Fault-tolerant quantum computer architectures using hierarchies of quantum error-correcting codes

by

Andrew W. Cross

B.S., Case Western Reserve University (2002)
S.M., Massachusetts Institute of Technology (2005)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author ....
Department of Electrical Engineering and Computer Science
May 9th, 2008

Certified by ....
Isaac L. Chuang
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Fault-tolerant quantum computer architectures using hierarchies of quantum error-correcting codes

by

Andrew W. Cross

Submitted to the Department of Electrical Engineering and Computer Science
on May 22nd, 2008, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Quantum computers have been shown to efficiently solve a class of problems for which no efficient solution is otherwise known. Physical systems can implement quantum computation, but devising realistic schemes is an extremely challenging problem largely due to the effect of noise. A quantum computer that is capable of correctly solving problems more rapidly than modern digital computers requires some use of so-called fault-tolerant components. Code-based fault-tolerance using quantum error-correcting codes is one of the most promising and versatile of the known routes for fault-tolerant quantum computation. This dissertation presents three main, new results about code-based fault-tolerant quantum computer architectures.

The first result is a large new family of quantum codes that go beyond stabilizer codes, the most well-studied family of quantum codes. Our new family of codeword stabilized codes contains all known codes with optimal parameters. Furthermore, we show how to systematically find, construct, and understand such codes as a pair of codes: an additive quantum code and a classical (nonlinear) code.

Second, we resolve an open question about universality of so-called transversal gates acting on stabilizer codes. Such gates are universal for classical fault-tolerant computation, but they were conjectured to be insufficient for universal fault-tolerant quantum computation. We show that transversal gates have a restricted form and prove that some important families of them cannot be quantum universal. This is strong evidence that so-called quantum software is necessary to achieve universality, and, therefore, fault-tolerant quantum computer architecture is fundamentally different from classical computer architecture.

Finally, we partition the fault-tolerant design problem into levels of a hierarchy of concatenated codes and present methods, compatible with rigorous threshold theorems, for numerically evaluating these codes. The methods are applied to measure inner error-correcting code performance, as a first step toward elucidation of an effective fault-tolerant quantum computer architecture that uses no more than a physical, inner, and outer level of coding. Of the inner codes, the Golay code gives the highest pseudothreshold of 2×10^{-3}. A comparison of logical error rate and overhead shows that the Bacon-Shor codes are competitive with Knill's C_4/C_6 scheme at a base error rate of 10^{-4}.

Thesis Supervisor: Isaac L. Chuang
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

Very little of the work I have done in graduate school would be possible without the support and encouragement of advisors, colleagues, collaborators, friends, and family. There isn't enough space here to thank anyone appropriately.

Thanks foremost to my advisor Isaac Chuang for giving me a chance in his group, guiding me when I was unsure, and supporting the directions I chose. He created opportunity for me to learn and live in several academic communities and challenged me to grow as an engineer and scientist. Thank you also to my doctoral committee members: Al Aho for encouragement when I most needed it and insight always, David DiVincenzo for providing research direction and being an inspiring person and scientist (I cannot thank David enough), and G. David Forney for teaching an excellent class and sharing his perspective and wisdom.

Many thanks to the IBM Physics of Information group for hosting me from May 2006 to December 2007. Special thanks to David DiVincenzo, John Smolin, and Barbara Terhal for guiding my research, advising me, and sharing their thoughts and ideas freely. Thank you to Fred Mintzer for providing use of Blue Gene L.

Special thanks to the following people: Panos Aliferis, whose approach I greatly respect, and from whom I learned much about the joys of research; Tzvetan Metodi for his courage to communicate our ideas to the computer architecture community; Krysta Svore for encouraging me to build our software tools and for involving me in her work at IBM Research; Bei Zeng for her cleverness, energy, and willingness to work with me on problems of quantum coding theory; Colleagues in lab for helping me appreciate what it means to actually do a quantum gate, especially Rob Clark and Jarek Labaziewicz. I highly respect and am thankful for the time spent speaking with and learning from colleagues: Charlie Bennett, Sergey Bravyi, Ken Brown, Xie Chen, Fred Chong, Hyeyoun Chung, Dean Copsey, Aram Harrow, Francois Impens, Roger Koch, John Kubiatowicz, Igor Markov, Rod van Meter, Mark Oskin, Graeme Smith, Matthias Steffen, Darshan Thaker, and Terri Yu. I have appreciated a lot of help from Marilyn Pierce in the EECS Graduate Office.

I have been funded by a National Defense Science and Engineering Graduate Fellowship, the QuARC project under a DARPA Quantum Information Science and Technology grant, an internship at IBM Research, and the Urabe CREST Project of the Japan Science and Technology Agency.

This thesis is dedicated to my wife Abby, my parents, and my teachers; particularly Michael Branicky, Karin Ricci, Jeffrey Wilson, and Wojbor Woyczynski.

Contents

Chapter 1

Introduction

1.1 Prologue

Modern digital computers are realizations of mathematical objects called universal Turing machines, originally defined in 1936 [Tur37]. Turing machines have a tape on which to read and write symbols. A read-write head reads the current symbol from the tape and decides what to do next based on a transition rule. Given the current symbol and internal state, the rule indicates what symbol to write to the tape and whether or not to move the head left or right. A *universal* Turing machine can simulate any other Turing machine when a description of that machine is initially written on the tape.

Despite the fact that modern digital computers do not look much like the Turing machines that inspired their creation, each can simulate the other efficiently. The broader idea that *any* physical model of computation can be efficiently simulated on a (probabilistic) Turing machine is called the Strong Church-Turing Thesis [BV93]. There is no way to prove this important thesis, since it is difficult to apprehend what "physical" means in a mathematical sense, but we may agree that it is either true or false. Significantly, our experiences suggest that this thesis is true.

