STUDIES OF A QUANTUM SCHEDULING ALGORITHM AND ON QUANTUM ERROR CORRECTION

by

FENG LU
B.S. Tsinghua University, 1998
M.S. University of Central Florida, 2006

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2007

Major Professors:
Dan C. Marinescu

ABSTRACT

Quantum computation has been a rich field of study for decades because it promises possible spectacular advances, some of which may run counter to our classically rooted intuitions. At the same time, quantum computation is still in its infancy in both theoretical and practical areas. Efficient quantum algorithms are very limited in number and scope; no real breakthrough has yet been achieved in physical implementations.

Grover's search algorithm can be applied to a wide range of problems; even problems not generally regarded as searching problems can be reformulated to take advantage of quantum parallelism and entanglement leading to algorithms which show a square root speedup over their classical counterparts. This dissertation discusses a systematic way to formulate such problems and gives as an example a quantum scheduling algorithm for an $R||C_{max}$ problem. This thesis shows that quantum solution to such problems is not only feasible but in some cases advantageous.

The complexity of the error correction circuitry forces us to design quantum error correction codes capable of correcting only a single error per error correction cycle. Yet, time-correlated errors are common for physical implementations of quantum systems; an error corrected during a certain cycle may reoccur in a later cycle due to physical processes spe-

cific to each physical implementation of the qubits. This dissertation discusses quantum error correction for a restricted class of time-correlated errors in a spin-boson model. The algorithm proposed allows the correction of two errors per error correction cycle, provided that one of them is time-correlated. The algorithm can be applied to any stabilizer code, perfect or non-perfect, and simplified the circuit complexity significantly comparing to the classic quantum error correction codes.

ACKNOWLEDGMENTS

The first person I want to thank is my advisor, Prof. Dan C. Marinescu. He gave me guidance, encouragement and sound advice when most needed. He also gave me the freedom on my research work while strongly supporting my academic endeavors. I would like to thank Eduardo Mucciolo and Gabriela M. Marinescu, who gave me a physical insight to the quantum world. Also many thanks to Pawel Wocjan and Michael Leuenberger who enhanced my understanding of mathematics behind quantum power.

Many thanks to the members of CCC(Chinese Culture Club), without you guys, my life in orlando will be lonely and boring. Finally, I am forever grateful to my parents, for their constant support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The idea of quantum computer was first explored in the 1970's and early 1980's by physicists and computer scientists such as Charles H. Bennett, Paul A. Benioff and the late Richard P. Feynman. The idea emerged when they were pondering the fundamental limits of classical computation. If technology continued to abide by Moore's Law, the continually shrinking size of circuits on silicon chips would eventually reach a point where individual elements would be no larger than a few atoms. Here at the atomic scale, quantum mechanical, instead of classical, governs the behavior and properties of the circuit. This then raised the question of whether a new kind of computer could be devised based on the principles of quantum physics.

Feynman produced an abstract model in 1982 that showed how a quantum system which used quantum mechanics intrinsically could be used to do computations and might be more powerful than a classical computer. Later, in 1985, Deutsch realized that Feynman's assertion could eventually lead to a general purpose quantum computer and published a crucial theoretical paper showing that any physical process, in principle, could be modelled perfectly by a quantum computer. Thus, a quantum computer would have capabilities far beyond those of any traditional classical computer. Since then, quantum computing has attracted

lots of scientists to study on different directions of this area: quantum algorithms, quantum error correction, quantum encryptions, physical implementations, and so on. Many creative and exciting developments were achieved and quantum computation shows more and more potentials. Up to date, it is widely believed that if large-scale quantum computer could be built, it would be able to solve certain problems faster than any classical computer, which makes it a possible candidate for the next generation of high-performance computer.

Although scientists have explored quantum computing and quantum information for decades, progress in this research area is still far behind what people initially expected. The problems and applications which gain significant improvements over their classical counterparts are still very limited. Experimental physicists have proposed and explored many possible physical implementations and hardware architectures of quantum computers, but a real breakthrough for a large scale one has not been achieved in any of them. Decoherence, the alteration of the quantum state as a result of the interaction with the environment, still represents a major problem for the practical realization of quantum computers. Quantum computing is still in its infancy in both theoretical and practical areas.

Up to now, quantum computers and quantum information technology still remains in its pioneering stage. Scientists from different areas, such as physics, computer science, mathematics, are preparing and providing the knowledge needed to thrust quantum computers up to their rightful position as the fastest computational devices in existence. We can say, once it matures, it may make today's modern computer out of date.

This dissertation is organized as follows: in chapter 2, some basic concepts and related works are introduced; chapter 3 gives an overview of grover algorithms, and chapter 4 describes Grover-type algorithm and gives an $R||C_{max}$ scheduling algorithm as an example in detail; chapter 5 addresses quantum error corrections; chapter 6 presents a quantum error correction code for time-correlated errors; chapter 7, the conclusions.

CHAPTER 2
BASIC CONCEPTS AND RELATED WORKS

2.1 Quantum Computing and Quantum Information Theory

Moore's law has been holding true throughout the last decades of the twentieth century. Nevertheless, it seems obvious that an exponential increase of the number of components on a solid state chip will eventually hit a "wall" and this dream will eventually end sometimes during the first two decades of the twenty-first century. Conventional approaches to the fabrication of computer technology are running up against fundamental difficulties of size. Quantum effects are beginning to interfere in the functioning of electronic devices as they become smaller and smaller. Another physical limitation is the heat removal for a circuit with densely packed logic gates. The amount of heat generated per volume increases with the calculated power of the circuit, while the ability of remove heat is proportional to the the surface area.

One possible solution to the problem posed by the eventual failure of Moore's law is to move to a different computing paradigm. One promising solution is provided by quantum computation, based on the idea of using quantum particles which obey the rules of quantum mechanics, instead of classic physics. In the early 1980s, Feynman introduced this idea that quantum computers, which used quantum mechanics intrinsically, might be more powerful

than a computer mired in the classical world. Later, in 1985, Deutsch realized that Feynman's assertion could eventually lead to a general purpose quantum computer and published a crucial theoretical paper showing that any physical process, in principle, could be modelled perfectly by a quantum computer. Thus, a quantum computer would have capabilities far beyond those of any traditional classical computer. After Deutsch published this paper, researchers began to find interesting applications for such a machine.

In 1994, Peter Shor found a polynomial time algorithm for the factorization of n-bit numbers on quantum computers [Sho94]. His discovery generated a wave of enthusiasm for quantum computing, for two major reasons: the intrinsic intellectual beauty of the algorithm and the fact that efficient integer factorization is a very important practical problem. The security of widely used cryptographic protocols is based upon the conjectured difficulty of the factorization of large integers.

In 1996, Grover described a quantum algorithm for searching an unsorted database containing N items in a time of order $\sqrt{N}$ while on a classical computer the search requires a time of order N [Gro96]. The critical aspect of a quantum algorithm is to create a superposition of all possible states and amplify the "solution". The speedup of Grover's algorithm is achieved by exploiting both quantum parallelism and the fact that in quantum theory a probability is the square of a propability amplitude. Bennett and co-workers [BBB97] and Zalka [Zal99] showed that Grover's algorithm is optimal. No classical or quantum algorithm can solve this problem faster than a time of order $\sqrt{N}$.

All these achievements continuously showed us the computational potential of quantum computer over classic computer.

Quantum information theory, like classical information theory, deals with information which can be transmitted by using quantum systems as carriers. Modern classical computers guard against error largely by being digital instead of analog. At each time step the hardware corrects the bit to standard 0 and 1 (digital instead of analog). This prevents small errors from building up into large errors, which are therefore drastically reduced. The same technique cannot be used in a quantum computer, because measurement on a single qubit would destroy the entanglements that distinguish a quantum computer from a classical one. Unlike the classic digital circuits, quantum operations are fully linear and reversible and therefore will quickly accumulate noise. The no-cloning property makes it more difficult to implement a reliable quantum communication.

Decoherence, the alteration of the quantum state as a result of the interaction with the environment, is probably the most challenging problem faced by quantum computation and quantum communication [Pre98]. *Quantum error correcting codes*, like classical error correcting codes, allow us to deal algorithmically with decoherence. Over time, researchers have come up with several codes: Peter Shor's 9-qubit-code, also named the *Shor code*, encodes 1 logical qubit in 9 physical qubits and can correct for one bit flip and one phase flip error; Steane implemented the same code using 7 instead 9 qubits, named *Steane code*; In 1996, Robert Calderband, Peter Shor and Andrew Steane discovered an important class

of quantum codes known as *CSS code*. A more general class of codes are the *Stabilizer codes* of Daniel Gottesman. By building upon the basic ideas of classical linear coding theory, these discoveries greatly facilitated a rapid understanding of quantum error correcting and its applications.

Quantum teleportation is used merely as a technical term for "transmission of quantum information on a classical channel". Teleportation is impossible with purely classical means; that is an important observation emphasizing the difference between quantum information and classical information theory. By using the quantum entanglement of an auxiliary pair of systems, a classical channel may be upgraded to do precisely this "impossible task". This new process, first published in 1993 by Bennett [BBC93], was experimentally realized in 1997 [BPM97].

Quantum cryptography uses quantum mechanics for secure communications. In traditional cryptography, various mathematical techniques are employed to restrict eavesdroppers from learning the contents of encrypted information. In quantum cryptography, quantum phenomena such as quantum entanglement is employed to carry information. Thus eavesdropping can be viewed as measurements which will disturb the system and therefore leave traces. Typical implementations for quantum key distribution include polarized photons [BBB92] and entangled photons [Eke91]. Some quantum key distribution protocols have been implemented and the associated devices seem to be close to commercialization [SGG02].

Different methods of implementing quantum computers have been proposed and many exciting achievements have been made, such as using photons, trapped ions or neutral atoms, cavity QED and nuclear magnetic resonance. At this time it is very difficult to assess in full the relative merits and potential of each of these approaches. Thus no one can tell which one will be the "winning" technology in the coming future.

Up to date, small quantum computing devices became reality, quantum cryptography has been demonstrated and is even at the level of real-world applications. However, some potentially large obstacles still remain that prevent us from making full operational quantum computers. Among these difficulties, error correction, decoherence, and hardware architecture are probably the most formidable. The experimental physics involved is very demanding. It still remains a great challenge to physicists and computer engineers to develop such full quantum computers in a near future.

2.2 Basic Concepts in Quantum Mechanics

Although classical computers have become more compact and considerably faster in performing their task, the task remains the same: to manipulate and interpret an encoding of binary bits into a desired computational result. In classical computer, a bit is the fundamental unit of information, 0 or 1. In a quantum computer, the fundamental unit of information (called a quantum bit or qubit) is not binary but rather more n-ary in nature. A qubit can exist in a state corresponding to the logical state 0 or 1 as in a classical bit, or it can be

in states corresponding to a blend or superposition of these states. In other words, a qubit can exist as a zero, a one, or simultaneously as both 0 and 1, with a coefficient representing the probability amplitude for each state. Actually, a qubit is a 2-dimensional, normalized vector in a Hilbert space with base vectors $\mid 0\rangle$ and $\mid 1\rangle$. An arbitrary single-qubit state can be represented as:

$$\mid \psi\rangle = \alpha \mid 0\rangle + \beta \mid 1\rangle$$

This is a normalized state where the probability amplitudes α and β are complex numbers, with $|\alpha|^2 + |\beta|^2 = 1$. A classical computer with n bits has 2^n possible states and this is an n-dimensional vector space over Z_2.

The inner product of two state vectors represents the generalized "angle" between the states and gives an estimate of the overlap between the states $\mid \psi_a\rangle$ and $\mid \psi_b\rangle$. The inner product $\langle \psi_a \mid \psi_b\rangle = 0$ defines orthogonal states; the implication of $\langle \psi_a \mid \psi_b\rangle = 1$ is that $\mid \psi_a\rangle$ and $\mid \psi_b\rangle$ are the same state.

A quantum computer with n qubits is a state in a 2^n-dimensional complex vector space. It can be represented as the tensor product "$\otimes$" of the respective state spaces of all the individual qubits. We abbreviate: $\mid 0\rangle \otimes \mid 1\rangle = \mid 01\rangle$, etc. The tensor product of two qubits over the bases $\{\mid 0\rangle, \mid 1\rangle\}$ can be expressed with the four basic vectors $\mid 00\rangle$, $\mid 01\rangle$, $\mid 10\rangle$,

$|\ 11\rangle$. For example,

$$|\ 0\rangle\otimes|\ 1\rangle = \begin{pmatrix} 1 \\ \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ 1 \\ \\ 0 \\ \\ 0 \end{pmatrix} = 0\ |\ 00\rangle + 1\ |\ 01\rangle + 0\ |\ 10\rangle + 0\ |\ 11\rangle$$

A quantum register of n qubits has the associated Hilbert space $H_2 \otimes H_2 \otimes \cdots \otimes H_2 = H_{2^n}$

and is of the form

$$\sum_{i=0}^{2^n-1} \alpha_i\ |\ i\rangle$$

with $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$

Here, in Hilbert space H_{2^n} of dimension 2^n, the orthogonal basis $\{|\ i\rangle\}$

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

is defined as the *computational basis*.

The Pauli matrices: $G = \{\sigma_I, \sigma_x, \sigma_y, \sigma_z\}$ describe transformations of interest of one qubit state:

$$\sigma_I = \begin{pmatrix} 1 & 0 \\ \\ 0 & 1 \end{pmatrix} \qquad \sigma_x = \begin{pmatrix} 0 & 1 \\ \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & \text{-i} \\ \\ \text{i} & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ \\ 0 & \text{-1} \end{pmatrix}$$

An error, like any other physical process, is a unitary transformation of the state space. The space of errors to a single qubit is spanned by the four Pauli matrices.

Unlike classical bits, the state of some multiple qubits may not be written as a product of single-qubit states. Such a state is called *entangled state*. For example, the state

$$\frac{1}{\sqrt{2}}(\mid 01\rangle + \mid 10\rangle)$$

cannot be decomposed into a product of two single-qubit states. Entanglement is a basic concept in quantum computing and it plays a major role in many quantum algorithms and quantum error-correcting codes.

One fact about quantum states that has profound implications for quantum computation is *the No Cloning Theorem*. It is a result of quantum mechanics which forbids the creation of identical copies of an arbitrary unknown quantum state. The proof is straightforward:

Suppose the state of a quantum system A, which we wish to copy, is $\mid \psi\rangle$. In order to make a copy, we take a system B with the same state space and initial state $\mid e\rangle$. The initial, or blank, state must be independent of $\mid \psi\rangle$, of which we have no prior knowledge. The composite system is then described by the tensor product, and its state is $\mid \psi\rangle \mid e\rangle$. Then the "copier" operation U provides:

$$U \mid \psi e\rangle = \mid \psi\psi\rangle$$

Suppose the state we want to copy is $\mid \psi\rangle + \mid \phi\rangle$, it follows that,

$$U(\mid \psi\rangle + \mid \phi\rangle) \mid e\rangle = U \mid \psi e\rangle + U \mid \phi e\rangle = \mid \psi\psi\rangle + \mid \phi\phi\rangle$$

However,

$$\mid \psi\psi\rangle + \mid \phi\phi\rangle \neq (\mid \psi\rangle + \mid \phi\rangle) \otimes (\mid \psi\rangle + \mid \phi\rangle)$$

So, the operation U failed to copy $\mid \psi\rangle + \mid \phi\rangle$. This theorem means that, to protect quantum information from errors, we can not simply make backup copies of the original quantum states.

The process of extracting information from a set of qubits is called *quantum measurement*. We describe quantum measurements of a single qubit in the state $\alpha \mid 0\rangle + \beta \mid 1\rangle$ as yielding the results 0 or 1 and leaving the qubit in the corresponding states $\mid 0\rangle$ or $\mid 1\rangle$, with respective probabilities $|\alpha|^2$ and $|\beta|^2$.

Quantum measurements are subject to Born's rule. This fundamental postulate of quantum mechanics is typically formulated as:

Consider a system in state $\mid \psi\rangle$ *and an observable* A *with eigenvectors* $\{ \mid \alpha_i\rangle \}$ *and eigenvalues* $\{ a_i \}$. *Then the probability for a measurement of the observable* A *to yield a particular value* a_i *is:*

$$p(a_i) = \mid \langle \alpha_i \mid \psi \rangle \mid^2 .$$

The state then collapses into $\mid \alpha_i\rangle$ after the measurement.

Figure 2.1 (a) illustrates the application of the Born rule for the measurement of the state of one qubit. In Figure 2.1 (b) we illustrate the generalized Born rule; we use a one-qubit measurement gate to measure only the first qubit of a register of n qubits in entangled state $\mid \psi^n\rangle = \alpha_0 \mid 0\rangle \mid \psi_0^{n-1}\rangle + \alpha_1 \mid 1\rangle \mid \psi_1^{n-1}\rangle$. Based upon the result of the measurement we decide that the remaining $(n-1)$ qubits are either in state $\mid \psi_0^{n-1}\rangle$ or in $\mid \psi_1^{n-1}\rangle$.

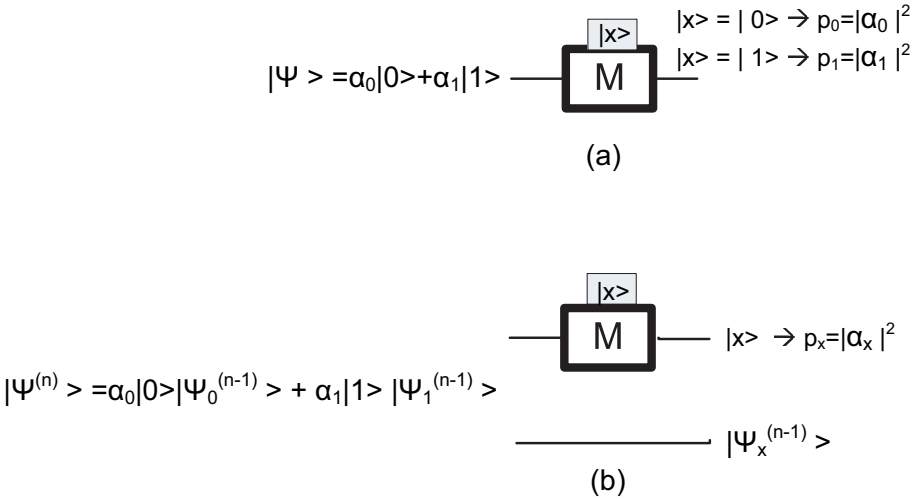

Figure 2.1: (a) The measurement of a single qubit in state $\mid \psi\rangle = \alpha_0 \mid 0\rangle + \alpha_1 \mid 1\rangle$ using a one-qubit measurement gate. (b) Use a one-qubit measurement gate to measure only the first qubit of a register of n qubits

We can also apply the generalized Born rule to measure the first k qubits of an n-qubit register originally in state

$$\mid \psi^n\rangle = \sum_{i=0}^{2^k-1} \alpha_i \mid i\rangle \otimes \mid \psi_i^{n-k}\rangle.$$

When the result of the measurement performed on the first k qubits is $x = r$ then we know that the remaining $(n-k)$ qubits are in state $\mid \psi_r^{n-k}\rangle$.

2.3 Quantum Circuits

In the classical world, a logic circuit is composed of *wires* and *gates*. Wires carry information to and from gates. Gates perform simple computational operations. For example, the NOT gate will flip the input bit, taking 0 to 1 and vice versa.

A *logic gate* computes a Boolean function $f : \{0,1\}^k \rightarrow \{0,1\}^l$ from a fixed number of input bits to some fixed number of output bits. For example, the NOT gate is a gate with one input bit and one output bit, which takes the function $f(a) = 1 \oplus a$, where $\oplus$ denotes addition modulo 2. There are many other elementary logic gates such as the OR gate, the AND gate, the XOR gate and the NAND gate. These gates can be interconnected to realize an enormous variety of functions $f : \{0,1\}^k \rightarrow \{0,1\}^l$. A circuit specifies the input bits and the output bits and how the gates are connected among each other. Any Boolean function can be implemented by such elemental gates. A classical circuit has been proven to be equivalent to the Turing model.

In quantum mechanics, a quantum circuit is a specific model for a quantum computational device. In this model, a quantum computer is built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate the quantum information. Experiments have already been carried out which can be regarded as implementing a seven-qubit quantum computer that implements Shor's factorization algorithm. Quantum circuits are also theoretically interesting as a tool for understanding the power and limitations of quantum computation.

