quant-ph/9807006.

[187] R. Marx, A. F. Fahmy, J. M. Myers, W. Bermel, and S. J. Glaser, "Realization of a 5-bit nmr quantum computer using a new molecular architecture," *Phys. Rev. A*, Vol. 62, No. 012310, pp. 1–8, 2000. doi:10.1146/annurev.physiol.62.1.1

[188] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang, "Experimental realization of order-finding with a quantum computer," *Phys. Rev. Lett.*, Vol. 15, pp. 5452–5455, December 15, 2000. doi:10.1103/PhysRevLett.85.5452

[189] T. Chaneliere, D. N. Matsukevich, S. D. Jenkins, S.-Y. Lan, T. A. B. Kennedy, and A. Kuzmich, "Storage and retrieval of single photons transmitted between remote quantum memories," *Nature*, Vol. 438, pp. 833–836, 2005. doi:10.1038/nature04315

[190] W. J. Munro, K. Nemoto, and T. P. Spiller, "Weak nonlinearities: a new route to optical quantum computation," *New J. Phys.*, Vol. 7, p. 137, 2005, quant-ph/0507084.

[191] A. Andre, D. DeMille, J. M. Doyle, M. D. Lukin, S. E. Maxwell, P. Rabl, R. Schoelkopf, and P. Zoller, "Polar molecules near superconducting resonators: a coherent, all-electrical, molecule-mesoscopic interface," 2006, e-print: quant-ph/0605201.

[192] J. C. Blakesley, P. See, A. J. Shields, B. E. Kardynal, P. Atkinson, I. Farrer, and D. A. Ritchie, "Efficient single photon detection by quantum dot resonant tunneling diodes," *Phys. Rev. Lett.*, Vol. 94, No. 6, p. 067401, 2005.

[193] A. N. Al-Ahmadi and S. E. Ulloa, "Extended coherent exciton states in quantum dot arrays," *Appl. Phys. Lett.*, Vol. 88, No. 043110, 2006.

[194] S. Koike, H. Takahashi, H. Yonezawa, N. Takei, S. L. Braunstein, T. Aoki, and F. Furusawa, "Demonstration of quantum telecloning of optical coherent states," *Phys. Rev. Lett.*, Vol. 96, pp. 060504, 2006. doi:10.1103/PhysRevLett.96.060504

Biographies

Fred Chong is a professor of computer science at the University of California at Santa Barbara. Prof. Chong received his BS in 1990, MS in 1992, and PhD in 1996, all from MIT. He was an assistant professor at UC Davis from 1996–2001, was an associate professor at UC Davis from 2001–2005, and has been a professor at UCSB from 2005-present. Dr. Chong's research interests include quantum computing architectures, nanoscale electronics, embedded processing, computer security, and the environmental impact of computing technologies. Prof. Chong is a UC Davis Chancellor's Fellow (2002–2007) and received an NSF CAREER Award (1998–2002).

Tzvetan Metodi is a 5th year computer science PHD student at the University of California at Davis as a member of the computer architecture laboratory under the guidence of Professor Frederic T. Chong. Tzvetan received his B.A. in physics also at UC Davis in 2002. He spent August 2003 through December 2003, and January 2006 through June 2006 as a visiting scholar at MIT under the guidance of Professor Isaac L. Chuang. Tzvetan is a member of the Quantum Computer Architecture Center (QARC), which intends to create an interoperable software tool chain for fault-tolerant quantum computer architecture synthesis and evaluation.