Quantum Turing machines are mathematical models of computing machines that are in many respects similar to (probabilistic) Turing machines [BV93, Yao93]. However, quantum Turing machines may branch and run separate programs weighted by "probability amplitudes" that are given by complex numbers rather than usual probabilities. The amplitudes are chosen in such a way that the quantum Turing machine is time-reversible; information about its past computations is never lost. A quantum computer is a realization of a quantum Turing machine.

Quantum computing is a challenge to the Strong Church-Turing Thesis because quantum computers have been shown to efficiently solve a class of problems for which no efficient solution is otherwise known. The class of problems contains factoring as a special case [Sho94], which quantum computers can solve with complexity slightly better than $O(n^3)$. In contrast, the best known algorithm on a classical computer is called the number field sieve (NFS). The NFS has complexity $O\left(exp\left[(\log n)^{1/3}(\log\log n)^{2/3}\right]\right)$ [CP01], which is not a polynomial function of the number of digits of input, and therefore not efficient.

One reason for current interest in realizing a quantum computer is that large quantum computers break important parts of modern cryptography. The ability to solve problems called hidden subgroup problems that contain factoring and the discrete logarithm as special cases allows us to break elliptic-curve and RSA-public-key cryptosystems and solve other number theoretic problems as well. For example, the classical complexity of encrypting a message using the RSA-public-key cryptosystem is comparable to the quantum complexity of finding the key on a quantum computer, so it appears likely that increasing the key size in this case is a rather weak defense against an adversary with a quantum computer.

In addition to the spectacular successes with hidden subgroup problems, quantum computers are also known to exhibit a number of polynomial speedups over probabilistic classical computers. The most well-known algorithm is Grover's search algorithm, which gives a quadratic speedup for searching an unstructured database [Gro96]. For a few more examples, a quadratic speedup of classical simulated annealing is possible [SBBK08], as is a cubic speedup for the problem of solving exponential congruences [vS08]. Quantum computers probably cannot solve general unstructured search problems in polynomial time, however. This is the same as saying that quantum computers cannot efficiently solve NP-complete problems, and there is evidence (but no proof) that this is true [BBBV97].

Small quantum computers may also have advantages over modern digital computers. They may be able to compute physical quantities that are beyond the reach of classical methods by simulating dynamics of many-body quantum systems, for example [BDL07]. Small quantum computers may also assist in already-practical quantum cryptographic protocols, serving as quantum repeaters, or in other protocols that need small quantum memories or simple operations by the sender or receiver.

Though the computational motivations for quantum computing are sufficient, it is important to mention an exciting fundamental motivation. Some argue that if quantum computers cannot be engineered for what turns out to be a fundamental reason, then the postulates of quantum mechanics, the most thoroughly exercised modern physical theory, may need to be revised [Aar04]. On the other hand, if quantum computers can be built, quantum mechanics will be confirmed in the strongest sense on macroscopic scales comparable with our otherwise classical world.

Physical systems can implement quantum computation, but devising realistic schemes for implementing quantum computation is an extremely challenging problem. The principal challenge, however, is not finding physical systems that have the right kind of state space, nor is it finding a sequence of interactions to implement an adequate set of quantum logic gates. The principal challenge is noise.

Noise in quantum mechanical systems has several basic forms, but arguably the most damaging form of noise is caused by uncontrollable interactions between the quantum system and its surrounding environment. This raises a dilemma. On the one hand, we would like to isolate the quantum system completely so that it cannot interact with anything without our knowledge. On the other, we want to reach in and absolutely control how the quantum system evolves and computes, and, ultimately, couple the system to the environment to extract the final classical output of the computer. What actually occurs in experiments appears to be somewhere in between these extremes of perfect isolation and absolute control.

Optimistic estimates of noise strength, as measured by a quantity called the average or worst-case gate infidelity, are presently no smaller than 10^{-6} for a sufficient set of gates to realize quantum computation. The most difficult, but necessary, experimentally realized gates presently have infidelities closer to 10^{-3} [BKRB08]. Therefore, optimistically, the output appears almost uniformly random due to noise and other inaccuracy after no more than roughly one million of these gates. Yet, the most spectacular applications require computations to run for one trillion or more basic steps to solve problems presently out of reach of digital computers. This may seem like a big number but modern digital computers can execute this many basic steps in about an hour (and we know that servers can run for years without crashing).

Therefore, a quantum computer that is capable of correctly solving problems more rapidly than modern digital computers requires some use of so-called fault-tolerant components. Fault-tolerant components continue to compute the correct function despite weak noise within themselves. They do so without an extravagant increase in circuit size or a severe decrease in computation speed.

In the early history of computing, the vacuum tube technology for building classical logic gates was very unreliable, so John von Neumann, a father of modern computing, proposed the following solution [vN56]. His idea was to encode the computer's state by copying it several times and to never decode it (until the very end of the computation). Periodically, he would apply a voting circuit that took the majority value of each bundle of wires and reset the bundle to that value. This scheme was the first method for realizing fault-tolerant classical components.

Shorly afterwards, the transistor replaced the vacuum tube as a much more reliable technology. In some sense, the transistor leads to reliable gates because the bit value is encoded into the state of many electrons. The technology restores the bit value toward zero or one during each gate, so it can be considered naturally fault-tolerant. Therefore, fault-tolerant techniques like von Neumann's have only had limited use in modern computing (so far).