Ordinarily, the logic gates in a classical computer, other than the NOT gate are not reversible. For example, for an AND gate one cannot generally recover the two input bits from the output bit. If a logic gate is irreversible, then some of the input information is irretrievably erased after the execution of the gate. **Landauer's principle** provides the connection between energy consumption and irreversibility [PV01]. It states that the erasure of information is inevitably accompanied by the generation of heat. This becomes a big obstacle for the further improvement of classical computers since the amount of energy dissipated keeps increasing as the computer power increases.

However, as a first step in describing a quantum circuit, it is instructive to observe that reversible gates are theoretically possible; moreover, these are actually of practical interest, since they do not increase the entropy.

A quantum gate executes a quantum operation on quantum information. A *quantum operation* can also be defined as a rotation of the state $\mid \psi\rangle$ in the N-dimensional Hilbert space. It is also convenient to represent a quantum gate (operation) in matrix form in Hilbert space. Any quantum operation is reversible, linear and preserves the norm. In fact a linear transformation is a valid quantum operation if and only if it is unitary.

An operator U is unitary if

$$U^\dagger U = UU^\dagger = I \qquad \text{with } U^\dagger \text{ the adjacent of } U$$

Operation U_1 followed by U_2 is equivalent to

$$U_3 = U_2 U_1$$

If U_1 acts on Hilbert space H_A and U_2 acts on Hilbert space H_B, then

$$U_3 = U_2 \otimes U_1$$

acts on the composite Hilbert space $H_{AB} = H_A \otimes H_B$

Some elemental quantum gates are introduced as follows:

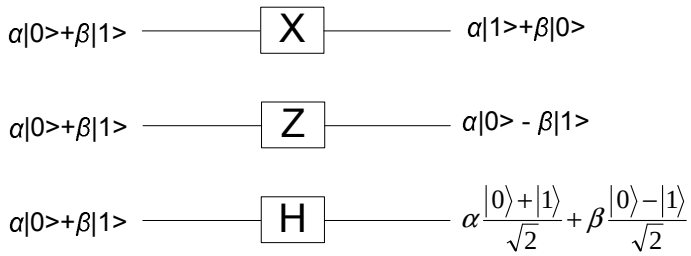

Figure 2.2: Circuit representations of single qubit logic gates

- NOT gate

  The quantum NOT gate acts linearly. It takes the state $\alpha \mid 0\rangle + \beta \mid 1\rangle$ to the corresponding state $\alpha \mid 1\rangle + \beta \mid 0\rangle$ in which the vectors of $\mid 0\rangle$ and $\mid 1\rangle$ have been flipped. The quantum NOT gate can be represented as:

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

When it acts on a general qubit, the output from the quantum NOT gate is:

$$X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

- Z gate

  The Z gate is another single qubit gate as shown in Figure 2.2. It leaves $\mid 0\rangle$ unchanged, and flips the sign of $\mid 1\rangle$. It can be represented in matrix form as:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & \text{-1} \end{pmatrix} \qquad Z \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}$$

- *Hadamard* gate

  The *Hadamard* gate is sometimes described as a "square-root of NOT" gate, in that it turns $\mid 0\rangle$ into $(\mid 0\rangle + \mid 1\rangle)/\sqrt{2}$, "halfway" between $\mid 0\rangle$ and $\mid 1\rangle$, and turns $\mid 1\rangle$ into $(\mid 0\rangle - \mid 1\rangle)/\sqrt{2}$, which is also "halfway" between $\mid 0\rangle$ and $\mid 1\rangle$. The *Hadamard* gate is considered as one of the most useful quantum gates.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & \text{-1} \end{pmatrix} \qquad H \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix} = \alpha \frac{\mid 0\rangle + \mid 1\rangle}{\sqrt{2}} + \beta \frac{\mid 0\rangle - \mid 1\rangle}{\sqrt{2}}$$

- CNOT gate

  The typical multi-qubit quantum logic gate is the *controlled*-NOT or CNOT gate. This gate has two input qubits: the *control* qubit and the *target* qubit. Its circuit

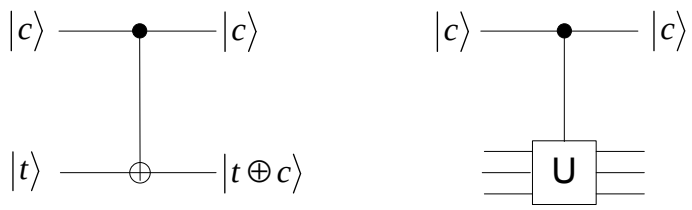

Figure 2.3: Circuit representations of *control*-NOT and *control*-U gate

representation is shown in Figure 2.3. The top line represents the control qubit while the bottom line represents the target qubit. This gate leaves the target qubit unchanged if the control qubit is set to $|\ 0\rangle$, and flips the target qubit if the control qubit is set to $|\ 1\rangle$. That corresponds to the state transformations:

$$|\ 00\rangle \rightarrow |\ 00\rangle; \qquad |\ 01\rangle \rightarrow |\ 01\rangle; \qquad |\ 10\rangle \rightarrow |\ 11\rangle; \qquad |\ 11\rangle \rightarrow |\ 10\rangle.$$

The gate's matrix representation is:

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

The function of a CNOT gate can also be summarized as $|\ c, t\rangle \rightarrow |\ c, c \oplus t\rangle$, where $\oplus$ denotes addition modulo two.

- *Controlled*-U gate

The *controlled*-U gate is an extension of the *controlled*-NOT gate. Suppose U is any unitary matrix acting on some number n of qubits which can also be regarded as a quantum operation on these n qubits. The *controlled*-U gate has one *control qubit* and n *target qubits*. If the control qubit is set to $|\,0\rangle$, the target qubit is left untouched. If the control qubit is set to $|\,1\rangle$ then the gate U is applied to the n target qubits.

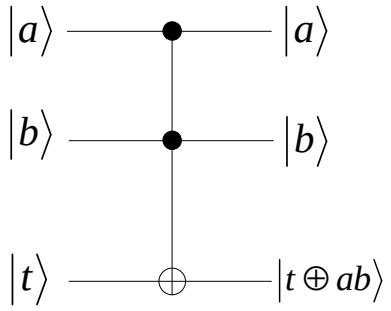

Figure 2.4: Circuit representation of Toffoli gate

- Tofolli gate

  The quantum Tofolli gate acts on 3 qubits as shown in Figure 2.4. Two control qubits are unaffected by the action of the Tofolli gate. The third qubit is the target qubit that is flipped if both control qubits are set to $|\,1\rangle$ and otherwise is left unchanged.

A small set of classical gates (e.g. AND, OR, NOT) can be used to compute any arbitrary classical function. We say that such a set is universal for classical computation. Similarly, single qubit gates and CNOT gate are universal for quantum computation [NC00]. Any unitary operation on n qubits can be implemented by a combination of single qubit gates and CNOT gates.

2.4 Physical Implementations of Quantum Computer

Implementations of quantum computers will be a difficult experimental challenge. Several physical implementations were proposed and at this time it is very difficult to assess in full the relative merits and potential of each of these approaches. Significant progress is reported in each of these areas, but a real breakthrough has not been achieved in any of them.

David DiVincenzo(IBM) criteria gave the five (plus two) essential characteristics of the physical implementations of quantum computation and information processing [Div00]:

- *A physical system has well characterized qubits and is scalable.* The embodiment of a qubit is a quantum two-level system, such as the two spin states of a spin 1/2 particle, or the ground and some excited state of an atom, or the vertical and horizontal polarization of a single photon. A "well characterized" qubit should have accurately known physical parameters, including: the internal Hamiltonian (which determines the energy eigenstates of the qubit); the presence of and couplings to other states of the qubit; the interactions with other qubits; and the couplings to external fields that might be used to manipulate the qubit. If the qubit has a third, fourth, and other higher energy levels, the probability of the transitions to such states from the characteristic states must be very low and under the control of the physical system.

- *The ability to initialize the state of the qubits to a simple state, such as* $|\,000...\rangle$. As in classical computing, the registers must be initialized to a known value before the start of

the computation. Furthermore, quantum error correction requires a continuous, fresh supply of qubits in a low-entropy state, such as the $|\ 0\rangle$ state. The time it takes to initialize (to "zero") a qubit is very important; if this time is relatively long compared with the gate-operation time, the quantum computer will have to be provided with a "qubit conveyor belt" which will take away "used qubits", initialize them, and bring them back into the quantum computation process. There are two main approaches to setting qubits to a standard state. A quantum system can be cooled either by natural cooling (when the ground state of the qubit Hamiltonian is the state of interest), or by projecting the system into the state of interest through a relevant measurement. The cooling times by projection could be much shorter than by natural relaxation.

- *Long relevant decoherence times which should be much longer than the gate operation time.* The decoherence of a quantum system is due to its interactions with the environment and represents the principal mechanism for the emergence of classical behavior. Decoherence times characterize the dynamics of a qubit (or any quantum system) in contact with its environment. The decoherence times relevant for a quantum computer are associated with the degrees of freedom that characterize the embodiment of the qubits. The decoherence time must be long enough to allow the quantum system to evolve for as long as the completion of the computation requires.

- *A "universal" set of quantum gates.* A set of quantum gates is "universal" in the sense that a network of them is capable of implementing any unitary operation on

an arbitrary number of qubits. A quantum algorithm is typically specified as a sequence of unitary transformations U_1, U_2, U_3, ..., each acting on a small number of qubits, typically no more than three. In principle, the first step towards a physical implementation of the algorithm is identifying the Hamiltonians which generate these unitary transformations, i.e., $U_1 = e^{iH_1 t/\hbar}$, $U_2 = e^{iH_2 t/\hbar}$, $U_3 = e^{iH_3 t/\hbar}$. The next step is designing the physical apparatus so that the interaction Hamiltonians H_1, H_2, H_3, ..., are successively turned on at precise moments of time and for predetermined lengths of time. In reality, only some types of Hamiltonians can be turned on and off as requested by a quantum computation. Most of the physical implementations proposed so far consider only two-body interactions. However, even two-qubit interactions are not easily achievable.

- *The physical system must have a qubit-specific measurement capability.* The result of a computation is read out by measuring specific qubits. If an ideal measurement is performed on a qubit with density matrix:

  $\rho = p \mid 0\rangle\langle 0 \mid + (1-p) \mid 1\rangle\langle 1 \mid + \alpha \mid 0\rangle\langle 1 \mid + \alpha^* \mid 1\rangle\langle 0 \mid,$

  it should give outcome "0" with probability p and "1" with probability $1-p$ independent of α, the state of neighboring qubits, or any other parameters of the system and the state of the rest of the computer is not changed. Actually, realistic measurements are expected to have quantum efficiencies less, even much less than 100%. The low quantum efficiency measurement could be circumvented either by rerunning the com-

putation for a number of times, or by "copying" the value of a single output qubit to several qubits using CNOT gates and measuring each of them.

A physical system which is also involved in quantum communication besides quantum computation, must satisfy two additional requirements relevant for the task of transmitting intact qubits from one place to another:

- *The system must have the ability to interconvert stationary and flying qubits.* The term "flying qubits" emphasizes that the optimal embodiment of qubits that are readily transmitted from place to place is likely to be very different from the optimal qubits for reliable local computation. The photon states with the qubit encoded either in the polarization or in the spatial wave function of the photon are considered as the flying qubit of choice at this time. The light transmission through optical fibers is a well developed technology and reliable enough for the transmission of qubits. A current difficult aspect is the downloading of the qubits from the quantum computer into a travelling mode into the transmission system and the uploading process.

- *The flying qubits must be faithfully transmitted between specified locations.* Up to date, experiments on quantum cryptography have been very concerned with the preservation of the photon quantum state during transmission through optical fibers or through the atmosphere.

Up to date, there have been several serious proposals for quantum computer physical implementation and they can be classified into six basic categories:

1. **Nuclear Magnetic Resonance (NMR)**: The *nuclear magnetic resonance* (NMR) was proposed by David Cory [CFH96] and by Neil Gershenfeld and Isaac Cuang [GC97] as a working concept and a very attractive candidate for a quantum computer due to the extremely long coherence times associated with the nuclear spins. A qubit can be implemented using any quantum mechanical system with two basis states. In NMR computers the two spin states of a spin-1/2 atomic nucleus in a magnetic field represent the basis states of a qubit. The qubits are atomic nuclei within the same molecule. Different atomic nuclei in a molecule can be distinguished and manipulated using the nuclear magnetic resonance technique. So a molecule can be used as a quantum computer. Simple logic gates which only affect a single qubit are easily implemented using radio frequency fields. These interact strongly with nuclear spins, allowing them to be controlled with great precision. Some quantum algorithms, such as a simple version of Grover's quantum search algorithm ($N \leq 10$), have been implemented on NMR quantum computers.

2. **Ion Traps**: The ion trap is the first technology proposed for quantum information processing devices [CZ95]. Ions, or charged atomic particles, can be confined and suspended in free space using electromagnetic fields. Qubits are stored in stable electronic states of each ion, and quantum information is processed and transferred through the

collective quantized motion of the ions in the trap. Lasers are used to induce coupling between the qubit states (for single qubit operations) or coupling between the internal and the external states (for entanglement between qubits). Many fundamental quantum operations have been demonstrated experimentally with high accuracy in trapped ion systems, and a strategy has been developed for scaling the system to arbitrarily large number of qubits by shuttling ions in an array of ion traps. This makes the trapped ion system one of the most promising architectures for a scalable, universal quantum information processor.

3. **Optical cavity quantum electrodynamics**(CQED): CQED involves coupling of single atoms to only a few optical modes of the cavity. It is implemented by placing single atoms within an optical cavity with very high Q. Because of the high Q, photons inside the cavity have an opportunity to interact many time with the atoms before escaping. In this approach, the Hamiltonian describes atom, photon, and atom-photon interactions. The qubit is the single photon. A QED cavity computer was demonstrated by Turchette, Hood, Lange, Mabuchi, and Kimble in 1995 [THL95]. Gates for $\pi/8$ and Hadamard are implemented by beam splitter and the like. A 2-qubit quantum XOR gate was implemented by Kimble's group.

4. **Superconductive quantum interference devices** (SQUIDs): The SQUID, or superconductive quantum interference device, is a highly sensitive instrument employed for non-destructive measurement of magnetic fields, with a host of applications in both

biophysics and materials technology. It is composed of a cooled superconductive metal ring separated by a thin insulating barrier from a non-superconducting metal, forming a Josephson junction. It shows potential for quantum computing applications by forming the qubit component of a quantum computer, through simply treating the direction of the persistent current – clockwise or counterclockwise – as the value of the bit. One advantage of these qubits is the ability to precisely engineer the Hamiltonian of the system, which includes single qubit design, multiple qubit design, and measurement design. Another advantage of SQUIDs is that they can be fabricated en masse on a chip. Large-scale integration is quite conceivable

5. **Quantum dots**: Quantum dots are atom-like compound semiconductors structures where charge carriers are confined in all three spatial dimensions. A semiconductor quantum dot may contain from a few thousand to tens of thousands of atoms arranged in a nearly defect-free three-dimensional crystal lattice. The carriers in the quantum dot interact strongly with lattice vibrations and could be influenced by defects, surfaces, or interfaces. The quantum dots have unusual properties which recommend them for special scientific and technological applications; these unusual properties are due to confinement of electric motion and strong sensitivity to external fields. For self-assembled quantum dots, some of the energy levels are forbidden to electrons and this energy region is called the *bandgap*. In bulk semiconductor material, the overwhelming majority of the electrons stay at the energy levels below the bandgap; Only a minuscule

percentage of the semiconductor electrons may by found in the energy levels above the bandgap. Electrons below the bandgap could be stimulated to jump above the bandgap only if given enough energy through heat, voltage, or photon flux.

Quantum dot systems are considered viable candidates for a quantum computer implementation. The qubit can be defined as a two-level system, the ground state and the excited state of a single-electron quantum dot controlled by optical resonant pulses [BDE95], or as the spin of the excess electron in a single-electron quantum dot [LD98], or as the spin states of two conduction electrons optically excited from the valence band in the presence of a magnetic field applied perpendicular to the confinement direction of the quantum dot [IAB99], or as the localization of the excess electron charge in one or the other of two coupled single-electron quantum dots [Tan00].

6. **Topological quantum computer**: Topological QC is a new concept quantum computer which uses quasiparticles called *anyons* where their world lines form threads that cross over one another to form braids in a two-dimensional world. These braids form the logic gates of a computer. The advantage of a topological QC using quantum braids over using other quantum particles is that the former is much more stable. The smallest perturbations that can cause a quantum particle to decohere, and create errors in the computation do not change the topological properties of the braids. It was proved in 2002 that a Topological quantum computer can, in principle, perform any computation that a trapped quantum particle type quantum computer can do [FKL03].

In a key development for Topological quantum computers, in 2005 Vladimir J. Goldman, Fernando E. Camino and Wei Zhou claimed to have direct experimental confirmation that quasiparticles occurring in the fractional quantum Hall state are anyons, a crucial first step in the implementation of a Topological quantum computer.

What is the "winning" technology going to be? Nobody can give the answer up to date. Even though we have lived with quantum mechanics for a century, our studies of quantum effects in complex artificial systems as the systems for quantum computing presented above are still in its infancy. The ideas of quantum information theory will stimulate more and more creative and exciting developments of complex quantum systems for the coming future.

2.5 Quantum Algorithms

By the early nineties it was known that a quantum computer may be faster than any classical computer for certain problems. Nonetheless these observations were largely driven by academic curiosity. There was not much motivation for people to spend lots of money or time trying to build a quantum computer.

This situation changed when Peter Shor devised a polynomial time algorithm for factoring large numbers on a quantum computer in 1994. This discovery drew great attention to the field of quantum computing because of the intrinsic intellectual beauty of the algorithm and the fact that efficient integer factorization is a very important practical problem.

Generally, current quantum algorithms which offer a significant speed-up over their classical counterparts can be divided into three broad categories [NC00, Sho03]:

1. Algorithms that find the periodicity of a function using Fourier Transforms, a tool which is also widely used in classical algorithms. The Deutsch-Jozsa algorithm, Simon's algorithm [Sim97], Shor's algorithms for factoring and for computing discrete logarithms [Sho97], and Hallgren's algorithm to solve the Pell's equation are all members of this class.

2. The quantum search algorithms is completely different class of algorithms which can perform an exhaustive search of N items in $\sqrt{N}$ time. Grover's algorithms [Gro96, Gro97, Gro98] belong to this class.

3. Algorithms for simulating quantum systems, as suggested by Feynman. This is a potentially large class of algorithms, but not many algorithms in this class have been developed so far. Once quantum computers become a reality we should expect the development of a considerable number of programs to simulate quantum systems of interest [MM04, NC00].

The algorithms used to simulate quantum mechanical systems follow the evolution in time of a fairly large number of quantum particles. The result of such a simulation reflects what the outcome of a measurement of the physical characteristic of the quantum system would reveal. Quantum Monte Carlo algorithms(quantum MC) are widely used today to simulate quantum physical systems. In Monte Carlo simulations, state probability amplitudes are represented by the expected values of random variables calculated during the simulation. Such simulations produce large statistical errors that can only be reduced by repeating the

calculations many times. The advantage of using quantum algorithms for simulations of quantum physical systems is that they allow us to determine the value of relevant physical properties with polynomial bounded statistical errors.

How powerful are quantum computers and quantum algorithms? Nobody can give an exact answer to such a problem yet. It is still doubtful that a quantum computer is more powerful than a classical computer. Let us take a brief look at what is known about the power of quantum computer and classical computer.

An algorithm can be characterized by the number of operations and amount of memory it requires to compute an answer given an input of a certain size . These characterizations of the algorithm determine what is called the *algorithm's complexity*. Specifically, the complexity of an algorithm is determined by looking at how the number of operations and memory usage required to complete the program scales with the input size of the program. Generally, computer scientists group problems into the following complexity classes:

- **P**: Polynomial time, the running time of the given algorithm is in the worst case some polynomial function of the input size.
- **NP**: Nondeterministic polynomial time, a candidate for an answer can be verified as a correct answer or not in polynomial time.
- **NP-complete**: the set of problems such that if any member of this set is in P, then the set P equals the set NP.

- **BPP**: the set of problems can be solved using randomized algorithms in polynomial time, if a bounded probability of error is allowed in the solution to the problem.

- **BQP**: the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances.