There are several visions for how to realize fault-tolerant quantum components (see [CFP02, Kit03, Sho96]). It is not obvious a-priori that fault-tolerant quantum computation is actually possible – it is thoroughly remarkable that quantum mechanics allows it. The proposed methods can be hewn roughly along a line between those that provide physical fault-tolerance like the transistor and those that provide code-based fault-tolerance like von Neumann's solution.

Unfortunately, the existing proposals for physical fault-tolerance are inadequately supported by experiment, at present. For one interesting and beautiful example, Alexei Yu Kitaev has proposed a brilliant scheme for physical fault-tolerance using exotic states of condensed matter systems [Kit03]. This scheme encodes quantum information into highly non-local degrees of freedom that cannot be damaged by local interactions but can nevertheless be transformed by braiding quasi-particles. However, the necessary states of matter have not been conclusively observed, although there continue to be promising developments [DHU$^+$08].

On the other hand, von Neumann-like approaches using quantum error-correcting codes are one of the most promising and versatile of the known routes for fault-tolerant quantum computation [ABO99, AGP06, AGP08, SDT07]. Indeed, the theory of quantum codes is rich [Got97, CRSS98, CS96, AC08, KKKS06, RHSS97], and small quantum codes have been experimentally shown to correct errors [KLMN01]. Furthermore, remarkable theorems have been proven for a variety of different types of noise showing that efficient fault-tolerant computation using codes is possible as long as the noise strength is less than a constant that does not depend on the size of the computation [TB05, AKP06].

Several *physical* resources, whose specifics depend on quantum codes, are needed within a fault-tolerant quantum computer. Massive parallelism is necessary to continually correct errors, and this requires physical computational units that can act simultaneously. The number of units depends on the particular choice of codes. Access to low noise states is also necessary, so these states and the physical hardware that creates them must be made available relatively near their point of use. The specific states and hardware also depend on the choice of codes. Finally, careful layout of hardware elements is needed to avoid excessive faults from long range interactions. The detail of this layout will depend on the code choice, and codes whose associated hardware can be implemented with minimal data movement may be desirable. All of these physical resources are *necessary* for successful fault-tolerant quantum computation using quantum codes [ABO99]. Therefore, the structure of fault-tolerant components, and hence of the system microarchitecture of a quantum computer, is strongly impacted by the choice of quantum error-correcting codes; see Figure 1-1.

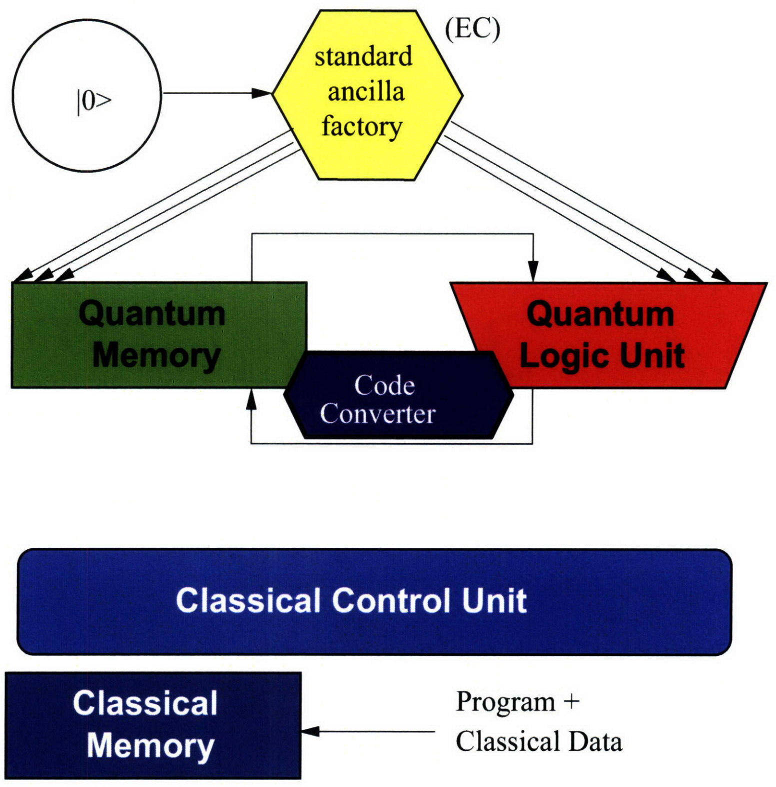


Figure 1-1: This is a stored-program architecture for a fault-tolerant quantum computer. The program is stored in a classical memory together with classical data. Quantum data is stored in a quantum memory. The separation of memory and computation also allows use of efficient codes for each purpose [TMC+06], although a code converter is then needed in the datapath. Although we may imagine a sequential machine at the logical level, the microarchitecture is necessarily parallel; perhaps single-instruction-multiple-data (SIMD) [MTC+05b, MTC+05a, COI+03]. The figure also illustrates the necessary source of low noise ancilla states. These begin in some standard state $|0\rangle$ and are encoded into many standard code-specific ancilla states by the standard ancilla factory. This factory must be implemented in a careful way within each processing unit at the microarchitecture level, to avoid excessive faults as the ancilla are communicated to their point of use.

1.2 Questions addressed by this dissertation

The new results in this dissertation address three questions spanning the domain of fault-tolerant quantum computation. This section presents these three broad, motivating questions and discusses the significance of each.

1.2.1 How do we systematically find new quantum codes?

Motivated by quantum fault-tolerance, and the overarching importance of quantum codes in fault-tolerant quantum computer architecture, the first broad question this dissertation addresses is simply: *How do we systematically find new quantum codes?*

The largest and most important family of quantum codes is the family of stabilizer codes. These quantum codes are analogous in many ways to classical linear codes. The quantum circuits for working with stabilizer codes can be efficiently constructed and have polynomial size. Efficient decoders and error-correcting algorithms exist if the classical code underlying the stabilizer code can be efficiently decoded. Stabilizer codes can be systematically found using techniques for finding classical codes.