- **postBQP**: the complexity class consisting of all of the computational problems solvable in polynomial time on a quantum computer with postselection. [Aar05] *Postselection* is the power of discarding all runs of a computation in which a given event does not occur.

Problems which can be solved in polynomial time or less are generally deemed to be "tractable". Problems which require more than polynomial time are usually considered to be "intractable". **BPP** is widely regarded as being the class of problems which should be considered efficiently solvable on a classical computer.

BQP is defined as the class of problems which can be solved efficiently on a quantum computer, where a bounded probability of error is allowed. Exactly where BQP fits with P and NP is as yet unknown. It has been proved that $NP \subseteq postBQP = PP$ [Aar05], but $postBQP$ is not likely easy implemented in reality.

Thus what kind of problems should we focus on which could gain benefits from quantum computers? Peter Shor speculates [Sho03] that the number of such problems for which the quantum algorithms may offer a substantial speedup over the classical algorithms may be

very limited. To see the spectacular speedups we have to concentrate on problems not in the classical computational class P(although quantum algorithms may be faster). Many believe that quantum algorithms solving NP-complete problems in polynomial time do not exist, even though no proof of this assertion is available at this time. Shor argues that if we assume that no polynomial time quantum algorithms exist for solving NP-complete problems, then the class of problems we have to search for is neither NP-complete, nor P, and the population of this class is relatively small.

2.6 Classical vs Quantum Error Correction

Classical error correction operates by the judicious use of redundancy, that is, making the system larger in order to make it more resilient to perturbations. A good encoding should

1. make transmission resilient to errors,

2. reduce the amount of data transmitted through a channel, and

3. ensure information confidentiality.

The type of redundancy, or encoding, employed must be carefully chosen according to the type of noise it wants to fight. The simplest example is *the Repetition Code* which is given by the map $0 \rightarrow 000,\ 1 \rightarrow 111$. The information can be decoded with majority logic: If the majority of the three bits is 0, output 0, otherwise output 1.

In classical error correction, the Hamming distance $d(x, y)$ between two codewords x and y is defined as the number of coordinate positions in which they differ. The *Hamming distance* of a code is the minimum distance between any pairs of codewords.

Normally, in classical error correction, if we want to correct d bit errors in an n bit word we can map every n bit word into a bigger $n + q$ bit code word so that the minimum Hamming distance between each valid mapping is $2d + 1$. We define that each code's scope includes all $n + q$ bit words with distance $x \leq d$ to the code. This way, d or less errors will never transform a valid word into another valid word's scope, because the Hamming distance between each valid word is at least $2d + 1$, and such errors only lead to invalid words that are inside the code's scope and can be recovered correctly.

Linear code: Classical coding theory tends to concentrate on linear codes, a subclass of all possible codes with a particular relation between codewords. Suppose we wish to encode k bits using n bits. A linear code C is a one-to-one mapping f of k-tuples to n-tuples:

$$V_k(F) \rightarrow V_n(F)$$

Once we select a set of k linearly independent vectors in $V_n(F)$ we can construct a generator matrix G for C. Then the code C can be constructed by multiplying the *generator matrix* G with message vectors (vectors with k components, or k-tuples). Thus codewords in C are arbitrary linear combinations of the basis codewords of the generator matrix; therefore the name "linear code."

Example. Consider a 3-dimensional subspace of a vector space over the finite field with $q = 2$ (binary field) and $n = 6$. The generator matrix is

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

Let the message space be

$$M = \{(000), (001), (010), (011), (100), (101), (110), (111)\}$$

Then the codewords of linear code C can be constructed as $\{c_i = m_i G \quad m_i \in M\}$

Given a generator matrix G, we can calculate the dual matrix H, which is an $(n-k) \times n$ matrix of 0s and 1s with $GH^T = 0$. The matrix H is called the *parity check matrix* for the code since it annihilates with all the codewords. The *dual code* is defined to be the code with generator matrix H^T and parity check matrix G^T.

Although many classical error-correction techniques are heuristic, they cannot be directly used for quantum error correction. First of all, the classical techniques assume we can measure all of the bits in the computer; that will collapse the states in quantum computers. Secondly, a quantum computer needs to preserve not only 0 and 1, but also superposition or entangled states. Furthermore, no-cloning theorem prohibits the simple replication of a quantum state. At last, the errors we are facing in quantum error correction include not only bit flip error which flips $\mid 0\rangle \rightarrow \mid 1\rangle$ or $\mid 1\rangle \rightarrow \mid 0\rangle$, but also phase flip which changes

the phase $|\,1\rangle \rightarrow -\,|\,1\rangle$. All these things must be carefully considered when we construct quantum error-correcting codes.

Just like for a repetition code in classical error correction, we are able to encode a single logical qubit as multiple physical qubits and thus we can correct quantum errors [Ste96]. For example, we can encode the state of a qubit

$$|\,\psi\rangle = \alpha_0\,|\,0\rangle + \alpha_1\,|\,1\rangle$$

as a linear combination of $|\,00000\rangle$ and $|\,11111\rangle$:

$$|\,\varphi\rangle = \alpha_0\,|\,00000\rangle + \alpha_1\,|\,11111\rangle,$$

or as

$$|\,\varphi\rangle = \alpha_0\,|\,0_L\rangle + \alpha_1\,|\,1_L\rangle,$$

with $|\,0_L\rangle$ and $|\,1_L\rangle$ expressed as a superposition of codewords of a classical linear code.

When we use the above encoding scheme, a random error can cause departures from the subspace spanned by $|\,00000\rangle$ and $|\,11111\rangle$. We shall be able to correct small errors (e.g. 1 bit error) because the component which was $|\,00000\rangle$ is likely to remain in a sub-space $C_0 \subset H_{2^5}$ spanned by the six vectors, $|\,00000\rangle, |\,00001\rangle, |\,00010\rangle, |\,00100\rangle, |\,01000\rangle$, and $|\,10000\rangle$, while the component which was $|\,11111\rangle$ is likely to remain in a sub-space $C_1 \subset H_{2^5}$ spanned by the six vectors, $|\,11111\rangle, |\,11110\rangle, |\,11101\rangle, |\,11011\rangle, |\,10111\rangle$, and $|\,01111\rangle$. The two subspaces, C_0 and C_1 are disjoint. More quantum error codes will be introduced in Section 5.

CHAPTER 3
GROVER'S ALGORITHM

3.1 Grover's Searching Algorithms

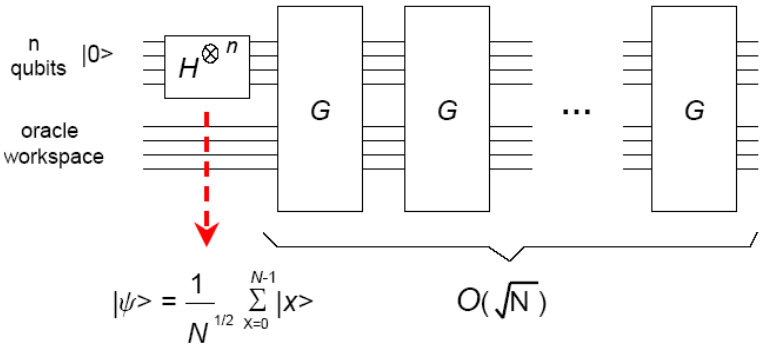

Figure 3.1: The steps required by Grover's algorithm.

Consider a search space $T_{search} = \{E_x\}$ consisting of $N = 2^n$ elements. Each element $E_x, 0 \leq x \leq 2^n - 1$, is uniquely identified by a binary n-tuple x, called *the index* of the element. We assume that $M \leq N$ elements satisfy the requirements of a query and we wish to identify one of them.

The classic approach is to repeatedly select an element E_j, decide if the element is a solution to the query, and if so, terminate the search. If there is a single solution ($M = 1$) then a classical exhaustive search algorithm requires $O(2^n)$ iterations.

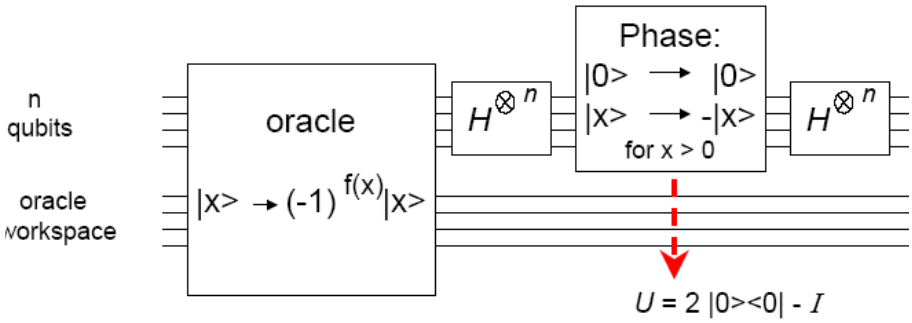

Figure 3.2: A quantum circuit for the Grover's iteration

The basic idea of Grover's quantum search algorithm is illustrated in Figure 3.1. We apply a Walsh-Hadamard transform to create an equal superposition state which includes all elements of the search space:

$$\mid \psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \mid x\rangle.$$

Then we perform Grover's iterations. The circuit for this algorithm is presented in Figure 3.2

To abstract this process we consider a function $f(x)$ with $0 \leq x \leq 2^n - 1$ such that

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is not a solution} \\ 1 & \text{if } x \text{ is a solution.} \end{cases}$$

Consider an oracle, a black box accepting as input n qubits representing the index x of an element $E_x, 0 \leq x \leq 2^n - 1$. The black box has also an *oracle qubit*, $\mid q\rangle$, initially set to $\mid 0\rangle$ and reset to $\mid 1\rangle$ when the oracle recognizes a solution to the search problem we pose. The

black box performs the following transformation

$$\mid x\rangle \mid q\rangle \quad \mapsto \quad \mid x\rangle \mid q \oplus f(x)\rangle.$$

The oracle qubit can be initially in the state

$$\mid q\rangle = (1/\sqrt{2})(\mid 0\rangle - \mid 1\rangle).$$

If an input $\mid x\rangle$ is not a solution to our search problem, then the oracle qubit is unchanged. On the other hand, a solution to the search problem transforms $\mid 0\rangle$ into $\mid 1\rangle$. Thus, the transformation performed by the black box is

$$\mid x\rangle\ (\mid 0\rangle - \mid 1\rangle)/\sqrt{2} \quad \mapsto \quad \begin{cases} \mid x\rangle\ (\mid 0\rangle - \mid 1\rangle)/\sqrt{2} & \text{if } f(x) = 0 \implies E_x \text{ is not a solution} \\ - \mid x\rangle\ (\mid 0\rangle - \mid 1\rangle)/\sqrt{2} & \text{if } f(x) = 1 \implies E_x \text{ is a solution.} \end{cases}$$

This transformation can be rewritten as

$$O \mid x\rangle\ (\mid 0\rangle - \mid 1\rangle)/\sqrt{2} = (-1)^{f(x)} \mid x\rangle\ (\mid 0\rangle - \mid 1\rangle)/\sqrt{2}.$$

The state of the oracle qubit does not change and can be omitted from the description of the quantum search algorithm

$$\mid x\rangle \quad \mapsto \quad (-1)^{f(x)} \mid x\rangle.$$

Let U be the following transformation in the n-qubit space:

$$U = 2 \mid 0\rangle\langle 0 \mid - I.$$

Then a conditional phase shift in Figure 3.2 applied to the system is:

$$S_p = H^{\otimes n} U H^{\otimes n} = H^{\otimes n}(2 \mid 0\rangle\langle 0 \mid -I)H^{\otimes n} = 2 \mid \psi\rangle\langle\psi \mid -I.$$

A Grover's iteration consists of O, the transformation performed by the oracle, followed by a conditional phase shift:

$$G = S_p O = (2 \mid \psi\rangle\langle\psi \mid -I)O$$

Thus, the quantum search algorithm could be written as:

$$G^R \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \mid x\rangle \mid q\rangle = [(2 \mid \psi\rangle\langle\psi \mid -I)O]^R \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \mid x\rangle(\mid 0\rangle - \mid 1\rangle)/\sqrt{2} \approx \mid x_0\rangle(\mid 0\rangle - \mid 1\rangle)/\sqrt{2}$$

When after $R = O(\sqrt{\frac{N}{M}})$ iterations, we measure the first n qubits we obtain x_0, the solution to the search problem.

3.2 Amplitude Amplification

Amplitude amplification represents a generalization of the Grover's quantum search idea [BHM02, Ho00]. Let A be a unitary operator in a Hilbert space, H_N with an orthonormal basis $\mid 0\rangle, \mid 1\rangle, \ldots \mid N-1\rangle$; the only condition imposed on A is to be invertible, thus A must not involve any measurements.

If $\chi : \{0, 1, \ldots, N-1\} \mapsto \{0, 1\}$ is a Boolean function we say that the basis state $\mid x\rangle$ is a "Good" state if $\chi(x) = 1$ and $\mid x\rangle$ is a "Bad" state" if $\chi(x) = 0$. The central piece of the

amplitude amplification is an operator Q defined as:

$$Q = Q(A, \chi, \phi, \varphi) = -AS_0(\phi)A^{-1}S_\chi(\varphi).$$

with ϕ and φ two angles such that $0 \leq \phi, \varphi \leq \pi$ and S_χ an operator which conditionally changes the amplitudes of "Good" states:

$$\mid x\rangle \mapsto \begin{cases} e^{i\varphi} \mid x\rangle & \text{if} \quad \chi(x) = 1 \\ \mid x\rangle & \text{if} \quad \chi(x) = 0. \end{cases}$$

Similarly, S_0 amplifies the amplitude by a factor $e^{i\phi}$ if the state is not $\mid 0\rangle$.

Let a denote the probability of finding a "Good" element x; amplitude amplification allows to find a "Good" x after an expected number of applications of A and of the inverse of A; the number of iterations is proportional to $1/\sqrt{a}$. We also define the angle θ such that:

$$\sin^2(\theta) = a.$$

Grover's algorithm is a particular instance of amplitude amplification when:

- the oracle implements the Boolean function $f = \chi$,
- the transformation A is the Walsh-Hadamard transform $W = H^{\otimes n}$ on n qubits, and
- the Grover's iteration $-WS_0WS_f$ corresponds to the operator $Q = -AS_0A^{-1}S_\chi$.

This iteration carried out by transformation Q can be regarded as a *rotation* in the two-dimensional space spanned by the state of a uniform superposition of non-solutions and the

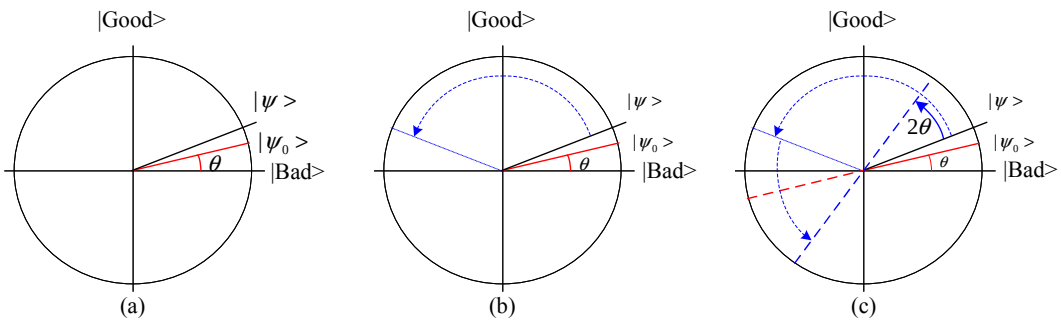

Figure 3.3: The search operator Q performs a rotation toward $\mid Good\rangle$ states by 2θ radians. (a) The current state $\mid \psi\rangle$ and the initial state $\mid \psi_0\rangle$. (b) The oracle operation S_χ performs a reflection of the current state $\mid \psi\rangle$ about the vector $\mid Good\rangle$. (c) AS_0A^{-1} performs a reflection about the initial state $\mid \psi_0\rangle$

state consisting of a uniform superposition of solutions to the search problem. The initial state may be expressed as:

$$\mid \psi_0\rangle = \sqrt{a} \mid Good\rangle + \sqrt{1-a} \mid Bad\rangle$$

Figure 3.3 presents the effect of the transformation $Q = -AS_0A^{-1}S_\chi$ as:

- the oracle operation S_χ performs a reflection about the vector $\mid Good\rangle$.

$$S_\chi \mid x\rangle = \mid x\rangle \quad (\chi(x) = 1) \qquad\qquad S_\chi \mid x\rangle = - \mid x\rangle \quad (\chi(x) = 0)$$

- AS_0A^{-1} performs a reflection about the initial state $\mid \psi_0\rangle$

$$S_0 \mid 0\rangle = \mid 0\rangle \qquad\qquad S_0 \mid x\rangle = - \mid x\rangle \quad (x \neq 0)$$

- $Q = -AS_0A^{-1}S_\chi$ performs a rotation toward $\mid Good\rangle$ vector by 2θ radians, where $\sin^2\theta = a$

Each iteration Q will rotate the system state by 2θ radians toward the solutions of the searching problem. Thus after m iterations, the measurement on the final state $Q^m \mid \psi_0\rangle$ will produce a "Good" state with probability equal to $\sin^2((2m+1)\theta)$. This algorithm **Qsearch** finds a good solution using an expected number of applications of A and A^{-1} which are in $O(\frac{1}{\sqrt{a}})$. [BHM02, Ho00]

3.3 Extension of Grover's Algorithm

Grover's algorithm can find a solution in N items, requiring only $O(N)$ operations. After that, many special extensions of Grover's algorithm were developed to address different applications.

1. **Fix-point Search:** Grover's algorithm is able to find a target state in an unsorted database of size N in only $O(\sqrt{N})$ queries. It is achieved by designing the iterative transformations in a way that each iteration results in a small rotation of the state vector in a two-dimensional Hilbert space that includes the target state. If the number of iterative steps is perfectly chosen, the process will stop just at the target state. Otherwise, it may overshoot the target. By replacing the selective inversions by selective phase shifts of $\pi/3$, the algorithm preferentially converges to the target state irrespective of the step size or number of iterations. [Gro05] This feature leads to robust search algorithms and also to new schemes for quantum control and error correction [RG05]. A different approach can be found in [TGP06].

2. **Partial Quantum Search:** Partial Quantum Search considers the search problem where given a quantum database $f : \{0,1\}^n \to \{0,1\}$ such that $f(x) = 1$ for a unique $x \in \{0,1\}^n$, we are required to determine only the first k bits of the address x. The quantum algorithms and low bound can be found in [GR05, KG06].

3. **Quantum Counting:** To determine the number of solutions, M, to an N item search problem, it takes $O(N)$ consultations on a classical computer. On a quantum computer it is possible to estimate the number of solutions much more quickly by combining the Grover's algorithm with the phase estimation based on quantum Fourier transform. [BHT98]

4. **Quantum Walk:** Quantum random walks on graphs have been shown to display many interesting properties, including exponentially fast hitting times when compared with their classical counterparts [AAK01, Kem03]. It also provides an intuitive framework on which to build novel quantum algorithms which run faster than their classical counterparts. It can be divided into two models: discrete time quantum random walk and continuous-time quantum random walk. Many applications such as search algorithm [SKW03], element distinctness [Amb07] are developed upon quantum walk.

CHAPTER 4
GROVER-TYPE QUANTUM SCHEDULING ALGORITHM

4.1 Grover-type Problems

Recently, Furrow overviewed applications based upon Grover-type algorithms [Fur06]. He considered three classes of applications: a) Graph algorithms, e.g. Breadth-First Search (BFS), Depth-First Search (DFS), bipartite matching; b) Computational Geometry algorithms, e.g. Maximum points on a line; c) Dynamic programming algorithms, such as coin changer. The author reports the quantum versus classical complexity for BFS and DFS, $O(\sqrt{VE \lg V})$, versus $O(E)$, with V the number of vertices and E the number of edges of the graph; for bipartite matching, $O(V\sqrt{(E+V)\lg V})$, versus $O((E+V)\sqrt{V})$); for maximum points that lie on a line in R^2, out of N points, $O(N^{3/2}\lg N)$ versus $O(N^2 \lg N)$, and so on.

Most of the problems discussed in [Fur06] are intrinsically search problems and the idea of applying Grover's search comes naturally to mind. There is an even larger class of problems which, at the first sight, do not seem directly related to Grover's search. Applications such as scheduling and resource allocation are not naturally search problems; nevertheless they share some common properties, can be reformulated to take advantage of quantum parallelism and

entanglement, and lead to algorithms which show polynomial speedups over their classical counterparts.