However, stabilizer codes do not always have optimal parameters, so it is natural to wonder how to find, construct, and understand codes with optimal parameters. Furthermore, it is extremely desirable to develop quantum coding theory along the lines of modern classical coding theory so that insights from modern coding theory can be used to construct families of quantum codes. In particular, a correspondence between classical nonlinear codes and quantum codes has been almost entirely absent from quantum computing literature.

1.2.2 How does quantum code structure relate to fault-tolerant gate structure?

Fault-tolerant quantum components are designed to compute on information encoded in a quantum code. Codes provide a shape and form that must be preserved by fault-tolerant components. Therefore, the structure of a quantum code relates directly to the structure of a fault-tolerant gate on that code.

What we call a *standard approach* to fault-tolerance (see Chapter 5) includes a well-known kind of fault-tolerant component called a transversal gate. This kind of fault-tolerant gate is sufficient for universal classical fault-tolerant computing, but the same statement does not appear to be true for quantum fault-tolerance. Prior to our work, however, the structure of transversal gates on quantum codes was not well understood, so the truth of this statement could not be resolved one way or the other.

This uncertainty about the structure of such fundamental fault-tolerant components as transversal gates motivates our next broad question: *How does the structure of a quantum code relate to the structure of a fault-tolerant set of gates for that code?* This question may lead us to identify essential architectural components and may provide a clearer understanding of what kinds of gates can easily be made fault-tolerant. Indeed, some of the more exotic transversal gates already find clever application in universal gate set constructions, but only a few such code examples are well-known. New codes we may unearth, with different exotic gates, might find application to fault-tolerance as well.

1.2.3 How can codes in a fault-tolerant system be evaluated and compared?

Designing a code-based fault-tolerant quantum computer is a daunting task. Clearly the task must be partitioned into managable pieces with established interfaces between them. Our view is that concatenation, a method for combining codes, is an essential concept for quantum fault-tolerance [For66]. We believe that an effective architecture will likely use multiple codes due to the high noise rates observed in physical systems. This view makes the problem more complicated, but suggests one way to partition it, motivating the question: *How can we rigorously evaluate and fairly compare code performance within a hierarchy of concatenated codes, without building and/or simulating the entire system? What results can we expect from such an evaluation?* These questions may lead us to a clearer picture both of how to design such a complex system and what the design space "looks like"; i.e., are there a great many codes that are good for this task, or is the palette limited?

1.3 Structure and organization of this dissertation

This dissertation is organized into three parts corresponding to what we might see panning across a computer's architecture; see Figure 1-2 for a diagram of how the different chapters and sections relate to one another. Part I gives models of devices, which are organized into components in Part II. The components are then integrated into fault-tolerant systems in Part III.

Chapters 2 and 3 in Part I provide perspective and background so that Parts II and III can be appreciated. The new results of this dissertation are contained in Parts II and III. Specifically, Chapters 4 and 6 in Part II give new results concerning codes and fault-tolerant components.

Chapter 5 provides the necessary background to motivate our discussion of fault-tolerant gates in Chapter 6. Finally, in Part III, Chapters 7 and 8 give new results about fault-tolerant systems. The dissertation concludes with Chapter 9.

Chapter 4 introduces a new family of quantum codes, called codeword-stabilized (CWS) codes, and a framework for systematically identifying and understanding these codes. The CWS codes include another well-known, interesting, and important family of codes called stabilizer codes. Furthermore, CWS codes include non-stabilizer codes that have very good and sometimes optimal code parameters. The framework is general enough that special noise models may be considered. Our hope is that CWS codes can be developed in future work along similar lines as stabilizer codes, so that small CWS codes and the coincident framework find application in fault-tolerance and defeating specific types of noise. This part of the dissertation is based on joint work with Isaac Chuang, Graeme Smith, John Smolin, and Bei Zeng.

Chapter 6 presents new results about the structure of transversal gates on stabilizer codes. In particular, we are able to build on previously known results about so-called linear stabilizer codes to describe the form of elements of the full automorphism group of any stabilizer code. We are also able to describe the form of transversal gates on stabilizer codes. This allows us to resolve a long-standing conjecture about transversality and universality in the special case of some kinds of transversal gates on stablilizer codes. This part of the dissertation is based on joint work with Bei Zeng and Isaac Chuang.

Chapter 7 presents a view of the role of quantum codes in an effective fault-tolerant system architecture. In particular, given the high noise rates in experimental systems, concatenation is viewed as an essential concept (despite the fact that it is not strictly necessary). This leads to the notion of a hierarchy of codes. Within this hierarchy, noise will likely be defeated by multiple specialized codes that achieve high thresholds and desired logical error rates with a minimum of overhead. The chapter then shows how to apply the Aliferis-Gottesman-Preskill (AGP) method to numerically evaluate fault-tolerant components built from larger quantum codes than the original method could evaluate. The main observations are that the AGP method can be applied in at least two interesting Monte-Carlo settings and that errors from faults in preceeding parts of a circuit can effectively be ignored for the three principal types of error-correcting circuits.