A scheduling problem is characterized by a tuple $(\alpha \mid \beta \mid \gamma)$ where α denotes the machine environment, β summarizes the set of constraints, and γ denotes the optimality criterion. The makespan of a schedule, C_{max} is the maximum completion time of any job in the schedule. For example, $P||C_{max}$ and $R||C_{max}$ require the shortest makespan and apply to identical machine environment and, respectively, a non-homogeneous one.

When we turn our attention to problems when a deadline is imposed, or when we wish to find a schedule with a given range of possible average completion time we discover that a full range of scheduling problems have a quantum counterpart which can take advantage of Grover's search.

We illustrate these ideas with an example: given a deadline, or a range of deadlines, our algorithm allows us to determine if a solution to an $R||C_{max}$ problem with N jobs and M machines exists, and if so, it provides the schedule. The time complexity of the quantum scheduling algorithm is $O(\sqrt{M^N})$ while the complexity of its classical counterpart is $O(M^N)$.

Real-time systems are subject to deadlines and Quality of Service (QoS) constraints imposed to many systems require a given range of average completion times. Thus, the classes of scheduling algorithms we discuss in our research are of significant practical importance. Such algorithms have a quantum counterpart that enjoys a square root speedup.

4.2 Introduction to Scheduling Problems

Scheduling is the problem of assigning tasks to a set of resources subject to a set of constraints, over time, in an "optimal" manner.

We are given a set of $N = 2^n$ jobs, $J = \{J_1, J_2, \ldots J_N\}$ and a set of $M = 2^m$ machines, $M = \{M_1, M_2, \ldots M_M\}$. A *schedule* S for the sets (J, M) specifies which T_{ij} units of time machine M_j uses to process job J_i. We call C_i^S the *completion time* of job J_i under the schedule S. The *makespan* of the schedule S is the maximum completion time of any job in schedule S:

$$C_{max}^S = \max_i C_i^S.$$

We are often interested in scheduling problems involving *multiple* machines. We distinguish three cases in the *parallel machine environment*:

1. *Identical parallel environment.* All the machines are identical and job J_i requires T_i units of time on any machine,

2. *Uniformly related parallel environment.* Each machine M_j has a speed $s_j > 0$ and job J_i if processed entirely on machine M_j would take T_i/s_j units of time, and

3. *Unrelated parallel machine environment.* Different machines have different capabilities and the speed of machine M_j on job J_i is s_{ij}. Then the processing time of job J_i on machine M_j is $T_{ij} = T_i/s_{ij}$, $0 \leq T_{ij} \leq Q$ and $Q = 2^q$.

These machine environments are denoted by P, Q, and R, respectively.

Examples of scheduling constraints include deadlines (e.g., job i must be completed by time t), resource capacities (e.g., there are only 5 machines), precedence constraints on the order of tasks (e.g., one must be done before another), and priorities on tasks (e.g., finish job j as soon as possible while meeting the other deadlines). A *priority rule* assigns to job J_i a priority π_i. A *busy schedule* defines the situation where when one machine becomes available it starts processing the job with the highest priority.

Each scheduling problem could have problem-specific optimization criteria. The one which requires the minimization of the makespan is referred to in scheduling theory as C_{max}. Other optimization criteria can be considered. For example, we may wish to optimize the average completion time of all jobs:

$$\frac{1}{N}\sum_{i=1}^{N} C_i^S$$

an optimization criterion denoted as $\sum_{i=1}^{N} C_i^S$.

Many scheduling problems are $NP-$ hard on classical computers and have proved to be very difficult even for a relatively small instance. For example, a 10-job-10-machine job-shop scheduling problem posed in 1963 remained unsolved until 1989 [CP89]. Most classical scheduling algorithms gain speedup only on some special cases with relatively small instance or they try to find good schedules instead of the optimal one. Furrow [Fur06] also gave a special case of the $P||C_{max}$ problem that can be solved by dynamic programming, which

requires that the number of different jobs processing times be bounded by a constant. It turns out that there are polynomial approximations for some $P||C_{max}$ problems.

We consider an $R||C_{max}$ scheduling problem in which all machines are unrelated. All jobs are available at the beginning time and there are no precedence constraints. As no preemption is allowed, once job J_i started processing on machine M_j it must complete its execution before another job, J_k can be processed on M_j.

Given a set of $N = 2^n$ jobs and $M = 2^m$ machines we can construct 2^{m2^n} different schedules. This $R||C_{max}$ scheduling problem is $NP-$ hard and is difficult for classical algorithms even with small instance. Up to now, most classical algorithms use linear programming based rounding techniques(LP-rounding) to search an approximate schedule instead of the optimal one [GSU04,SS02]. If the number of machines m is part of the input, the best approximation algorithm to date is a 2-approximation by Lenstra, Shmoys and Tardos [LST90] which can find a schedule with $C_{max} < 2 * C^{opt}_{max}$. Moreover, the problem cannot be approximated within a factor strictly smaller than 3/2, unless $P = NP$ [LST90]. No proper classical algorithm addresses a general case of searching the optimal schedule except the exhausting search which has time complexity $O(M^N)$ on a classical computer. We suggest a reformulation of such problems to take advantage of Grover-type search and gain square root speedup on searching the optimal schedules over their classical counterparts.

4.3 Quantum Scheduling Algorithm

4.3.1 Information Encoding

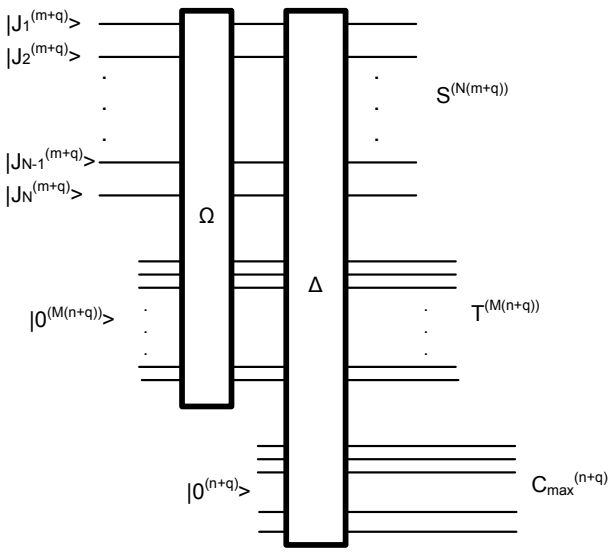

Figure 4.1: A quantum circuit to prepare the makespan vector

Let Ω be an operator which given a schedule constructs the running time on each machine for that schedule. When applied to the equal superposition of all schedules, it produces a superposition of the running time $\mid T\rangle$ on each machine for all schedules:

$$\mid \widehat{ST} \rangle = \Omega(\mid S\rangle \mid 0\rangle).$$

where, $\mid \widehat{ST} \rangle$ denotes an entangled state of S and T, while the tensor product of S and T is $\mid ST\rangle$. Let Δ be an operator which computes a superposition of the makespan of all

schedules:

$$|\overbrace{STC_{max}}\rangle = \Delta(|\overbrace{ST}\rangle \mid 0\rangle) = \Delta\ \Omega(|\ S\rangle \mid 0\rangle \mid 0\rangle).$$

Figure 4.1 outlines our procedure to produce an equal superposition of the makespans of all schedules. First we prepare job vectors in a equal superposition of all possible schedules $|\ S^{(N(m+q))}\rangle$. Using a quantum circuit Ω, we construct the superposition of the running time on each machine for every schedule. Then we construct the makespan of each schedule using the operation Δ. The notation $|\ J_i^{(m+q)}\rangle$ indicates that the vector $|\ J_i\rangle$ consists of $m+q$ qubits.

As we have a set of $N = 2^n$ jobs running on $M = 2^m$ machines, we assume that the jobs could have different processing times on different machines and that the processing times are integers in the range $0 \leq T_{ij} < Q = 2^q, \quad 1 \leq i \leq 2^n, \quad 1 \leq j \leq 2^m$. We also assume that we have a quantum system with $r = m + q$ qubits for each job.

Given job J_i running on machine M_j, we encode the *job-machine* information as a vector $|\ e_i^j\rangle$ obtained as the tensor product of the machine index, j, and the processing time, T_{ij}:

$$|\ e_i^j\rangle = |\ j\rangle \otimes |\ T_{ij}\rangle.$$

Then we define *job state vectors* for job i, $|\ J_i\rangle$ as any superposition of its job-machine vectors. We can have an equal superposition of job-machine vectors as:

$$|\ J_i\rangle = \frac{1}{2^{m/2}} \sum_{j=1}^{2^m} |\ e_i^j\rangle.$$

A *schedule* is a tensor product of job-machine vectors:

$$\mid S_k\rangle = \otimes_{i=1}^{2^n} \mid e_i^{j_i}\rangle, \quad 1 \leq j_i \leq 2^m,$$

which includes one job-machine vector for each job. A specific machine may be present in multiple job-machine vectors, when the schedule requires multiple jobs to be executed on the same machine, or may not appear in any job machine vectors of the schedule $\mid S_k\rangle$ if none of the jobs is scheduled on that machine.

The equal superposition of all schedules is:

$$\mid S\rangle = \frac{1}{\sqrt{\sigma}} \sum_{k=1}^{\sigma} \mid S_k\rangle, \quad \sigma = 2^{m2^n}$$

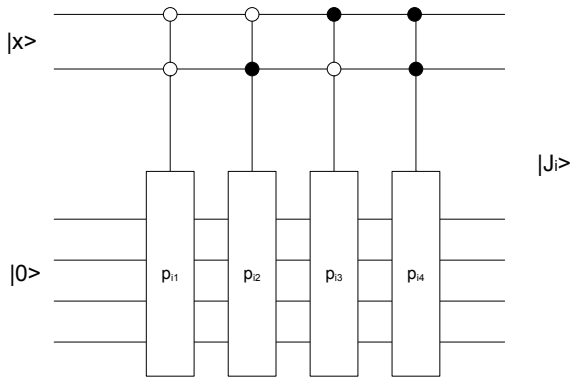

Figure 4.2: A quantum circuit to prepare the job state vectors

First, we need to prepare the job vector $\mid J_i\rangle$ in an equal superposition state which includes the processing times of job J_i on all machines, as shown in Figure 4.2. We use

m index qubits to control the job-machine information encoding. As each index qubit is prepared in state $1/\sqrt{2}(|\ 0\rangle + |\ 1\rangle)$, the target qubits will be prepared in superpositions of all possible job-machine states, $e_i^j, \quad 1 \leq i \leq n,\ 1 \leq j \leq m$.

Example: Table 4.1 summarizes the processing time of 8 jobs on 4 machines, where $0 < T_{ij} < 2^4 = 16$. Thus $n = 3$, $m = 2$, and $q = 4$. The running time of job J_1 on machines M_1, M_2, M_3 and M_4 are, respectively, $1, 3, 7$, and 15 units of time.

Table 4.1: An example of 8 jobs running on 4 machines

Job/Machine	M_1	M_2	M_3	M_4
J_1	1	3	7	15
J_2	2	1	9	3
J_3	6	2	5	8
J_4	11	13	7	4
J_5	15	12	3	10
J_6	10	7	8	14
J_7	5	2	3	9
J_8	1	10	11	13

The four vectors used to encode the processing time of job J_1 on machines M_1, M_2, M_3 and M_4 are respectively:

$$|\ e_1^1\rangle = |\ 000001\rangle, \qquad |\ e_1^2\rangle = |\ 010011\rangle, \qquad |\ e_1^3\rangle = |\ 100111\rangle, \quad \text{and} \quad |\ e_1^4\rangle = |\ 111111\rangle.$$

For J_2 the basis vectors are $|\ 000010\rangle$, $|\ 010001\rangle$, $|\ 101001\rangle$, $|\ 110011\rangle$, and so on.

As we set each control qubits in an equal superposition as $1/\sqrt{2}(|\ 0\rangle + |\ 1\rangle)$, the job states $|\ J_i\rangle$ will be prepared in a equal superposition of all basis states, for example

$$|\ J_1\rangle = \frac{1}{2}(|\ 000001\rangle + |\ 010011\rangle + |\ 100111\rangle + |\ 111111\rangle).$$

A schedule vector $|\ S\rangle$ is the tensor product of all job vectors. As each job vector is prepared as an equal superposition, the schedule vector is in the equal superposition of all possible schedules.

We now provide two examples of schedule vectors:

(i) Schedule S_1:

$$[J_1 \mapsto M_1,\ \ J_2 \mapsto M_2,\ \ J_3 \mapsto M_1,\ \ J_4 \mapsto M_4,\ \ J_5 \mapsto M_3,\ \ J_6 \mapsto M_2,\ \ J_7 \mapsto M_3,\ \ J_8 \mapsto M_1]$$

Schedule S_1 corresponds to the following job state vectors, where the underline part denotes the machine index:

$|\ J_1\rangle = |\ \underline{00}0001\rangle$ $\quad$ $|\ J_2\rangle = |\ \underline{01}0001\rangle$ $\quad$ $|\ J_3\rangle = |\ \underline{00}0110\rangle$ $\quad$ $|\ J_4\rangle = |\ \underline{11}0100\rangle$

$|\ J_5\rangle = |\ \underline{10}0011\rangle$ $\quad$ $|\ J_6\rangle = |\ \underline{01}0111\rangle$ $\quad$ $|\ J_7\rangle = |\ \underline{10}0011\rangle$ $\quad$ $|\ J_8\rangle = |\ \underline{00}0001\rangle$

The schedule vector for schedule S_1 is:

$$|\ S_1\rangle = |\ \underline{00}0001\rangle \otimes |\ \underline{01}0001\rangle \otimes |\ \underline{00}0110\rangle \otimes |\ \underline{11}0100\rangle$$
$$\otimes |\ \underline{10}0011\rangle \otimes |\ \underline{01}0111\rangle \otimes |\ \underline{10}0011\rangle \otimes |\ \underline{00}0001\rangle$$

The completion times on all machines are: $C^{A_1}(M_1) = 1 + 6 + 1 = 8,\ \ C^{A_1}(M_2) = 1 + 7 = 8,\ \ C^{A_1}(M_3) = 3 + 3 = 6,\ \ C^{A_1}(M_4) = 4.$

The makespan of schedule S_1 is equal to the largest completion time over all machines: $C_{max}^{A_1} = 8$.

(ii) Schedule S_2:

$$[J_1 \mapsto M_2, \quad J_2 \mapsto M_1, \quad J_3 \mapsto M_3, \quad J_4 \mapsto M_4, \quad J_5 \mapsto M_3, \quad J_6 \mapsto M_2, \quad J_7 \mapsto M_1, \quad J_8 \mapsto M_2]$$

The schedule vector is:

$$\begin{aligned} \mid S_2\rangle = & \mid \underline{01}0011\rangle \otimes \mid \underline{00}0010\rangle \otimes \mid \underline{10}0101\rangle \otimes \mid \underline{11}0100\rangle \\ & \otimes \mid \underline{10}0011\rangle \otimes \mid \underline{01}0111\rangle \otimes \mid \underline{00}0101\rangle \otimes \mid \underline{01}1010\rangle) \end{aligned}$$

The completion times for schedule S_2 for all machines are:

$C^{A_2}(M_1) = 2 + 5 = 7, \quad C^{A_2}(M_2) = 3 + 7 + 10 = 20, \quad C^{A_2}(M_3) = 5 + 3 = 8, \quad C^{A_2}(M_4) = 4.$

with makespan $C_{max}^{A_2} = 20$.

The schedule vector can also be in a superposition of some basic states, for example, the schedule vector could be in an equal superposition of the two schedules, S_1 and S_2:

$$\begin{aligned} \mid S\rangle &= 1/\sqrt{2}(\mid S_1\rangle + \mid S_2\rangle) \\ &= 1/\sqrt{2}(\mid \underline{00}0001\rangle \otimes \mid \underline{01}0001\rangle \otimes \mid \underline{00}0110\rangle \otimes \mid \underline{11}0100\rangle \\ &\qquad \otimes \mid \underline{10}0011\rangle \otimes \mid \underline{01}0111\rangle \otimes \mid \underline{10}0011\rangle \otimes \mid \underline{00}0001\rangle \\ &\quad + \mid \underline{01}0011\rangle \otimes \mid \underline{00}0010\rangle \otimes \mid \underline{10}0101\rangle \otimes \mid \underline{11}0100\rangle \\ &\qquad \otimes \mid \underline{10}0011\rangle \otimes \mid \underline{01}0111\rangle \otimes \mid \underline{00}0101\rangle \otimes \mid \underline{01}1010\rangle). \end{aligned}$$

4.3.2 Calculating Makespans

To calculate the makespan of each schedule, the first step is to summarize the running times on each machine. Now we are concerned with the construction of a quantum adder similar to a classic adder, e.g. $\mid 5\rangle\hat{+}\mid 6\rangle =\mid 11\rangle$. A quantum adder has:

- n inputs $a_1, a_2, \cdots, a_n$, each one is a register of q qubits,
- one carry in c, and
- one output $S = \sum_{i=1}^{n} a_i + c$, a register of $q + n$ qubits.

To obtain the total running time of machine M_j under schedule S_k we add the execution times, (in our examples those qubits of a job-machine vector which are not underlined) for all jobs assigned to M_j and create a running time vector for machine M_j, $\mid T_j\rangle$. The *running time vector for schedule* S_k is the tensor product of the execution time on individual machines under schedule S_k:

$$\mid T^{S_k}\rangle =\mid T_1^{S_k}\rangle\otimes\mid T_2^{S_k}\rangle\cdots\otimes\mid T_M^{S_k}\rangle.$$

For example, for machine M_1 under schedule S_1 (see Section 4.3.1):

$$\mid T_1^{S_1}\rangle : \quad \mid 0001\rangle\hat{+}\mid 0000\rangle\hat{+}\mid 0110\rangle\hat{+}\mid 0000\rangle\hat{+}\mid 0000\rangle\hat{+}\mid 0000\rangle\hat{+}\mid 0001\rangle,$$

or

$$\mid T_1^{S_1}\rangle : \quad \mid 0001000\rangle.$$

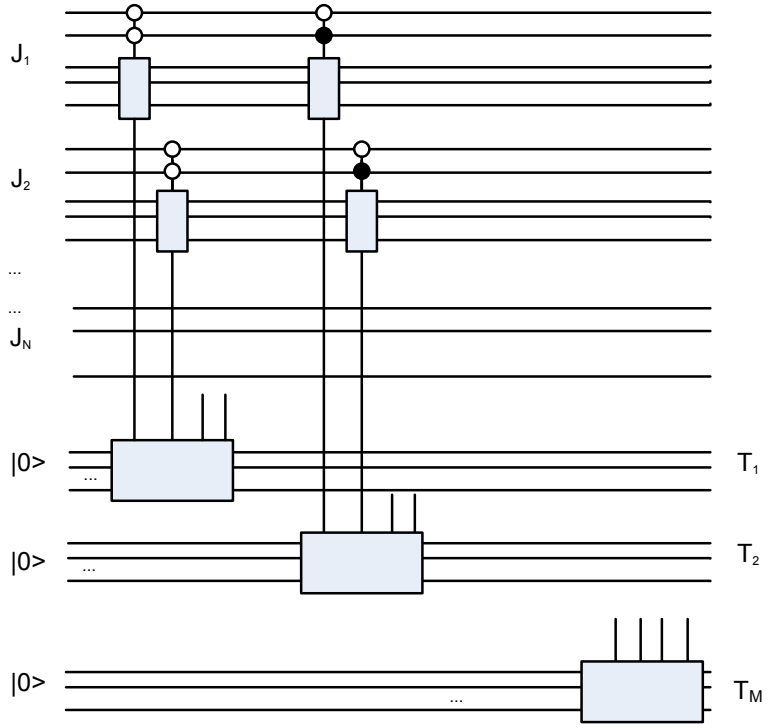

Figure 4.3: A circuit to compute the sum of the execution time of jobs assigned to each machine.

We wish to construct a quantum circuit to sum the execution time of jobs assigned to each machine M_j. We use the index qubits as control qubits to control the summation of each index, or each machine; the index qubits are entangled with the target qubits which give the running time on that machine. As the input qubits $\mid S\rangle$ are prepared in the equal superposition of all possible schedules, this operation will prepare the running time of machines under all possible schedules.

In Figure 4.3, $T_1, T_2, \ldots T_M$ represent the execution time on machines $M_1, M_2, ...M_M$ respectively. When the index qubits are not "active", the respective input for the summation circuit will be zero. For example, if the index of $e_2^2 =\mid \underline{01}0001\rangle$ is $\underline{01}$, rather than $\underline{00}$, then the index qubits of e_2^2 for $\mid T_1\rangle$ are not active.

How to implement arithmetic on a quantum computer has been addressed by many authors. Many detailed quantum circuits and discussions can be found in [Gos98, TS06, MI05]. For the example discussed in Section 4.3.1, after the summation operation, the system will be in the state:

$$\mid \widetilde{ST}\rangle = 1/\sqrt{2}(\mid S_1T^1\rangle + \mid S_2T^2\rangle)$$

or

$$\begin{aligned}\mid \widetilde{ST}\rangle = 1/\sqrt{2}(&\mid \underline{00}0001\rangle\otimes \mid \underline{01}0001\rangle\otimes \mid \underline{00}0110\rangle\otimes \mid \underline{11}0100\rangle\otimes \mid \underline{10}0011\rangle\otimes \mid \underline{01}0111\rangle\\ &\otimes \mid \underline{10}0011\rangle\otimes \mid \underline{00}0001\rangle\otimes \mid 0001000\rangle\otimes \mid 0001000\rangle\otimes \mid 0000110\rangle\otimes \mid 0000100\rangle\\ +&\mid \underline{01}0011\rangle\otimes \mid \underline{00}0010\rangle\otimes \mid \underline{10}0101\rangle\otimes \mid \underline{11}0100\rangle\otimes \mid \underline{10}0011\rangle\otimes \mid \underline{01}0111\rangle\\ &\otimes \mid \underline{00}0101\rangle\otimes \mid \underline{01}1010\rangle\otimes \mid 0000111\rangle\otimes \mid 0010100\rangle\otimes \mid 0001000\rangle\otimes \mid 0000100\rangle)\end{aligned}$$

Now we have the system in state $\mid \widetilde{ST}\rangle$, of which the last $M(n+q)$ qubits provide the running time of all M machines under all schedules. The makespan of a schedule is equal to the maximum running time among the M machines under that schedule. We want to construct a `Max` circuit, as shown in Figure 4.4, to compute the makespan of each schedule. The quantum circuit computes the maximum over an input set, for example,

$\text{Max}(|\ 5\rangle, |\ 7\rangle, |\ 4\rangle) = |\ 7\rangle$.

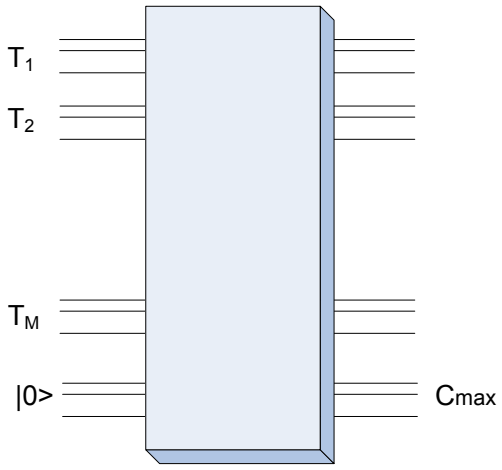

Figure 4.4: The quantum circuit to compute the makespan, the maximum running time among all machines. The output of this circuit is an entangled state.

We give an example of such a `Max` circuit for two 2-qubit inputs x and y as shown in Figure 4.5. With two ancilla qubits, this quantum circuit gives the C_{max} of the two inputs on the last two qubits. Table 4.2 shows the truth table of the `Max` circuit.

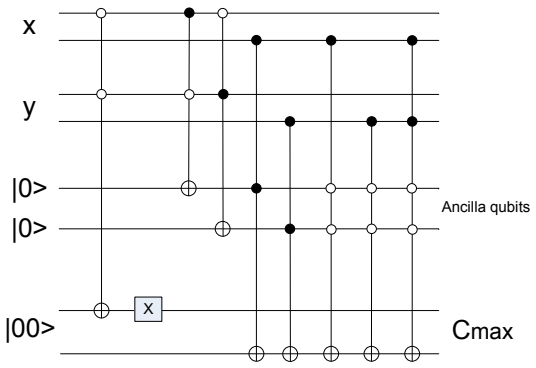

Figure 4.5: A two-qubit quantum circuit for the `Max` function.

Table 4.2: The truth table for `Max` quantum circuit with two 2-qubit input registers $|x\rangle, |y\rangle$.

$\|x\rangle$	$\|y\rangle$	$\|Max\rangle$	$\|x\rangle$	$\|y\rangle$	$\|Max\rangle$
00	00	00	00	10	10
01	00	01	01	10	10
10	00	10	10	10	10
11	00	11	11	10	11
00	01	01	00	11	11
01	01	01	01	11	11
10	01	10	10	11	11
11	01	11	11	11	11

The output of a quantum circuit as in Figure 4.4 is an entangled state, rather than the tensor product $|T_1\rangle \otimes |T_2\rangle \otimes \cdots \otimes |T_M\rangle \otimes |C_{max}\rangle$. This means that once we make a measurement on the system, we project all the vectors $T_1, T_2, \ldots T_M$ onto one special schedule with its makespan C_{max}. Recall that the $|T\rangle$ was prepared in an equal superposition of running times of the machines under all possible schedules. Thus, such a `Max` operation prepares the C_{max} register in an equal superposition of the makespans of all possible schedules.

4.3.3 Searching the Schedule

Up to now, we discussed the implementation of the quantum circuits presented in Figure 4.1:

- We prepare each job vector $|J_i\rangle$ in a superposition state which includes the running times on all machines. The first m qubits of a job vector are used for the index of

the machine and remaining q qubits are used for the running time of that job on the machine.

- We summarize the execution time of all jobs according to their machine indexes, which produces all schedules and gives the running time of each machine T_j under these schedules. The system is prepared in an entangled state:

$$|\overbrace{ST}\rangle = \frac{1}{\sqrt{\sigma}} \sum_{\text{each schedules k}} (\bigotimes_{i=1}^{N} |\ J_{ik}\rangle \bigotimes_{j=1}^{M} |\ T_{jk}\rangle) \qquad \sigma = 2^{m2^n},$$

  a superposition of job vectors and running time vectors of all possible schedules.

- We obtain the maximum running time among all machines using the `Max` quantum circuit and prepare the system in state:

$$|\overbrace{STC_{max}}\rangle = \frac{1}{\sqrt{\sigma}} \sum_{\text{each schedules k}} (\bigotimes_{i=1}^{N} |\ J_{ik}\rangle \bigotimes_{j=1}^{M} |\ T_{jk}\rangle\ |\ C_{max\ k}\rangle) \qquad \sigma = 2^{m2^n}$$

As we can see, our algorithm successfully prepares the system in an equal superposition of all 2^{m2^n} possible schedules. We define this whole preparation process as Z. This Z transformation does not carry out a measurement of the system at any time. Therefore, there exists an inverse transformation operation Z^{-1}. We can use the amplitude amplification algorithm to search the schedule with the makespan $D_k = \mu$. If we find such a makespan, the job vectors will be projected as well. These projections will give us the actual mapping of jobs to machines corresponding to the schedule with the given makespan.

The searching process consists of the following steps:

- Apply the Z transformation on $\mid 0\rangle$ to prepare the system in an equal superposition of all 2^{m2^n} possible schedules, $\mid \psi\rangle$.

- Repeat the following steps $O(\sqrt{\sigma})$ times:

  Apply Q on $\mid \psi\rangle$, in which $Q = -ZS_0Z^{-1}S_\chi$

- Measure the resulting state.

- Return the result, in which the job vectors give the detailed arrangement of the schedule.

The oracle in our search algorithm exhibits a difference in one significant aspect when compares to the oracle in Grover's algorithm which checks all qubits to reverse the solution(s) of the searching problem. In our case, the oracle only checks a subset, C_{max} qubits. It is easy to implement such an oracle using an idea similar to the one in [NC00].

4.4 Scheduling Problems with a Quantum Counterpart

Many other scheduling problems can be reformulated to take advantage of quantum search and exhibit a square root speedup versus their classical counterparts. Such problems require that a synthetic measure of performance, μ be within a given range, $\mu_{min} \leq \mu \leq \mu_{max}$.

The generic quantum algorithm proceeds as follows:

1. Devise an encoding scheme for the information required to compute μ_{S_i} for a given schedule S_i.

2. Design an algorithm to compute the synthetic measure of performance, μ_{S_i}.

3. Construct $\mid Q\rangle$ with $Q = (\mu_{S_i}, S_i)$ in a superposition state for all possible schedules.

4. Design the quantum circuits for the specific encoding scheme and for the algorithms. Given a set of values $\{q_1, q_2, \ldots q_n\}$ the circuit should be able to compute a function $\mu_{S_i} = f(q_1, q_2, \ldots q_n)$. For example, we may wish to compute the maximum, the minimum, or an average value.

5. Design an oracle to identify a specific value of the function μ_{S_i}.

6. Use the quantum search to find if there is a value $\mu_{S_i} = \mu_{min}$. If so, determine the corresponding schedule S_i. Continue this process until $\mu_{S_i} = \mu_{max}$.

7. If no schedule can be found then report failure, otherwise provide the list of all schedules S_i and the corresponding measures of performance, μ_{S_i}.

Consider for example the $R||\sum_{i=1}^{N} C_i^S$ scheduling problem when the goal is to optimize the average completion time in unrelated parallel machine environment. It has the similar encoding process as the $R||C_{max}$ problem. During the process we construct all schedules, we also summarize the completion time of different jobs using some simple arithmetic circuits, followed by Grover-type search. Other scheduling problems such as minimizing the average waiting time, could take advantage of quantum search.

Oftentimes, we have to search for schedules that optimize the largest subset of a set of synthetic measures of performance, $\mu, \nu, \pi, \rho, \theta \ldots$. For example, we could have multiple synthetic performance indicators related to: timing, resource utilization, cost, and quality of service. In this case we would run repeatedly the scheduling algorithm for each performance measure and search for a solution in the given range for each measure. Once we have conducted all individual searches we determine the intersection of all schedules that satisfy all conditions; if the set is empty we remove individual conditions one by one until we find the a non-empty set.

Scheduling is also intimately related to planning when we have a complex goal and the sequence of actions to reach each goal has to be determined. The scenario described above is repeated for each plan, thus the square root speedup of algorithms based upon quantum search becomes even more important.

4.5 Summary of Grover-type Problems

When a deadline is imposed, or when we wish to find a schedule with a given range of possible average completion time we discover that a full range of scheduling problems have a quantum counterpart which can take advantage of Grover's search.

Many scheduling problems, resource allocations, and path-finding problems, share some common properties with the $R||C_{max}$ problem discussed in our research: a well-defined initial

state, a well-defined desired state or a range of desired states, many paths to reach the desired state, and well-defined attributes of an optimal path.

The quantum algorithmic solution to such problems requires the following steps:

- Prepare the initial state in an equal superposition of all possible choices.
- Use some reversible quantum arithmetic to compute the specialized property (makespan in our case) needed.
- Construct the necessary oracle circuit.
- Use Grover-type algorithms to search for the desired solution.

The solution we propose based upon Grover's algorithm is not universally applicable. Problem requiring specific optimization criteria may require quantum circuits that cannot be easily implemented with reversible quantum arithmetic. Moreover, Grover-type algorithms lead to a square-root improvement over the exhausting search, while many good classical algorithms may have better performance for some special problems.

CHAPTER 5
QUANTUM ERROR CORRECTION CODES

5.1 Quantum Errors

All physical systems, quantum or classical, are subject to errors. The main problems include environmental noise, which is due to incomplete isolation of the system from the outside, and control errors, which are caused by calibration errors and random fluctuations in operations. However, a classical system may be stabilized to a very high degree, either by making the ratio of system size to perturbation size very large (passive stabilization), or by continuously monitoring the system and providing greatly enhanced 'inertia' against random fluctuation be means of feedback control (active stabilization). When applied in the quantum regime, the passive stabilization needs to build the system beyond a certain degree which is not easy by any currently attemptable method. The active stabilization seems also impossible from a quantum mechanics point of view since the feedback control involves dissipation, and therefore is non-unitary.

Is active stabilization of a quantum bit possible? The answer is YES. The stabilization, the quantum error correction, is based on the classical theory of error correction, which provides a very powerful technique by which classical information can be transmitted without errors through the medium of a noisy channel.

Quantum error correction is used in quantum computing to protect quantum information from errors due to decoherence and other quantum noise. It is essential for fault-tolerant quantum computation. It is designed to deal not just with noise during quantum information communication, but also with the protection of stored information. In this case, the user encodes the information in a storage system and retrieves it at a later time. Since the quantum information is not absolutely isolated from the environment(actually the influence is much larger than in the classical world), any error-correction method applicable to communication is also applicable to storage and vice versa. Furthermore, quantum error correction is also essential for faulty quantum gates, faulty quantum preparation, and faulty measurements. The discovery of powerful error correction methods therefore caused much excitement, since it converted large scale quantum computation from a practical impossibility to a possibility.

An error is a unitary transformation of the state space. The space of errors for a single qubit is spanned by the four Pauli matrices as shown in Table 5.1.

Table 5.1: Quantum errors for a single qubit are spanned by the Pauli matrices

I	no error	$\mid 0\rangle \rightarrow \mid 0\rangle \quad \mid 1\rangle \rightarrow \mid 1\rangle$
X	bit error	$\mid 0\rangle \rightarrow \mid 1\rangle \quad \mid 1\rangle \rightarrow \mid 0\rangle$
Z	phase error	$\mid 0\rangle \rightarrow \mid 0\rangle \quad \mid 1\rangle \rightarrow - \mid 1\rangle$
$Y = iXZ$	combination	$\mid 0\rangle \rightarrow i \mid 1\rangle \quad \mid 1\rangle \rightarrow -i \mid 0\rangle$

Generally, the evolution of a qubit in state $|0\rangle$ interacting with an environment $|E\rangle$ will yield a state as:

$$|0\rangle|E\rangle \rightarrowtail \beta_1|0E_1\rangle + \beta_2|1E_2\rangle$$

Similarly, a qubit in state $|1\rangle$ will change into a state:

$$|1\rangle|E\rangle \rightarrowtail \beta_3|1E_3\rangle + \beta_4|0E_4\rangle$$

Therefore, an arbitrary state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ interacts with the environment $|E\rangle$ as:

$$\begin{aligned}
(\alpha_0|0\rangle + \alpha_1|1\rangle)|E\rangle &\rightarrowtail \alpha_0\beta_1|0E_1\rangle + \alpha_0\beta_2|1E_2\rangle + \alpha_1\beta_3|1E_3\rangle + \alpha_1\beta_4|0E_4\rangle \\
&= \frac{1}{2}(\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_1|E_1\rangle + \beta_3|E_3\rangle) \\
&\quad + \frac{1}{2}(\alpha_0|0\rangle - \alpha_1|1\rangle)(\beta_1|E_1\rangle - \beta_3|E_3\rangle) \\
&\quad + \frac{1}{2}(\alpha_0|1\rangle + \alpha_1|0\rangle)(\beta_2|E_2\rangle + \beta_4|E_4\rangle) \\
&\quad + \frac{1}{2}(\alpha_0|1\rangle - \alpha_1|0\rangle)(\beta_2|E_2\rangle - \beta_4|E_4\rangle) \\
&= \frac{1}{2}|\psi\rangle(\beta_1|E_1\rangle + \beta_3|E_3\rangle) + \frac{1}{2}Z|\psi\rangle(\beta_1|E_1\rangle - \beta_3|E_3\rangle) \\
&\quad + \frac{1}{2}X|\psi\rangle(\beta_2|E_2\rangle + \beta_4|E_4\rangle) + \frac{1}{2}XZ|\psi\rangle(\beta_2|E_2\rangle - \beta_4|E_4\rangle)
\end{aligned}$$

This represents the most general evolution that may occur on a single qubit, and the last 3 terms represent phase-flip, bit-flip and bit-phase-flip respectively.

Similar to classical error correction, quantum error correction also operates by the judicious use of redundancy, that is, making the system larger than the information itself, which could make it more resistent to perturbations. As we discussed in section 2.6, many different properties from classical error correction must be carefully considered when we construct quantum error-correcting codes. The first quantum error correcting codes were

discovered independently by Shor [Sho95] and Steane [Ste96]. Some important quantum error-correcting codes are discussed next.

5.2 The Three-Qubit Phase-Flip Code

As we introduced in section 2.6, a quantum repetition code can correct one bit-flip error. Unlike classical system, a quantum system could also suffer phase-flip errors which leave $\mid 0\rangle$ unchanged, and flip the sign of $\mid 1\rangle$. In another word, the phase-flip operator Z is applied to the qubit.

Recall that the Hadamard gate

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

takes bit errors to phase errors and vice versa. If we apply the Hadamard gates on quantum repetition code before sending it into the noisy channel, the new code will be able to correct one phase-flip error instead of one bit-flip error. Thus we encode the state $\mid 0_L\rangle \rightarrow \mid +++\rangle$ and $\mid 1_L\rangle \rightarrow \mid ---\rangle$ as the logical states zero and one, where $\mid +\rangle \equiv (\mid 0\rangle + \mid 1\rangle)/\sqrt{2}$, $\mid -\rangle \equiv (\mid 0\rangle - \mid 1\rangle)/\sqrt{2}$. Similarly, the inverse of the Hadamard gate(which is also the Hadamard gate) is applied before the normal decoding procedure. The encoding and decoding circuits are shown in Figure 5.1. The two bottom lines represent two ancilla qubits which are used to measure the error syndrome without changing the code itself.

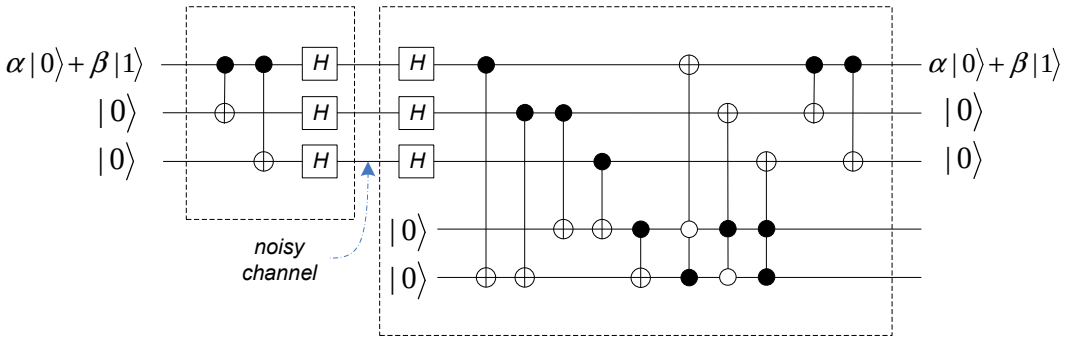

Figure 5.1: Encoding and decoding circuits for the phase-flip code.

5.3 The Shor Code

The Shor code is a simple code which can protect against the effects of an arbitrary error on a single qubit. The code is a combination of the three qubit phase-flip and bit-flip codes. It encode the logic qubits as follows:

$$|\ 0\rangle \rightarrow |\ 0_L\rangle = (|\ 000\rangle + |\ 111\rangle)(|\ 000\rangle + |\ 111\rangle)(|\ 000\rangle + |\ 111\rangle)$$

$$|\ 1\rangle \rightarrow |\ 1_L\rangle = (|\ 000\rangle - |\ 111\rangle)(|\ 000\rangle - |\ 111\rangle)(|\ 000\rangle - |\ 111\rangle)$$

The Shor code is able to protect against phase-flip and bit-flip errors on any qubit. Suppose a bit-flip error occurs on some qubit , say the first one, switching $|\ 0\rangle$ and $|\ 1\rangle$. Then by comparing the first two qubits, we find they are different, which is not allowed by any valid codeword. Therefore an error should have occurred which flipped either the first or second qubit. Note that we do not actually measure the first and second qubits, since this would destroy the superposition in the codeword; we can make a measurement $Z_1 Z_2$ which only measures the difference between them. Now we compare the first and third qubits.

Since the first qubit was flipped, it will disagree with the third; if the second qubit had been flipped, the first and third would be the same. Therefore, we have narrowed down the error to the first qubit and we can fix it simply by flipping it back. To handle possible bit-flips on the other blocks of three, we can make the same comparisons inside the blocks.