Chapter 8 presents constructions and results of a survey of inner codes for fault-tolerant quantum computation. The survey uses the methods of Chapter 7 to evaluate systems and compare codes fairly and rigorously, within the models we have assumed. These results show that the Bacon-

Shor codes and the Golay code have desirable thresholds and overheads that remain competitive when compared to non-standard error-detection-based fault-tolerance schemes. Many other codes are less favorable, however, suggesting that the palette of inner codes for fault-tolerant quantum computing is limited. The chapter also includes a thorough review of fault-tolerant computation on quantum polynomial codes. The review presents observations about these codes that do not appear elsewhere in the literature. This part of the dissertation is based on joint work with David DiVincenzo and Barbara Terhal.

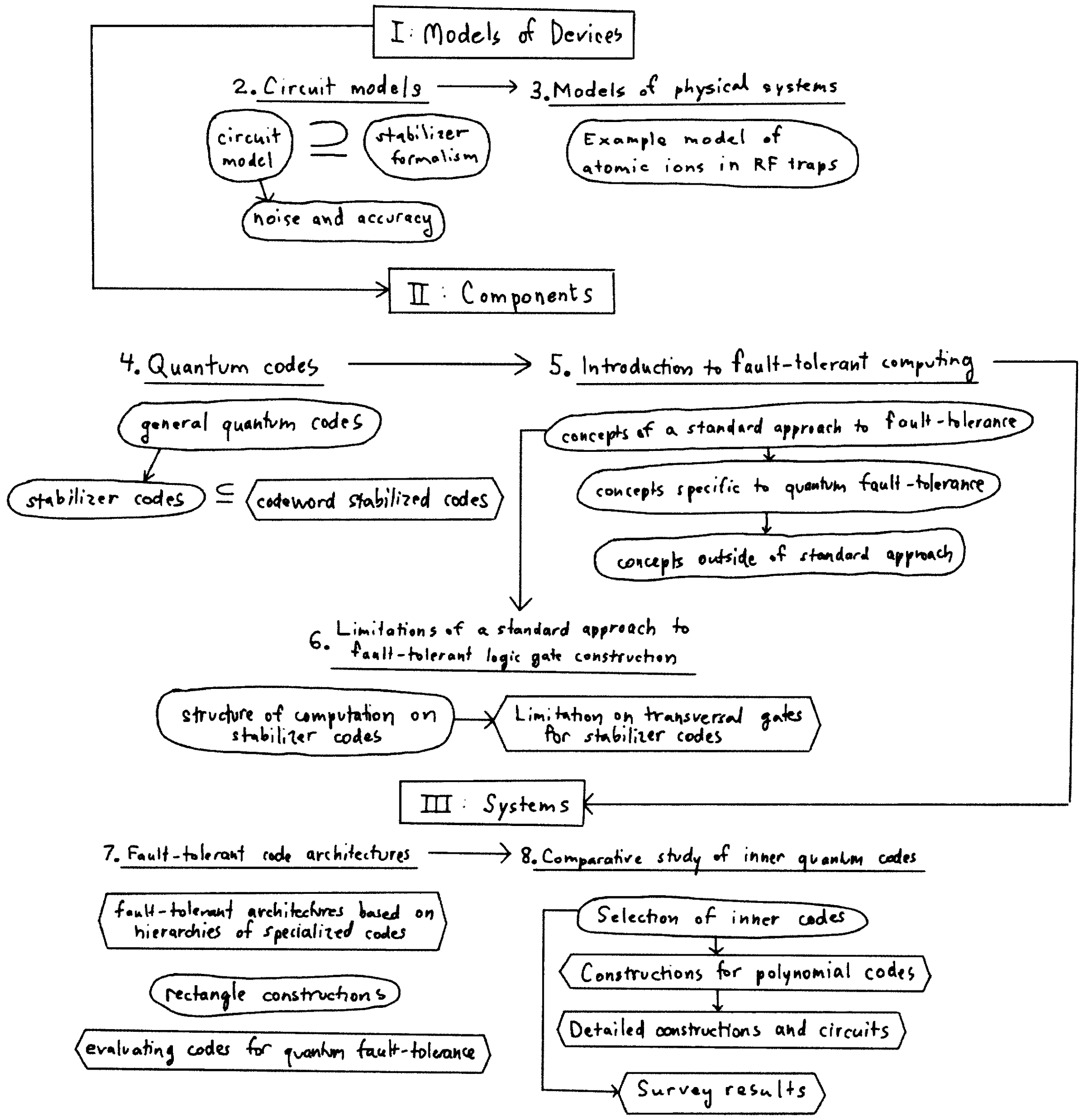

I: Models of Devices
2. Circuit models
3. Models of physical systems
circuit model
stabilizer formalism
Example model of atomic ions in RF traps
noise and accuracy
II: Components
4. Quantum codes
5. Introduction to fault-tolerant computing
general quantum codes
stabilizer codes
codeword stabilized codes
concepts of a standard approach to fault-tolerance
concepts specific to quantum fault-tolerance
concepts outside of standard approach
6. Limitations of a standard approach to fault-tolerant logic gate construction
structure of computation on stabilizer codes
Limitation on transversal gates for stabilizer codes
III: Systems
7. Fault-tolerant code architectures
8. Comparative study of inner quantum codes
fault-tolerant architectures based on hierarchies of specialized codes
rectangle constructions
evaluating codes for quantum fault-tolerance
Selection of inner codes
Constructions for polynomial codes
Detailed constructions and circuits
Survey results

Figure 1-2: How this dissertation is organized. Sections containing new results have hexagonal borders and arrows indicate what sections are prerequisites for others.

Part I

Models of Devices

Chapter 2

The circuit model of quantum computation

2.1 Introduction

This chapter introduces the postulates of quantum mechanics and how these postulates define elements of a circuit model for quantum computation in physical systems. The circuit model can describe idealized quantum computation in some physical systems. It is an adequate abstract foundation for later study of quantum computing components and systems beginning in Chapter 4.