However a phase-flip will leave the identity of the 0 and 1 alone, but alter their relative phase. For example, such a phase-flip flips the sign of the first block of qubits, changing $|\,000\rangle + |\,111\rangle$ into $|\,000\rangle - |\,111\rangle$, and vice versa. By comparing the sign of the first block of three with the second block of three, we can see that a sign error has occurred in one of these blocks. Then by comparing the signs of the first and third block of three, we narrow the sign error down to the first block, and flip the sign back to what it should be. Again, we do not actually measure the sign, which will destroy the state. Instead, we only compare signs between the blocks.

If we have both a bit-flip and a phase-flip on the same qubit, it is easy to see that going through both processes above, we can detect and correct them. Thus, the Shor code could correct any one-qubit error.

5.4 CSS Codes

Calderbank-Shor-Steane codes, or CSS codes for short, are quantum codes which let us identify and correct large qubit errors, i.e., errors described by Pauli matrices. CSS codes

derive from classical linear codes and represent a systematic way of building a quantum error correcting code.

In order to construct a CSS code you need to have two classical linear codes, $C_1[n, k_1]$ and $C_2[n, k_2]$ such that $C_2 \subset C_1$ and $C_1, C_2^{\perp}$ both correct t errors. The resulting code is a quantum code $CSS(C_1/C_2)$ capable of correcting errors on t qubits, which encodes $k_1 - k_2$ logical qubits in n physical qubits, so this code is $[n, k_1 - k_2]$.

The encoding is in a vector space spanned by all states constructed by taking a codeword $x \in C_1$ and then adding to it the whole of C_2:

$$\mid x + C_2\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} \mid x + y\rangle,$$

where $+$ is bitwise addition modulo 2 and $|C_2|$ is the number of elements in C_2. Suppose $x' \in C_1$ and $x - x' \in C_2$. then it is easy to see that $\mid x + C_2\rangle = \mid x' + C_2\rangle$, and thus the state $\mid x + C_2\rangle$ depends only upon the coset of C_1/C_2. Furthermore, if x and x' belong to different cosets of C_2, then for no $y, y' \in C_2$ does $x + y = x' + y'$, and therefore $\mid x + C_2\rangle$ and $\mid x' + C_2\rangle$ are orthonormal states.

An important example of CSS code is the Steane code which is constructed using the classic [7,4,3] Hamming code with parity check matrix:

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Let us take the C_1 [7,4,3] Hamming code and its dual as C_2. Thus we have $C_2 \subset C_1$. By definition the parity check matrix of C_2 is equal to the transposed generator matrix of C_1:

$$H_2 = G_1^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

An element that obviously must belong to C_1 is $\mid 0000000\rangle$. This element belongs to C_2 too. Therefore the first element of C_1/C_2, which we are going to identify with $\mid 0_L\rangle$ is going to be $\sum_{y \in C_2}(\mid 0000000\rangle + y)$. In order to find the second element of C_1/C_2, which we are going to identify with $\mid 1_L\rangle$, we need to find a vector in C_1 that does not belong to C_2. Such a vector is, for example, $\mid 1111111\rangle$.

Thus, we will give the codewords of the Steane code as:

$$\mid 0_L\rangle = \tfrac{1}{\sqrt{8}}[\mid 0000000\rangle + \mid 1010101\rangle + \mid 0110011\rangle + \mid 1100110\rangle]$$
$$+ \mid 0001111\rangle + \mid 1011010\rangle + \mid 0111100\rangle + \mid 1101001\rangle$$

$$\mid 1_L\rangle = \tfrac{1}{\sqrt{8}}[\mid 1111111\rangle + \mid 0101010\rangle + \mid 1001100\rangle + \mid 0011001\rangle]$$
$$+ \mid 1110000\rangle + \mid 0100101\rangle + \mid 1000011\rangle + \mid 0010110\rangle$$

5.5 Stabilizer Codes

The stabilizer formalism is a succinct manner to describe a quantum error correcting code by a set of quantum operators [Got97]. We first review several concepts and properties of stabilizer codes.

The 1-qubit Pauli group G_1 consists of the Pauli matrices, σ_I, σ_x, σ_y, and σ_z together with the multiplicative factors ± 1 and $\pm i$:

$$G_1 \equiv \{\pm\sigma_I, \pm i\sigma_I, \pm\sigma_x, \pm i\sigma_x, \pm\sigma_y, \pm i\sigma_y, \pm\sigma_z, \pm i\sigma_z\}.$$

The generators of G_1 are:

$$\langle \sigma_x, \sigma_z, i\sigma_I \rangle.$$

Indeed, every element of G_1 can be expressed as a product of a finite number of generators. For example:

$$-\sigma_x = i\sigma_I i\sigma_I \sigma_x, \qquad +i\sigma_x = i\sigma_I \sigma_x, \qquad -i\sigma_x = i\sigma_I i\sigma_I i\sigma_I \sigma_x.$$

The n-qubit Pauli group G_n consists of the 4^n tensor products of σ_I, σ_x, σ_y, and σ_z and an overall phase of ± 1 or $\pm i$. Elements of the group can be used to describe the error operators applied to an n-qubit register. The *weight* of such an operator in G_n is equal to the number of tensor factors which are not equal to σ_I.

For example, consider the case $n = 5$. The 5-qubit Pauli group consists of the tensor products of the form:

$$\mathbf{E}^{(1)} \otimes \mathbf{E}^{(2)} \otimes \mathbf{E}^{(3)} \otimes \mathbf{E}^{(4)} \otimes \mathbf{E}^{(5)} \quad \text{with} \quad \mathbf{E}^{(i)} \in G_1, \quad 1 \leq i \leq 5.$$

The operator:

$$\mathbf{E}_\alpha = \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I$$

has a weight equal to 2 and represents a bit-flip of qubit 2 and a phase-flip of qubit 4 when the qubits are numbered from left to right.

The stabilizer S of code Q is a subgroup of the n-qubit Pauli group,

$$S \subset G_n.$$

The generators of the subgroup S are:

$$M = \{\mathbf{M}_1, \mathbf{M}_2 \ldots \mathbf{M}_q\}.$$

The eigenvectors of the generators $\{\mathbf{M}_1, \mathbf{M}_2 \ldots \mathbf{M}_q\}$ have special properties: those corresponding to eigenvalues of $+1$ are the codewords of Q and those corresponding to eigenvalues of -1 are codewords affected by errors. If a vector $\mid \psi_i\rangle \in H_n$ satisfies,

$$\mathbf{M}_j \mid \psi_i\rangle = (+1) \mid \psi_i\rangle \qquad \forall \mathbf{M_j} \in M$$

then $\mid \psi_i\rangle$ is a codeword, $\mid \psi_i\rangle \in Q$. This justifies the name given to the set S, any operator in S stabilizes a codeword, leaving the state of a codeword unchanged. On the other hand

if:

$$\mathbf{M}_j \mid \varphi_k\rangle = (-1) \mid \varphi_k\rangle.$$

then $\mid \varphi_k\rangle = \mathbf{E_i} \mid \psi_k\rangle$, the state $\mid \varphi_k\rangle$ is a codeword $\mid \psi_k\rangle \in Q$ affected by error $\mathbf{E_i}$. The error operators affecting codewords in Q, $E = \{\mathbf{E}_1, \mathbf{E}_2 \ldots\}$ are also a subgroup of the n-qubit Pauli group

$$E \subset G_n.$$

Each error operator $\mathbf{E}_i$ is a tensor product on n Pauli matrices. Its weight is equal to the number of errors affecting a quantum word, thus with the number of Pauli matrices other than σ_I.

The coding space

$$Q = \{\mid \psi\rangle \in H_n \text{ such that } M_j \mid \psi\rangle = (+1) \mid \psi\rangle, \qquad \forall M_j \in S \}$$

is the space of all vectors $\mid \psi\rangle$ fixed by S.

It is easy to prove that S is a stabilizer of a non-trivial Hilbert subspace $V_{2^n} \subset H_{2^n}$ if and only if:

1. $S = \{\mathbf{S}_1, \mathbf{S}_2, \ldots\}$ is an Abelian group:

$$\mathbf{S}_i \mathbf{S}_j = \mathbf{S}_j \mathbf{S}_i, \quad \forall \mathbf{S}_i, \mathbf{S}_j \in S \quad i \neq j.$$

2. The identity matrix multiplied by -1 is not in S:

$$-\sigma_I^{\otimes n} \notin S.$$

If $\mathbf{E}$ is an error operator, and $\mathbf{E}$ anti-commutes with some element $\mathbf{M} \in S$, then $\mathbf{E}$ can be detected, since for any $\mid \psi_i\rangle, \mid \psi_j\rangle \in Q$:

$$\mathbf{ME} \mid \psi_i\rangle = -\mathbf{EM} \mid \psi_i\rangle = -\mathbf{E} \mid \psi_i\rangle$$

If our stabilizer code is a $[n, k, d]$ code, the cardinality of the stabilizer S and its generator M are:

$$\mid S \mid = 2^{n-k}, \qquad \mid M \mid = n - k$$

The error syndrome corresponding to the stabilizer $\mathbf{M}_j$ is a function of the error operator, $\mathbf{E}$, defined as:

$$f_{\mathbf{M}_j}(\mathbf{E}) : G \mapsto \mathbf{Z}_2 \qquad f_{\mathbf{M}_j}(\mathbf{E}) = \begin{cases} 0 & \text{if } [\mathbf{M}_j, \mathbf{E}] = 0 \\ 1 & \text{if } \{\mathbf{M}_j, \mathbf{E}\} = 0. \end{cases}$$

Let $f(\mathbf{E})$ be the $(n-k)$-bit integer given by the binary vector:

$$f(\mathbf{E}) = (f_{\mathbf{M}_1}(\mathbf{E}) f_{\mathbf{M}_2}(\mathbf{E}) \ldots f_{\mathbf{M}_{n-k}}(\mathbf{E})).$$

This $(n-k)$-bit integer is called the *syndrome of error* $\mathbf{E}$.

Proposition 1: The error syndrome uniquely identifies the qubit(s) in error if and only if the subsets of the stabilizer group which anti-commute with the error operators are distinct.

An error can be identified and corrected only if it can be distinguished from any other error in the error set. Let $Q(S)$ be the stabilizer code with stabilizer S. The *Correctable Set of Errors* for Q(S) includes all errors which can be detected by S and have distinct error syndromes.

Corollary 1: Given a quantum error correcting code Q capable to correct e_u errors, the syndrome does not allow us to distinguish the case when more than e_u qubits are in error. When we have exact prior knowledge about e_c correlated errors the code is capable of correcting these $e_u + e_c$ errors.

Proof: Assume that errors F_1, F_2 cause at most e_u qubits to be in error, thus F_1, F_2 are included in the *Correctable Set of Errors* of Q. Since errors F_1 and F_2 are distinguishable, there must exist some operator $M \in S$ which commutes with one of them, and anti-commutes with the other:

$$F_1^T F_2 M = -M F_1^T F_2$$

If we knew the exact correlated errors E in the system, we have:

$$(E^T F_1)^T (E^T F_2) M = (F_1^T E E^T F_2) M = F_1^T F_2 M = -M F_1^T F_2 = -M (E^T F_1)^T (E^T F_2)$$

which means that the stabilizer M commutes with one of the two errors $E^T F_1$, $E^T F_2$ and anti-commutes with the other. So error $E^T F_1$ is distinguishable from error $E^T F_2$. Therefore, if we know the exact prior errors E, we can identify and correct any $E^T F_i$ errors with the weight of F_i equal or less than e_u.

For example, consider a 5-qubit quantum error-correcting code Q with $n = 5$ and $k = 1$. The logical codewords for Q are:

$$\begin{aligned} |\, 0_L\rangle = \tfrac{1}{4} \quad & [|\, 00000\rangle + |\, 10010\rangle + |\, 01001\rangle + |\, 10100\rangle + |\, 01010\rangle - |\, 11011\rangle - |\, 00110\rangle - |\, 11000\rangle \\ & - |\, 11101\rangle - |\, 00011\rangle - |\, 11110\rangle - |\, 01111\rangle - |\, 10001\rangle - |\, 01100\rangle - |\, 10111\rangle - |\, 00101\rangle] \end{aligned}$$

$$\begin{aligned} |\, 1_L\rangle = \tfrac{1}{4} \quad & [|\, 11111\rangle + |\, 01101\rangle + |\, 10110\rangle + |\, 01011\rangle + |\, 10101\rangle - |\, 00100\rangle - |\, 11001\rangle - |\, 00111\rangle \\ & - |\, 00010\rangle - |\, 11100\rangle - |\, 00001\rangle - |\, 10000\rangle - |\, 01110\rangle - |\, 10011\rangle - |\, 01000\rangle - |\, 11010\rangle] \end{aligned}$$

The stabilizer S of this code is described by a group of 4 generators:

$$\mathbf{M}_1 = \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I, \quad \mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x,$$

$$\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z, \quad \mathbf{M}_4 = \sigma_z \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z.$$

It is easy to see that two codewords are eigenvectors of the stabilizers with an eigenvalue of $(+1)$:

$$\mathbf{M}_j \mid 0_L\rangle = (+1) \mid 0_L\rangle \qquad \text{and} \qquad \mathbf{M}_j \mid 1_L\rangle = (+1) \mid 1_L\rangle, \quad 1 \leq j \leq 4.$$

Table 5.2: Single bit-flip and phase-flip error operators for the 5-qubit code and generators that anti-commute with.

Error operator	Generator(s) anti-commute with the error
$\mathbf{E}_1 = \sigma_x \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_4 = \sigma_z \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z$
$\mathbf{E}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_1 = \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I$
$\mathbf{E}_3 = \sigma_I \otimes \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_1 = \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I$, $\mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x$
$\mathbf{E}_4 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_x \otimes \sigma_I$	$\mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x$, $\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z$
$\mathbf{E}_5 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_x$	$\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z$, $\mathbf{M}_4 = \sigma_z \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z$
$\mathbf{E}_6 = \sigma_z \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_1 = \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I$, $\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z$
$\mathbf{E}_7 = \sigma_I \otimes \sigma_z \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x$, $\mathbf{M}_4 = \sigma_z \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z$
$\mathbf{E}_8 = \sigma_I \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I \otimes \sigma_I$	$\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z$
$\mathbf{E}_9 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I$	$\mathbf{M}_1 = \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I$, $\mathbf{M}_4 = \sigma_z \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z$
$\mathbf{E}_{10} = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_z$	$\mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x$

Note also that M is an Abelian subgroup, thus each generator commutes with all the others. Indeed we have, for example:

$$\mathbf{M}_1\mathbf{M}_3 = (\sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I)(\sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z) = \sigma_I \otimes \sigma_z \otimes (\sigma_z\sigma_x) \otimes (\sigma_x\sigma_z) \otimes \sigma_z$$

$$\mathbf{M}_3\mathbf{M}_1 = (\sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z)(\sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes \sigma_I) = \sigma_I \otimes \sigma_z \otimes (\sigma_x\sigma_z) \otimes (\sigma_z\sigma_x) \otimes \sigma_z.$$

The Pauli operators anti-commute, $\sigma_x\sigma_z = -\sigma_z\sigma_x$, thus and $\mathbf{M}_1\mathbf{M}_3 = \mathbf{M}_3\mathbf{M}_1$.

Table 5.2 lists bit-flip and phase-flip error operators and the generator(s) which anti-commute with each one. For example, $\mathbf{E}_1$ anti-commutes with $\mathbf{M}_4$, thus a bit-flip on the first qubit can be detected; $\mathbf{E}_6$ anti-commutes with $\mathbf{M}_1$ and $\mathbf{M}_3$, thus a phase-flip of the first qubit can also be detected. Since each of these ten bit-flip or phase-flip errors anti-commute with distinct subsets of S we can distinguish individual errors and then correct them. An

example shows that the code cannot detect two qubit errors; indeed, the two bit-flip error

$$\mathbf{E}_1\mathbf{E}_2 = \sigma_x \otimes \sigma_x \otimes \sigma_I \otimes \sigma_I \otimes \sigma_I$$

is indistinguishable from $\mathbf{E}_9 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I$ because both $\mathbf{E}_1\mathbf{E}_2$ and $\mathbf{E}_9$ anti-commute with the same subset of stabilizers, $\{\mathbf{M}_1, \mathbf{M}_4\}$, and give the same error syndrome. Therefore, *the 5-qubit code can correct any single qubit error, but cannot correct two qubit errors.*

If we know the correlated error, for example, $\mathbf{E_3} = \sigma_I \otimes \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_I$ in the system, from Table 5.2, it is easy to see that the errors $\mathbf{E_3E_i}$ have distinct error syndromes for $1 \leq i \leq 10$. Therefore we can identify these errors and correct them.

5.6 Other Codes

Quantum Convolutional Codes is a family of codes which are used in order to protect a stream of quantum information in a long distance communication. They are the generalization to the quantum domain of their classical analogues, and hence inherit the most important properties. They impose a specific form such that on-line encoding, decoding and error correction become possible even in the presence of an infinitely long to-be-protected stream of information. In convolutional codes, qubits are not considered as independent of each other: the encoding operation cannot be decomposed into a tensor product of encoding operations acting on a small number of qubits. By contrast, an (N, K)-block code can

protect such a stream only by cutting it into successive K-qubit blocks. Some introduction and details can be found in [Cha98, Cha99, OT03, OT04].

Quantum Codes for Burst Error: Most quantum error correction codes assume the "independent qubit decoherence"(IQD) model in which the decoherence of each qubit is uncorrelated to the decoherence of any other qubit as they all interact with separate reservoirs. If this is not true, the errors are space-correlated, typically shown as a burst of errors. Many approaches are introduced for burst error correction, such as Quantum cyclic code [VRA99], Quantum interleaver [Kaw00].

Quantum Reed-Solomon Code [GGB99]: a classical $[N; K; d]$ Reed-Solomon code over $GF(2^k)$ with length $N = 2^k - 1$, distance d, and dimension $K = N - d + 1$, is a cyclic code with the generator polynomial:

$$g(x) = (x - \beta^j)(x - \beta^{j+1} \cdots (x - \beta^{j+d-2}))$$

with β a primitive element of $GF(2^k)$ of order N.

The quantum Reed-Solomon $[N; K; d]$ code encodes k(2N-K) qubits in states $\mid \varphi_1\rangle, \mid \varphi_2\rangle \cdots \mid \varphi_{k(2N-K)}\rangle$into kN qubits.

It can be constructed as a family of Calderbank-Shor-Steane (CSS) codes.

CHAPTER 6
QUANTUM ERROR-CORRECTION FOR TIME-CORRELATED ERRORS

6.1 Time-Correlated Errors

Different physical implementations reveal that the interactions of the qubits with the environment are more complex and force us to consider spatially- as well as, time-correlated errors. If the qubits on an n-qubit register are confined to a 3D structure, an error affecting one qubit will propagate to the qubits in a volume centered around the qubit in error. Spatially-correlated errors and means to deal with the spatial noise are analyzed in [AHH02, CSG04, KF05]. An error affecting qubit i of an n-qubit register may affect other qubits of the register. An error affecting qubit i at time t and corrected at time $t + \Delta$ may have further effect either on qubit i or on other qubits of the register. The error model considered in this thesis is based upon a recent study which addresses the problem of reliable quantum computing using solid state systems [NB06].

There are two obvious approaches to deal with quantum correlated errors :

- (i) design a code capable to correct these two or more errors, or
- (ii) use the classical information regarding past errors and quantum error correcting codes capable to correct a single error.

When a quantum error correcting code uses a large number of qubits and quantum gates it becomes increasingly more difficult to carry out the encoding and syndrome measurements during a single quantum error correction cycle. For example, if we encode a logical qubit using Shor's nine qubit code we need 9 physical qubits. If we use a two-level convolutional code based upon Shor's code then we need 81 physical qubits and $63 = 7 \times 9$ ancilla qubits to ensure that the circuit for syndrome calculation is fault-tolerant. While constructing quantum codes capable of correcting a larger number of errors is possible, we believe that the price to pay, the increase in circuit complexity makes this solution undesirable; this motivates our interest in the second approach.

Figure 6.1 illustrates the evolution in time of two qubits, i and j for two error correction cycles. The first error correction cycle ends at time t_2 and the second at time t_5. At time t_1 qubit i is affected by decoherence and flips; at time t_2 it is flipped back to its original state during the first error correction step; at time t_3 qubit j is affected by decoherence and it is flipped; at time t_4 the correlation effect discussed in this section affects qubit i and flips it to an error state. If an algorithm is capable to correct one correlated error in addition to a "new" error, then during the second error correction the errors affecting qubits i and j are corrected at time t_5.