In the second part of this chapter we review the stabilizer formalism. The stabilizer formalism is a way to describe and manipulate an important class of highly entangled quantum states. Many of the circuits that appear later in this dissertation transform stabilizer states to other stabilizer states. Furthermore, new contributions of this dissertation explore limitations and extensions of the stabilizer formalism. A crucial aspect of stabilizer circuits that we use in Chapters 7 and 8 is that they can be efficiently simulated on a classical computer.

The final section of this chapter introduces models of noise and inaccuracy in quantum systems. Noise is a significant phenomenon that acts to destroy quantum coherence in experimentally realized systems.

2.2 The quantum circuit model

This section introduces a model of ideal quantum bits (qubits), gates, and measurements. These elements are connected to form a quantum circuit that is in some ways analogous to a digital circuit. Gates are generated by time evolution under a Hamiltonian that describes physics of a system. The Hamiltonian and its parameters are selected and engineered to perform a number of basic gates which are composed to build circuits.

The elements of the circuit model mirror the fundamental postulates of quantum mechanics. These postulates describe the space of accessible states, how those states evolve in time, and how measurements on those states are described. The postulates are simple and consistent with decades of diverse experiments.

2.2.1 Qubits and states

A physical system represents a **qubit** if the system's state can be described as belonging to a particular space, and if it can take any state in that space.

A **qubit's state** is a unit vector in a complex 2-dimensional Hilbert space $\mathcal{H}$. When there are a finite number of dimensions, being a **Hilbert space** simply means that an inner product $\langle\psi_1|\psi_2\rangle$ is defined between any two vectors $|\psi_1\rangle$ and $|\psi_2\rangle$ of $\mathcal{H}$. The set $\{|0\rangle, |1\rangle\}$ is a fixed orthonormal basis for $\mathcal{H}$, meaning that $\langle i|i\rangle = 1$ and $\langle i|j\rangle = 0$ if $i \neq j$, and is called the **computational basis**. A state is written as $|\psi\rangle = a|0\rangle + b|1\rangle$ over this basis. The complex numbers a and b are called **amplitudes**, and, for states, they satisfy a normalization condition $||\psi|| := \sqrt{|a|^2 + |b|^2} = 1$ since states are unit vectors. The inner product between states $|\psi_1\rangle = a|0\rangle + b|1\rangle$ and $|\psi_2\rangle = c|0\rangle + d|1\rangle$ is $\langle\psi_1|\psi_2\rangle = a^*c + b^*d$, so $||\psi|| := \sqrt{\langle\psi|\psi\rangle}$.

The corresponding postulate of quantum mechanics is

> **Postulate 1** [NC00]: Associated to any isolated physical system is a Hilbert space known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.

A qubit's space is a special case of this general postulate.

Classical bits have two distinct states labeled 0 and 1. Qubits can be in analogous states $|0\rangle$ and $|1\rangle$ and also in more subtle superposition states such as $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$. Each complex number $a = a_r + ia_c$ has two real components, so a state vector $|\psi\rangle$ has four real components. Since the norm is constrained to be 1, three free real angles γ, θ, and ϕ determine the state of a single qubit $|\psi\rangle = e^{i\gamma}\left(\cos\theta/2|0\rangle + e^{i\phi}\sin\theta/2|1\rangle\right)$. The angle γ is a **global phase** that cannot be physically observed, as we will see in Section 2.2.3. Neglecting γ, the angles θ and ϕ identify a point on the surface of a sphere called the **Bloch sphere**. On the Bloch sphere, $|0\rangle$ is the North pole, $|1\rangle$ is the South pole, and states $(|0\rangle + e^{i\phi}|1\rangle)/\sqrt{2}$ lie on the equator; see Figure 2-1.

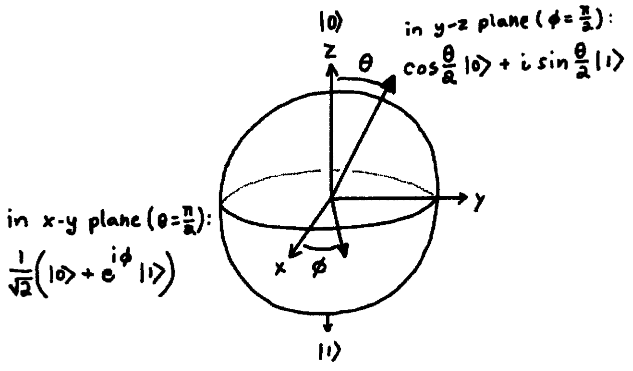

$|0\rangle$
in y-z plane ($\phi=\frac{\pi}{2}$):
$\cos\frac{\theta}{2}|0\rangle + i\sin\frac{\theta}{2}|1\rangle$
z
θ
y
in x-y plane ($\theta=\frac{\pi}{2}$):
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\phi}|1\rangle\right)$
x
ϕ
$|1\rangle$

Figure 2-1: Bloch sphere.

The state of several classical bits is given by a direct product of the state of each bit. The state of n bits is written as a string $b_1b_2\dots b_n$ in the space of all n bit strings $\{0,1\}^n$. There are 2^n such strings in the whole universe of classical states, but any particular state is described by n values.

The state of n qubits is more subtle because the state belongs to a tensor product space $\mathcal{H}^{\otimes n}$ that contains states that cannot be described by a list of amplitudes a_i and b_i for each of n isolated qubits. The tensor product space is a 2^n dimensional Hilbert space spanned by the orthonormal basis $\{|b_1b_2\dots b_n\rangle\}$ labeled by n-bit strings. For example, the state $(|00\rangle+|11\rangle)/\sqrt{2}$ is a reasonable 2-qubit state, as is $|0\rangle\otimes\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)$. There is no simple analog of the Bloch sphere for n qubits.