The quantum computer and the environment are entangled during the quantum computation. When we measure the state of the quantum computer this entanglement is translated

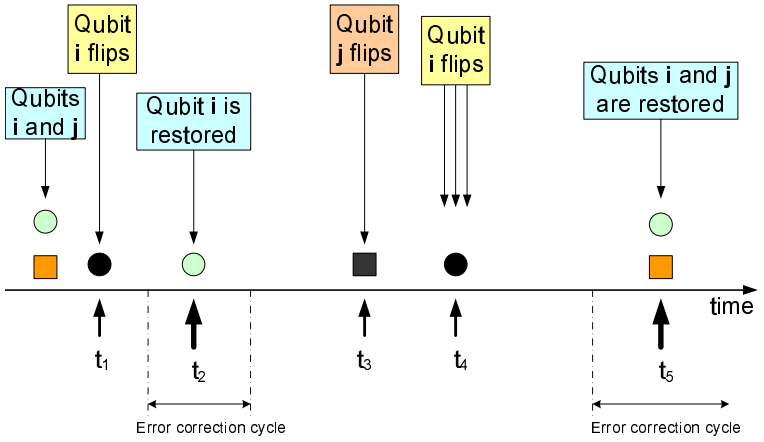

Figure 6.1: Time-correlated quantum errors. Two consecutive error correction cycles occurring at time t_2 and t_5 are shown. At time t_5 a code designed to correct a single error will fail while a code capable to handle time-correlated errors will correct the "new" error of qubit j and the "old" correlated error of qubit i.

into a probability ϵ that the measured state differs from the expected one. This probability of error determines the actual number of errors a quantum code is expected to correct.

If τ_{gate} is the time required for a single gate operation and τ_{dch} is the decoherence time of a qubit, then n_{gates}, the number of gates that can be traversed by a register before it is affected by decoherence is given by:

$$n_{gates} = \frac{\tau_{dch}}{\tau_{gate}}.$$

Quantum error correction is intimately related to the physical processes which cause decoherence. Table 6.1 presents sample values of the time required for a single gate operation τ_{gate}, the decoherence time of a qubit, τ_{dch}, and n_{gates}, the number of gates that can be

Table 6.1: The time required for a single gate operation, τ_{gate}, the decoherence time of a qubit, τ_{dch}, and the number of gates that can be traversed before a register of qubits is affected by decoherence, n_{gates}.

Qubit implementation	$\tau_{dch(sec)}$	$\tau_{gate(sec)}$	n_{gates}
Nuclear spin	10^4	10^{-3}	10^7
Trapped Indium ion	10^{-1}	10^{-14}	10^{13}
Quantum dots/charge	10^{-9}	10^{-12}	10^3
Quantum dots/spin	10^{-6}	10^{-9}	10^3
Optical cavity	10^{-5}	10^{-14}	10^9

traversed before a register of qubits is affected by decoherence, for several qubit implementations [HFC03, LOJ05, PJT05, VSB01]. We notice a fairly wide range of values for the number of quantum gate operations that can be performed before decoherence affects the state.

The information in Table 6.1, in particular the decoherence time, can be used to determine the length of an error correction cycle, the time elapsed between two consecutive error correction steps. The number of quantum gate operations limits the complexity of the quantum circuit required for quantum error correction.

The Quantum Error Correction theory is based upon the assumption that the quantum system has a constant error rate ϵ. This implies that once we correct an error at time t_c, the system behavior at time $t > t_c$ is decoupled from events prior to t_c.

The Markovian error model is violated by physical implementations of qubits; for example, in a recent paper Novais and Baranger [NB06] discuss the decoherence in a spin-boson model. The authors assume that the qubits are perfect, thus the only possible errors are due to de-phasing and consider a linear coupling to an ohmic bath. Then the Hamiltonian

of the model can be written as:

$$H = \frac{v_b}{2} \int_{-\infty}^{+\infty} dx[\partial_x \phi(x)]^2 + [\Pi(x)]^2 + \sqrt{\frac{\pi}{2}} \lambda \sum_n \partial_x \phi(n) \sigma_n^z.$$

with ϕ and $\Pi = \partial_x \theta$ canonic conjugate variables and σ_n^z the Pauli matrices acting in the Hilbert space of the qubits. v_b is the velocity of the bosonic excitations and the units are such that $\hbar = k_B = 1$. The exact time evolution between gates of a qubit, can be expressed as the product of two vertex operators of the free bosonic theory:

$$U_n(t, 0) = \exp[i\sqrt{\frac{\pi}{2}} \lambda(\theta(n, t) - \theta(n, 0))\sigma_n^z]$$

with λ the qubits bosonic coupling strength. Using this model the authors analyze the three qubit Steane code. They calculate the probability of having errors in quantum error correction cycles starting at times t_1 and t_2 and show that the probability of errors consists of two terms; the first is the uncorrelated probability and the second is the contribution due to correlation between errors in different cycles (Δ is the error correcting cycle):

$$P \approx (\frac{\epsilon}{2})^2 + \frac{\lambda^4 \Delta^4}{8(t_1 - t_2)^4}$$

We see that correlations in the quantum system decay algebraically in time, and the latest error will re-influence the system with a much higher probability than others.

Only phase-flip errors are discussed in detail in [NB06]. The approach introduced by the authors is very general and can be extended to include other types of errors. Nevertheless,

for other physical models, one does expect major departures from the results presented in [NB06].

6.2 Several Aspects of Code Building

Classical, as well as quantum error correction schemes allow us to construct codes with a well-defined error correction capability. If a code is designed to correct e errors, it will fail whenever more than e errors occur.

From previous section we know that if time-correlated errors are present in the system, we can not always detect them through the calculation of the syndrome. The same syndrome could signal the presence of a single error, two, or more errors, as shown by our example in Section 5.5; we can remove this ambiguity when there are two errors and one of them is a time-correlated error.

In this section we extend the error correction capabilities of any stabilizer code designed to correct a single error (bit-flip, phase-flip error, or bit-and-phase flip) and allow the code to correct an additional time-correlated error. The standard assumptions for quantum error correction are:

- Quantum gates are perfect and operate much faster than the characteristic response of the environment;
- The states of the computer can be prepared with no errors.

We also assume that:

- There are no spatial-correlation of errors, a qubit in error does not influence its neighbors;
- In each error correcting cycle, in addition to a new error E_a that occurs with a constant probability ε_a, a time-correlated error E_b may occur with probability $\varepsilon_b(t)$. As correlations decay in time, the qubit affected by error during the previous error correction cycle, has the highest probability to relapse.

Quantum error correction requires *non-demolition measurements of the error syndrome* in order to preserve the state of the physical qubits. In other words, a measurement of the probe (the ancilla qubits) should not influence the free motion of the signal system. The syndrome has to identify precisely the qubit(s) in error and the type of error(s). Thus, the qubits of the syndrome are either in the state with density matrix $\rho_0 =\mid 0\rangle\langle 0 \mid$ or in the state with density matrix $\rho_1 =\mid 1\rangle\langle 1 \mid$, which represent classical information.

A quantum non-demolition measurement allows us to construct the error syndrome $\Sigma_{current}$. After examining the syndrome $\Sigma_{current}$ an error correcting algorithm should be able to decide if:

1. No error has occurred; no action should be taken;
2. One "new" error, E_a, has occurred; then we apply the corresponding Pauli transformation to the qubit in error;

3. Two or more errors have occurred. There are two distinct possibilities: (a) We have a "new" error as well as an "old" one, the time-correlated error; (b) There are two or more "new" errors. A quantum error correcting code capable of correcting a single error will fail in both cases.

It is rather hard to distinguish between the last two possibilities. For perfect codes, the syndrome S_{ab} corresponding to two errors, E_a and E_b, is always identical to the syndrome S_c for some single error E_c. Thus, for perfect quantum codes the stabilizer formalism does not allow us to distinguish two errors from a single one; for a non-perfect code it is sometimes possible to distinguish the two syndromes and then using the knowledge regarding the time-correlated error it may be possible to correct both the "old" and the "new" error.

We want to construct an algorithm based on the stabilizer formalism capable to handle case 3(a) for perfect as well as non-perfect quantum codes. We assume that the quantum code has a codeword consisting of n qubits and uses k ancilla qubits for syndrome measurement. There are several aspects we need to consider carefully before we build the algorithm:

1. Duplication: since the qubit with error in the last error correction cycle may be influenced by a time-correlated error, we need to duplicate this qubit to identify the error. Furthermore, the time-correlated error may re-occur in X basis, Z basis or both. Thus we should duplicate the qubit in both X and Z basis.

2. Entanglement: in the last step we duplicate the qubit with error in last error correction cycle. Because of the non-cloning theorem, this process will entangle the ancilla with

the original codeword. Now we need to apply the syndrome measurement on the original codeword. If no corresponding actions applied on the ancilla, this syndrome measurement will destroy the codeword because of the entanglement. Therefore, to protect the codeword, the extended syndrome measurement need apply corresponding actions on the ancilla, which preserves the entanglement between the codeword and the ancilla.

3. Stabilizer: The extended syndrome measurement is built of new stabilizers: these new stabilizers (or generators) should also stabilize the codeword and satisfy all the stabilizers' requirements. Therefore, the new stabilizers should commute with all codewords, and commute with each other. Also at least one of them anti-commutes with the error we want to correct.

4. Disentanglement: After the extended syndrome measurement, we should be able to restore the original codeword - disentangle the ancilla from the codeword. This process should restore the original codeword to its initial status.

5. Distinct syndrome: Since the new code need to correct one more error than the original code, it must have distinct syndromes for different error combinations. Thus, these error combinations can be identified and corrected.

6. Error propagation: The algorithm uses some ancilla to duplicate the qubit with error in the last error correction cycle. However, these ancilla may also be influenced by

errors. These new errors introduced on the ancilla may propagate to the code qubits. This situation must be carefully considered in the algorithm.

6.3 Quantum Code for Correction of Time-Correlated Errors

6.3.1 Outline of the Algorithm

Depending on the aspects mentioned in the last section, the algorithm of quantum error correction for time-correlated errors is as follows:

1. At the end of an error correction cycle entangle the qubit affected by the error with two additional ancilla qubits. Thus, if the qubit relapses, the time-correlated error will propagate to the two additional ancilla qubits.

2. Carry out an *extended syndrome measurement* of the codeword together with the two additional ancilla qubits. This syndrome measurement should not alter the state of the codeword and keep the entanglement between the codeword and the two additional ancilla qubits intact.

3. Disentangle the two additional ancilla qubits from the codeword and then measure the two additional ancilla qubits.

4. Carry out the error correction according to the outcomes of Steps 2 and 3.

We use the Steane code to illustrate the details of the algorithm. The generators of the Steane 7-qubit code are:

$$\mathbf{M}_1 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x, \quad \mathbf{M}_2 = \sigma_I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_I \otimes \sigma_I \otimes \sigma_x \otimes \sigma_x,$$

$$\mathbf{M}_3 = \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x \otimes \sigma_I \otimes \sigma_x, \quad \mathbf{M}_4 = \sigma_I \otimes \sigma_I \otimes \sigma_I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z,$$

$$\mathbf{M}_5 = \sigma_I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_I \otimes \sigma_I \otimes \sigma_z \otimes \sigma_z, \quad \mathbf{M}_6 = \sigma_z \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I \otimes \sigma_z \otimes \sigma_I \otimes \sigma_z.$$

6.3.2 Steane Code for Correction of Time-Correlated Errors

1. Entangle the two additional ancilla qubits with the original code word. The time-correlated error may reoccur in different physical systems differently: a bit-flip could lead to a bit-flip, a phase-flip, or a bit-phase-flip. Therefore, we need to "backup" the qubit corrected during the previous error correction cycle in both **X** and **Z** basis. We entangle the qubit affected by an error with two additional ancilla qubits: a `CNOT` gate will duplicate the qubit in **Z** basis using the first ancilla qubit, the second ancilla will duplicate the qubit in **X** basis using two Hadamard gates and a `CNOT` gate (called an `HCNOT` gate), see Figure 6.2(a). In this example, qubit 3 is the one affected by error in the last error correction cycle. We duplicate qubit 3 in Z basis to ancilla qubit A, and duplicate qubit 3 in X basis to ancilla qubit B.

2. Extended syndrome measurement. We measure the extended syndrome S_{n+2} for an $(n+2)$ extended codeword consisting of the original codeword and the two extra ancilla qubit. Call Σ the result of the measurement of the extended syndrome S_{n+2}. When implementing

this extended syndrome measurement, we need to consider two aspects: (i) the entanglement between the additional ancilla and codewords should not be disturbed during the syndrome measurement, since after the measurement, we need to disentangle the extra ancilla from the codeword and keep the codeword intact; and (ii) the new stabilizers (or generators) should also stabilize the codeword and satisfy all the stabilizers' requirements. In our approach, the additional ancilla qubit **A** is a copy of the control qubit in **Z** basis. To keep the entanglement, we copy all **X** operations on the control qubit to qubit **A**. Similarly, the additional ancilla qubit **B** is a copy of the control qubit in **X** basis. We replicate all **Z** operations performed on the control qubit to this qubit. Since it is in **X** basis, the **Z** operations will change into **X** operations. Consider the example of the Steane code and the time-correlated error on qubit 3, Figure 6.2(b). In this case we replicate all **X** operations performed on qubit 3 to ancilla qubit **A**, and replicate all **Z** operations to additional ancilla qubit **B** as **X** operations. These operations will keep the entanglement intact during the syndrome measurement process. Also, it is easy to check that the new stabilizers (i) commute with each other, and (ii) stabilize the codewords.

3. Disentangle and measure the two additional ancilla qubits. After the syndrome measurement, we use a circuit similar to the one in Figure 6.2(a) to disentangle the two additional ancilla qubits from the codeword. The gates in this circuit are in reverse order as in Figure 6.2(a). Since the entanglement is not affected during the extended syndrome measurement, this process will disentangle the extra ancilla from the codewords and keep

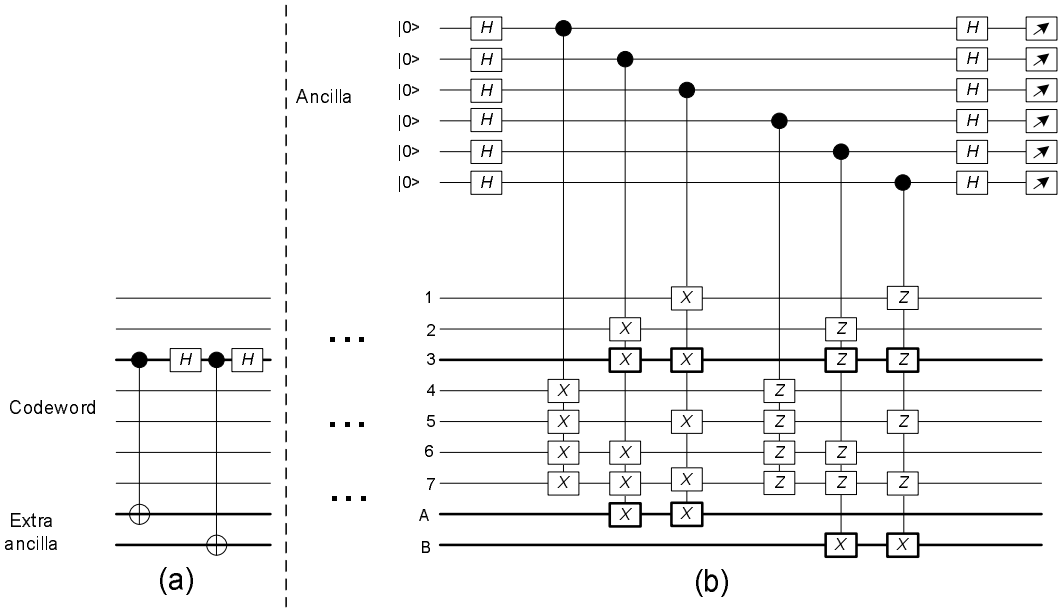

Figure 6.2: (a) Duplicate the qubit affected error during the last cycle in both **X** and **Z** basis; (b) Extended syndrome measurement on both the codeword and the additional ancilla qubits.

the codeword in its original state. If qubit 3 is affected by the time-correlated error, bit-flip, phase-flip or both, the disentanglement circuit will propagate these errors to the extra ancilla qubits: a `CNOT` gate will propagate a bit-flip of the control qubit to the target, and an `HCNOT` gate will propagate a phase-flip of the control qubit to the target qubit as a bit-flip, Figure 6.3. Therefore, measuring the additional ancilla qubits after the disentanglement will give us the information regarding the type of error qubit 3 has relapsed to.

4. Correct the errors according to the output of the extended syndrome measurement and of the measurement of the two additional ancilla qubits. Consider the following scenarios:

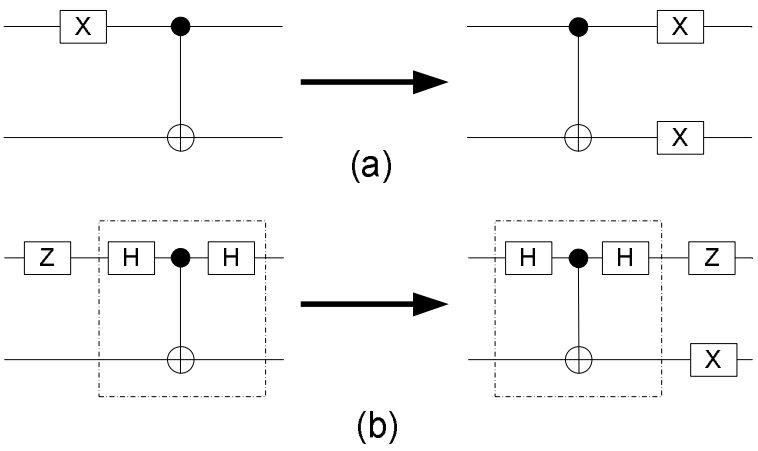

Figure 6.3: (a) A `CNOT` gate propagates a bit-flip of the control qubit to the target qubit (b) An `HCNOT` gate propagates a phase-flip on the control qubit to the target qubit as a bit-flip.

1. A "new" error affects a qubit i of the codeword, while qubit $j, j \neq i$ was the one in error during the previous cycle. Then the measurement of the extended syndrome produces the same outcome as the measurement of the original syndrome and the new error can be identified and corrected.

2. The time-correlated error of qubit j occurs and there is no "new" error. The error propagates to the two ancilla qubits. The result of the extended syndrome measurement is the same as the original one, which allows us to identify and correct the single error.

3. There are two errors, the time-correlated error of qubit j and a "new" error affecting qubits $i, i \neq j$ of the codeword. Then the extended syndrome measurement gives the combination of these two errors: $\Sigma = \Sigma_j \oplus \Sigma_i$. The measurement of the two additional ancilla qubits reveals the type of the time-correlated error: *10* – bit-flip, *01* – phase-

flip, *11* – bit-and-phase-flip, see Figure 6.3. Knowing that qubit j is in error and knowing the corresponding error type, we use the syndrome table and find the new error on qubit i. We can identify and correct these two errors individually.

4. There is a single "new" error affecting one of the two extra ancilla qubits. This error may propagate to the qubit of the codeword the ancilla is entangled with. We treat this scenario as (1) since the syndrome measurement can identify the propagated error. For example, in Figure 6.2, a phase-flip of the ancilla qubit **A** propagates to qubit 3. The measurement of the original syndrome allows us to identify this error on qubit 3 and correct it.

5. There are two errors, the time-correlated error on qubit j of the codeword and an error on the additional ancilla entangled with qubit j. As we mention in scenario (3), the error on the extra ancilla may back propagate to qubit j. The net result is that in the codeword, only the qubit j is affected by error: bit-flip, phase-flip or bit-and-phase-flip. This error is indicated by the syndrome measurement and can be corrected as in scenario (1).

The error correction process can be summarized as follows:

1. If the syndrome Σ indicates an error on qubit j, the one affected by error during the last cycle (qubit 3 in our example), correct the error;

2. Else, determine the outcome of a measurement of the two additional ancilla qubits:

(a) If the outcome is *00*, and Σ corresponds to a single error syndrome, correct this error;

(b) If the outcome is either *01*, or *10*, or *11*, qubit j in error during the previous cycle has relapsed. Then the syndrome is $\Sigma_i = \Sigma \otimes \Sigma_j$. Knowing that qubit j is in error and the corresponding error type (*10* – bit-flip, *01* – phase-flip, *11* – bit-and-phase-flip) we use the syndrome table to find the new error affecting qubit i.

(c) Otherwise, there are two or more new errors in the system and they cannot be corrected.

Example 1: Consider again the Steane seven-qubit code. Table 6.2 shows the syndromes for all single errors for the Steane 7-qubit quantum code. Assume that during the last error correction cycle qubit $j = 3$ was affected by an error. We use a `CNOT` gate to entangle qubit 3 with the additional ancilla qubit **A** in **Z** basis, and an `HCNOT` gate to entangle it with qubit **B** in **X** basis, Figure 6.2.

1. $\Sigma = 011000$, indicates a phase-flip on qubit 3. We correct this error. In this case the measurement of the additional ancilla qubit could be non-zero, but this will not affect the codeword.

Table 6.2: The syndromes for each of the three types of errors of each qubit of a codeword for the Steane 7-qubit code: X_{1-7} bit-flip, Z_{1-7} phase-flip, and Y_{1-7} bit-and-phase flip.

Error	Syndrome	Error	Syndrome	Error	Syndrome
X_1	000001	Z_1	001000	Y_1	001001
X_2	000010	Z_2	010000	Y_2	010010
X_3	000011	Z_3	011000	Y_3	011011
X_4	000100	Z_4	100000	Y_4	100100
X_5	000101	Z_5	101000	Y_5	101101
X_6	000110	Z_6	110000	Y_6	110110
X_7	000111	Z_7	111000	Y_7	111111

2. $\Sigma = 110000$ and the outcome of the measurement of the two additional ancilla qubits is 00. In this case there is only one "new" error in the system, a phase-flip on qubit $i = 6$ as indicated by Σ. We correct this error.

3. $\Sigma = 110000$ and the outcome of the measurement of the additional ancilla qubits is 01. In this case $\Sigma_j = \Sigma_{phase-flip} = \Sigma_{Z3} = 011000$. Therefore, the new error has syndrome $\Sigma_i = 110000 \otimes 011000 = 101000$; this indicates a phase-flip of qubit $i = 5$. Correct the phase-flips on qubit 3 and qubit 5.

The circuits described in this thesis have been tested using the Stabilizer Circuit Simulator, called ***GraphSim*** [AB06].

GraphSim is a stabilizer circuits simulator based on the graph-state formalism. The usual proof of the Gottesman-Knill theorem contains an algorithm which carry out this task in time $O(N^3)$, where N is the number of qubits. Aaronson and Gottesman presented an algorithm and its implementation in [AG04], whose time and space requirements scale only

quadratically with the number of qubits. *GraphSim* gives a further improvement by presenting an algorithm that for typical applications only requires time and space of $O(N \log N)$. This provides a valuable tool for investigating complex circuits as in our study.

The simulator is written in C++ and the implementation is done as a library to allow for easy integration into other projects. The authors also offer bindings to Python, so that the library can be used by Python programs as well.

Sample codes(in Python) for testing the algorithm for correction of Time-correlated errors are provided in the Appendix. In this sample, we use Steane's 7-qubit code and qubit 3 was in error in the last error-correction cycle. One can test different error scenarios using a code similar to the one in the Appendix.

6.4 Summary of Quantum Error-Correction for Time-Correlated Errors

Errors affecting the physical implementation of quantum circuits are not independent, often they are time-correlated. Based on the properties of time-correlated noise, we present an algorithm that allows the correction of one time-correlated error in addition to a new error. The algorithm can be applied to any stabilizer code when the two logical qubits $\mid 0_L\rangle$ and $\mid 1_L\rangle$ are entangled states of 2^n basis states in H_{2^n}.

The algorithms can be applied to perfect as well as non-perfect quantum codes. The algorithm requires two additional ancilla qubits entangled with the qubit affected by an error during the previous error correction cycle. The alternative is to use a quantum error

correcting code capable of correcting two errors in one error correction cycle; this alternative such as a concatenate code requires a considerably more complex quantum circuit for error correction.

APPENDIX
SAMPLE CODE OF THE STABILIZER CIRCUIT SIMULATOR

```
# Sample code for testing the algorithm for correction of
# Time-correlated errors using the stabilizer circuit simulator
# GraphSim

# Requires GraphSim from
# http://homepage.uibk.ac.at/~c705213/work/graphsim.html

# INPUT: Prepare the stabilizer circuits by using cnot, hadamard,
# cphase gates and so on. Introduce errors by using bitflip and
# phaseflip operations.

# OUTPUT: Output the syndrome by printing the measurements on
# different ancilla qubits.

import graphsim

import random random.seed()
gr=graphsim.GraphRegister(22,random.randint(0,1E6))

# Prepare Steane |0> codeword on qubit 0-6

gr.hadamard (4)
gr.hadamard (5)
gr.hadamard (6)
gr.cnot (6, 3)
gr.cnot (6, 1)
gr.cnot (6, 0)
gr.cnot (5, 3)
gr.cnot (5, 2)
gr.cnot (5, 0)
gr.cnot (4, 3)
gr.cnot (4, 2)
gr.cnot (4, 1)

# Entangle qubit 3 with the extra ancilla gr.cnot(2,7)
gr.hadamard(2)
gr.cnot(2,8)
gr.hadamard(2)

# Errors happen on qubit ?
```

```
gr.bitflip(2)
gr.phaseflip(3)

##### Begin current error correction cycle #####

# "Extended" Syndrome extraction

for i in range(9,15):
      gr.hadamard(i)

# Z syndrome
gr.cnot(9,3)
gr.cnot(9,4)
gr.cnot(9,5)
gr.cnot(9,6)
gr.cnot(10,1)
gr.cnot(10,2)
gr.cnot(10,5)
gr.cnot(10,6)
gr.cnot(10,7)
gr.cnot(11,0)
gr.cnot(11,2)
gr.cnot(11,4)
gr.cnot(11,6)
gr.cnot(11,7)

# X syndrome
gr.cphase(12,3)
gr.cphase(12,4)
gr.cphase(12,5)
gr.cphase(12,6)
gr.cphase(13,1)
gr.cphase(13,2)
gr.cphase(13,5)
gr.cphase(13,6)
gr.cnot(13,8)
gr.cphase(14,0)
gr.cphase(14,2)
gr.cphase(14,4)
gr.cphase(14,6)
gr.cnot(14,8)
```

```
# "Disentangle" the extra ancilla
gr.hadamard(2)
gr.cnot(2,8)
gr.hadamard(2)
gr.cnot(2,7)

# Measure and output the extra ancilla

print gr.measure(7)
print gr.measure(8)

#gr.print_adj_list()
#gr.print_stabilizer()
print

# Measure and output syndrome
for i in range(9,15):
  gr.hadamard(i)
  print gr.measure(i)
print

##############   error correction  ###############

# Error correction according to the syndrome and extra ancilla
gr.bitflip(2)
gr.phaseflip(3)

##### Normal syndrome extraction to double-check  #####

for i in range(9+6,15+6):
  gr.hadamard(i)

# Z syndrome
gr.cnot(9+6,3)
gr.cnot(9+6,4)
gr.cnot(9+6,5)
gr.cnot(9+6,6)

gr.cnot(10+6,1)
```

```
gr.cnot(10+6,2)
gr.cnot(10+6,5)
gr.cnot(10+6,6)

gr.cnot(11+6,0)
gr.cnot(11+6,2)
gr.cnot(11+6,4)
gr.cnot(11+6,6)

# X syndrome
gr.cphase(12+6,3)
gr.cphase(12+6,4)
gr.cphase(12+6,5)
gr.cphase(12+6,6)

gr.cphase(13+6,1)
gr.cphase(13+6,2)
gr.cphase(13+6,5)
gr.cphase(13+6,6)

gr.cphase(14+6,0)
gr.cphase(14+6,2)
gr.cphase(14+6,4)
gr.cphase(14+6,6)

# Measure and output the syndrome
for i in range(9+6,15+6):
  gr.hadamard(i)
  print gr.measure(i)
print
```

LIST OF REFERENCES

[AAK01] D. Aharonov, A. Ambainis, J. Kempe, and U. V. Vazirani. "Quantum walks on graphs." In *ACM Symposium on Theory of Computing*, pp. 50–59, 2001.

[Aar05] S. Aaronson. "Quantum Computing, Postselection, and Probabilistic Polynomial-Time." *Proceedings of the Royal Society A*, 461(2063):3473–3482, December 2005.

[AB06] S. Anders and H. J. Briegel. "Fast Simulation of Stabilizer Circuits Using a Graph-state Representation." *Phys. Rev. A*, 73(2):022334–+, February 2006.

[AG04] S. Aaronson and D. Gottesman. "Improved Simulation of Stabilizer Circuits." *Physical Review A (Atomic, Molecular, and Optical Physics)*, 70(5):052328, 2004.

[AHH02] R. Alicki, M. Horodecki, P. Horodecki, and R. Horodecki. "Dynamical Description of Quantum Computing: Generic Nonlocality of Quantum Noise." *Phys. Rev. A*, 65(6):062101–+, June 2002.

[Amb07] A. Ambainis. "Quantum Walk Algorithm for Element Distinctness." *SIAM Journal on Computing*, 37(1):210–239, 2007.

[BBB92] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. "Experimental Quantum Cryptography." *J. Cryptol.*, 5(1):3–28, 1992.

[BBB97] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. "Strengths and Weaknesses of Quantum Computing." *SIAM J. Comput.*, 26(5):1510–1523, 1997.

[BBC93] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. "Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels." *Phys. Rev. Lett.*, 70(13):1895–1899, Mar 1993.

[BDE95] A. Barenco, D. Deutsch, A. Ekert, and R. Jozsa. "Conditional Quantum Dynamics and Logic Gates." *Phys. Rev. Lett.*, 74(20):4083–4086, May 1995.

[BHM02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. "Quantum Amplitude Amplification and Estimation." *AMS Contemporary Mathematics*, 305:53–74, 2002.

[BHT98] G. Brassard, P. Høyer, and A. Tapp. "Quantum Counting." *Lecture Notes in Computer Science*, 1443:820+, 1998.

[BPM97] D. Bouwmeester, J. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger. "Experimental Quantum Teleportation." *Nature*, 390(6660):575–579, December 1997.

[CFH96] D. Cory, A. Fahmy, and T. Havel. "Nuclear Magnetic Resonance Spectroscopy: an Experimentally Accessible Paradigm for Quantum Computing.", 1996.

[Cha98] H. F. Chau. "Quantum Convolutional Error-correcting Codes." *Phys. Rev. A*, 58(2):905–909, Aug 1998.

[Cha99] H. F. Chau. "Good Quantum-convolutional Error-correction Codes and Their Decoding Algorithm Exist." *Phys. Rev. A*, 60:1966–1974, September 1999.

[CP89] J. Carlier and E. Pinson. "An Algorithm for Solving the Job-shop Problem." *Manage. Sci.*, 35(2):164–176, 1989.

[CSG04] J. P. Clemens, S. Siddiqui, and J. Gea-Banacloche. "Quantum error correction against correlated noise." *Physical Review A (Atomic, Molecular, and Optical Physics)*, 69(6):062313, 2004.

[CZ95] J. I. Cirac and P. Zoller. "Quantum Computations with Cold Trapped Ions." *Phys. Rev. Lett.*, 74(20):4091–4094, May 1995.

[Div00] D. P. Divincenzo. "The Physical Implementation of Quantum Computation." *Fortschritte der Physik*, 48:771–783, 2000.

[Eke91] A. K. Ekert. "Quantum Cryptography Based on Bell's Theorem." *Physical Review Letters*, 67:661–663, August 1991.

[FKL03] M. H. Freedman, A. Kitaev, M. J. Larsen, and Z. Wang. "Topological Quantum Computation." *Bull. Amer. Math. Soc.*, 40(1):31–38, 2003.

[Fur06] B. Furrow. "A Panoply of Quantum Algorithms." *ArXiv Quantum Physics e-prints*, June 2006.

[GC97] N. A. Gershenfeld and I. L. Chuang. "Bulk Spin-Resonance Quantum Computation." *SCIENCE: Science*, 275, 1997.

[GGB99] M. Grassl, W. Geiselmann, and T. Beth. "Quantum Reed-Solomon Codes." In *AAECC-13: Proceedings of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pp. 231–244, London, UK, 1999. Springer-Verlag.

[Gos98] P. Gossett. "Quantum Carry-Save Arithmetic." *ArXiv Quantum Physics e-prints*, August 1998.

[Got97] D. Gottesman. "Stabilizer Codes and Quantum Error Correction.", 1997.

[GR05] L. K. Grover and J. Radhakrishnan. "Is Partial Quantum Search of a Database Any Easier?" In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pp. 186–194, New York, NY, USA, 2005. ACM Press.

[Gro96] L. K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search." In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, New York, NY, USA, 1996. ACM Press.

[Gro97] L. K. Grover. "Quantum Mechanics Helps in Searching for a Needle in a Haystack." *Phys. Rev. Lett.*, 79(2):325–328, Jul 1997.

[Gro98] L. K. Grover. "A Framework for Fast Quantum Mechanical Algorithms." In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 53–62, New York, NY, USA, 1998. ACM Press.

[Gro05] L. K. Grover. "Fixed-Point Quantum Search." *Physical Review Letters*, 95(15):150501, 2005.

[GSU04] A. Grigoriev, M. Sviridenko, and M. Uetz. "Unrelated Parallel Machine Scheduling with Resource Dependent Processing Times." Research Memoranda 033, Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 2004. available at http://ideas.repec.org/p/dgr/umamet/2004033.html.

[HFC03] T. Hayashi, T. Fujisawa, H. D. Cheong, Y. H. Jeong, and Y. Hirayama. "Coherent Manipulation of Electronic States in a Double Quantum Dot." *Phys. Rev. Lett.*, 91(22):226804, Nov 2003.

[Ho00] P. Høyer. "Arbitrary Phases in Quantum Amplitude Amplification." *Phys. Rev. A*, 62(5):052304, Oct 2000.

[IAB99] A. Imamoglu, D. D. Awschalom, G. Burkard, D. P. DiVincenzo, D. Loss, M. Sherwin, and A. Small. "Quantum Information Processing Using Quantum Dot Spins and Cavity QED." *Phys. Rev. Lett.*, 83(20):4204–4207, Nov 1999.

[Kaw00] S. Kawabata. "Quantum Interleaver: Quantum Error Correction for Burst Error." *Journal of the Physical Society of Japan*, 69:3540–+, November 2000.

[Kem03] J. Kempe. "Quantum Random Walks: An Introductory Overview." *Contemporary Physics*, 44:307–327, April 2003.

[KF05] R. Klesse and S. Frank. "Quantum Error Correction in Spatially Correlated Quantum Noise." *Physical Review Letters*, 95(23):230503, 2005.

[KG06] V. E. Korepin and L. K. Grover. "Simple Algorithm for Partial Quantum Search." *Quantum Information Processing*, 5:5, 2006.

[LD98] D. Loss and D. P. DiVincenzo. "Quantum Computation with Quantum Dots." *Phys. Rev. A*, 57(1):120–126, Jan 1998.

[LOJ05] C. Langer, R. Ozeri, J. D. Jost, J. Chiaverini, B. DeMarco, A. Ben-Kish, R. B. Blakestad, J. Britton, D. B. Hume, W. M. Itano, D. Leibfried, R. Reichle, T. Rosenband, T. Schaetz, P. O. Schmidt, and D. J. Wineland. "Long-Lived Qubit Memory Using Atomic Ions." *Physical Review Letters*, 95(6):060502, 2005.

[LST90] J. K. Lenstra, D. B. Shmoys, and E. Tardos. "Approximation Algorithms for Scheduling Unrelated Parallel Machines." *Math. Program.*, 46(3):259–271, 1990.

[MI05] R. V. Meter and K. M. Itoh. "Fast Quantum Modular Exponentiation." *Physical Review A (Atomic, Molecular, and Optical Physics)*, 71(5):052320, 2005.

[MM04] D. C. Marinescu and G.M. Marinescu. *Approaching Quantum Computing*. Prentice Hall, Upper Saddle River, NJ, 2004.

[NB06] E. Novais and H. U. Baranger. "Decoherence by Correlated Noise and Quantum Error Correction." *Physical Review Letters*, 97(4):040501, 2006.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum Computing and Quantum Information*. Cambridge University Press, 2000.

[OT03] H. Ollivier and J. Tillich. "Description of a Quantum Convolutional Code." *Phys. Rev. Lett.*, 91(17):177902, Oct 2003.

[OT04] H. Ollivier and J. . Tillich. "Quantum Convolutional Codes: Fundamentals." *ArXiv Quantum Physics e-prints*, January 2004.

[PJT05] J. R. Petta, A. C. Johnson, J. M. Taylor, E. A. Laird, A. Yacoby, M. D. Lukin, C. M. Marcus, M. P. Hanson, and A. C. Gossard. "Coherent Manipulation of Coupled Electron Spins in Semiconductor Quantum Dots." *Science*, 309:2180–2184, September 2005.

[Pre98] J. Preskill. *Introduction to Quantum Computation and Information*, chapter 8. World Scientific, 1998.

[PV01] M. B. Plenio and V. Vitelli. "The Physics of Forgetting: Landauer's Erasure Principle and Information Theory." *Contemporary Physics*, 42:25–60, January 2001.

[RG05] B. W. Reichardt and L. K. Grover. "Quantum Error Correction of Systematic Errors Using a Quantum Search Framework." *Physical Review A (Atomic, Molecular, and Optical Physics)*, 72(4):042326, 2005.

[SGG02] D. Stucki, N. Gisin, O. Guinnard, G. Ribordy, and H. Zbinden. "Quantum Key Distribution over 67 km with a Plug and Play System." *New Journal of Physics,*, 4(41):1–8, 2002.

[Sho94] P. W. Shor. "Algorithms for Quantum Computation: Discrete Log and Factoring." In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134. Institute of Electrical and Electronic Engineers Computer Society Press, 1994.

[Sho95] P. W. Shor. "Scheme for Reducing Decoherence in Quantum Computer Memory." *Phys. Rev. A*, 52(4):R2493–R2496, Oct 1995.

[Sho97] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sho03] P. W. Shor. "Why Haven't More Quantum Algorithms Been Found?" *J. ACM*, 50(1):87–90, 2003.

[Sim97] D. R. Simon. "On the Power of Quantum Computation." *SIAM J. Comput.*, 26(5):1474–1483, 1997.

[SKW03] N. Shenvi, J. Kempe, and K. B. Whaley. "Quantum Random-walk Search Algorithm." *Phys. Rev. A*, 67(5):052307, May 2003.

[SS02] A. S. Schulz and M. Skutella. "Scheduling Unrelated Machines by Randomized Rounding." *SIAM J. Discret. Math.*, 15(4):450–469, 2002.

[Ste96] A. M. Steane. "Error Correcting Codes in Quantum Theory." *Phys. Rev. Lett.*, 77(5):793–797, Jul 1996.

[Tan00] T. Tanamoto. "Quantum Gates by Coupled Quantum Dots and Measurement Procedure in Field-Effect-Transistor Structure." *Fortschritte der Physik*, 48(9-11):1005 – 1021, 2000.

[TGP06] T. Tulsi, L. K. Grover, and A. Patel. "A New Algorithm for Fixed Point Quantum Search." *Quantum Information and Computation*, 6(6):483–494, September 2006.

[THL95] 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.*, 75(25):4710–4713, Dec 1995.

[TS06] E.M. Rains T.G. Draper, S.A. Kutin and K.M. Svore. "A Logarithmic-depth Quantum Carry-lookahead Adder." *Quantum Information and Computation*, 6(4):351–369, 2006.

[VRA99] F. Vatan, V.R. Raichowdry, and M. Anantram. "Spatially Correlated Qubit Errors and Burst-Correcting Quantum Codes." *IEEE Transactions on Information Theory*, 45:1703–1708, 1999.

[VSB01] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. "Experimental Realization of Shor's Quantum Factoring Algorithm Using Nuclear Magnetic Resonance." *Nature*, 414:883–887, December 2001.

[Zal99] C. Zalka. "Grover's Quantum Searching Algorithm Is Optimal." *Phys. Rev. A*, 60(4):2746–2751, Oct 1999.