Quantum state vectors that can be written as a tensor product $\bigotimes_i |\psi_i\rangle$ are said to be **unentangled**. Unentangled states can be described by $2n$ amplitudes. A state that cannot be described this way is **entangled**. For example, the state $(|00\rangle+|11\rangle)/\sqrt{2}$ is an entangled state called a **Bell pair** (or a **two qubit cat state**). Neither qubit of a Bell pair can be considered in isolation without losing some information about its state.

2.2.2 Unitary gates

An ideal single qubit **quantum gate** is a 2×2 unitary matrix that acts by left multiplication on states. Unitary matrices

$$U = \begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} \tag{2.1}$$

have complex matrix elements that satisfy $U^\dagger U = I$ where $U^\dagger$ is the transpose conjugate

$$(U^T)^* = \begin{pmatrix} a^* & -b \\ b^* & a \end{pmatrix} \tag{2.2}$$

of U and I is the identity matrix. These matrices form an abstract group $U(2)$ under matrix multiplication. A **group** is a set of elements together with a binary law of composition such that (a) the set contains an identity element, (b) the law composition is associative, and (c) each element of the set has an inverse in the set. The norm of states is preserved by unitary matrices. An ideal gate on n qubits is a $2^n \times 2^n$ unitary matrix in the group $U(2^n)$.

The corresponding postulate of quantum mechanics says precisely this:

> **Postulate 2** [NC00]: The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U that depends only on the times t_1 and t_2, where $|\psi'\rangle = U|\psi\rangle$.

A physical statement of this postulate associates the dynamics of a quantum system with an operator that generates the time evolution. This operator, the **Hamiltonian**, describes the energy of the system.

> **Postulate** $2'$ [NC00]: The time evolution of the state of a closed quantum system is described by the **Schrodinger equation,** $i\hbar\frac{d|\psi\rangle}{dt} = H|\psi\rangle$. In this equation, $\hbar$ is a physical constant known as Planck's constant whose value must be experimentally determined. H is a fixed Hermitian operator known as the Hamiltonian of the closed system.

A Hermitian operator H satisfies $H = H^\dagger$, so its eigenvalues are real. The spectral decomposition of the Hamiltonian $H = \sum_E E|\psi_E\rangle\langle\psi_E|$ gives the eigenstates $|\psi_E\rangle$ with energy eigenvalue E. The matrix $|\psi_E\rangle\langle\psi_E|$ is the **outer product** of the state vector $|\psi_E\rangle$ with itself. The solution to Schrodinger's equation is a unitary operator $U(t_1, t_2) := e^{-iH(t_2-t_1)/\hbar}$, making the connection between the two statements of Postulate 2.

A set of gates G is **exactly universal** if the unitary groups $U(2^n)$ can be generated by taking products of gates in G and a global phase $e^{i\theta}$ for all $n \geq 1$. If a gate $g[q_1, q_2, \dots, q_m] \in G$ is defined on an ordered list of input qubits $\{q_1, q_2, \dots, q_m\}$, we permit the gate to act on any subset of the total set of n qubits, arranged in any order. The gate acts like the identity on qubits outside that subset. Two-level unitary gates, which are gates that act non-trivially on less than 3 vector components, are exactly universal. The set of single qubit gates $U(2)$ together with the **controlled-NOT (CNOT) gate**

$$\Lambda(X) := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

$$= |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \tag{2.4}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \oplus \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} =: I \oplus X =: \Lambda(X)[1,2] \tag{2.5}$$

is another exactly universal set. For example, the **Toffoli gate** $\Lambda^2(X)$, a classically universal reversible gate that flips the third input if the first and second inputs are one [Tof80], can be constructed over the basis $\{U(2), CNOT\}$ as shown in Figure 2-2 [NC00]. The proofs that these sets are exactly universal are given in [NC00, KSV02].

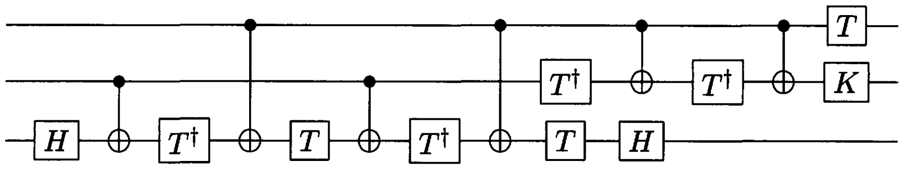


Figure 2-2: A Toffoli gate $\Lambda^2(X)$ can be constructed from CNOT and single qubit gates. The CNOT is drawn as a "$\cdot$" connected to "$\oplus$" by a wire. H, T, and K are single qubit gates.

One problem with exact universality is that there are some unitary gates in D dimensions that cannot be decomposed into products of fewer than $D-1$ two-level unitary gates. However, by approximating the unitary gate, it is possible to efficiently decompose it as a product of gates from a finite basis. We will review the basic definitions and one main result from [KSV02].

In a Hilbert space $\mathcal{H}$, there is a norm $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$. The **norm of an operator** A in the space of linear operators $\mathbf{L}(\mathcal{H})$ on $\mathcal{H}$ is

$$\|A\| := \sup_{|\psi\rangle\neq 0} \frac{\|A|\psi\rangle\|}{\||\psi\rangle\|}. \tag{2.6}$$

An operator $\tilde{U}$ is said to **approximate** U with precision ϵ if $||\tilde{U} - U|| \leq \epsilon$. An important fact is that approximation errors accumulate linearly

$$||\tilde{U}_m \tilde{U}_{m-1} \ldots \tilde{U}_1 - U_m U_{m-1} \ldots U_1|| \leq \sum_{i=1}^{m} \epsilon_i. \tag{2.7}$$

The operator U on n qubits is **approximated using ancillas** by the operator $\tilde{U}$ on $N > n$ qubits with precision ϵ if, for any n-qubit state $|\psi\rangle$,

$$||\tilde{U}(|\psi\rangle \otimes |0^{N-n}\rangle) - U|\psi\rangle \otimes |0^{N-n}\rangle|| \leq \epsilon |||\psi\rangle||. \tag{2.8}$$

Theorem 1 (Efficient approximation over a standard finite basis[KSV02]) *Any unitary operator U on a fixed number of qubits can be realized with precision ϵ by a polylog(ϵ^{-1}) size, polyloglog(ϵ^{-1}) depth circuit over the basis*

$$\left\{ H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, K := diag(1, i), K^{-1}, \Lambda(X), \Lambda^2(X) \right\} \tag{2.9}$$

using ancillas. There is a polynomial algorithm that constructs this circuit on the description of U.

H is called the **Hadamard** gate and K is called the **Phase** gate. Theorem 1 essentially holds for any basis $\mathcal{A}$ that is complete, and that result is known as the Solovay-Kitaev theorem (and associated algorithm). A basis $\mathcal{A}$ is **complete** if it is closed under inversion and the application of its elements generate a dense subgroup in the group of unitary gates on $k \geq 2$ qubits modulo a global phase. The set

$$\{H, K, \Lambda(X), T := \mathrm{diag}(1, e^{i\pi/4})\} \tag{2.10}$$

is also an important complete basis. T is called the **$\pi/8$-gate**, since $T = e^{i\pi/8}\mathrm{diag}(e^{-i\pi/8}, e^{i\pi/8})$.

2.2.3 Quantum measurements

The measurement postulate is

> **Postulate 3** [NC00]: Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If that state of the quantum system is $|\psi\rangle$ immediately before the mea-

surement, then the probability that result m occurs is given by $p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$, and the state of the system after measurement is $\frac{M_m|\psi\rangle}{\sqrt{p(m)}}$. The measurement operators satisfy the completeness condition $\sum_m M_m^\dagger M_m = I$.

The measurement postulate is perhaps a strange postulate – if the quantum system is enlarged to include the measurement device, then why is the enlarged system not described by Posulate 2? This question has bothered some people for a long time, but we pragmatically accept that Postulate 3 describes another kind of operation that can be applied in a quantum circuit.

The simplest ideal measurement to describe is a projective measurement in the basis $\{|0\rangle, |1\rangle\}$ on one qubit. This is a measurement described by $\{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$ with two possible outcomes $m \in \{0, 1\}$. If the state of a single qubit is $a|0\rangle + b|1\rangle$, the probability of outcome m is $\langle m|\psi\rangle$. This is $|a|^2$ for $m = 0$ and $|b|^2$ for $m = 1$. The post-measurement state is $|m\rangle$. The probabilities for each outcome are not given by these simple expressions when the qubit is entangled with another system, but are instead as given in Postulate 3. This particular measurement is called a Z-**basis measurement**.

2.2.4 Quantum circuits

A **quantum circuit** is an acyclic directed graph with the following interpretation [Deu89, Yao93]. Vertices represent gates drawn from a complete basis and edges represent wires carrying qubits from the output of one gate to the input of the next. Some vertices are marked as input vertices and the input qubits are in some initial state $|\psi\rangle$. Qubits may be measured in the computational basis (i.e. Z-basis) and the measurement outcomes may be used to control whether or not a later quantum gate is applied. Figure 2-3 shows how some gates are drawn in a quantum circuit. An example quantum circuit with some generic gates is shown in Figure 2-4.

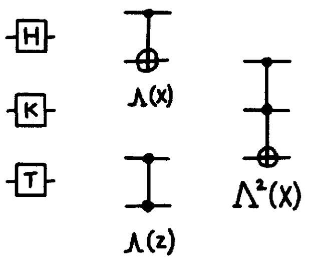


Figure 2-3: Each of the standard gates has a schematic representation as shown in this figure.

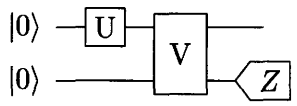


Figure 2-4: An example of a quantum circuit. Our convention is that time flows from left to right so the wires do not need to be labeled with arrows. U is a one qubit gate and V is a two qubit gate. The box labeled Z is a Z-basis measurement.

In later chapters, we may relax this definition, for example, to allow other kinds of measurements.

2.3 The stabilizer formalism

The stabilizer formalism was devised by Gottesman [Got97]. It provides a concise description for a large class of highly entangled states and subspaces. The formalism also allows these states to be easily transformed amongst themselves, and it provides insight into quantum teleportation and other circuits constructed from gates and measurements in the formalism. It is an essential framework for thinking about many areas of quantum information science. We use it frequently through this dissertation.

2.3.1 Applied group theory

The Pauli group

The **Pauli group** $\mathcal{G}_1$ is the group generated by (a) operators X and Z that satisfy the commutation relation $ZX = -XZ$ and (b) the complex phase i,

$$\mathcal{G}_1 := \langle X, Z, iI\rangle = \{\pm 1, \pm i\} \times \{I, X, Y := ZX(iI)^3, Z\}. \tag{2.11}$$

If the group were only generated by X and Z it would be the quaternion group [Art91]. The Pauli group has a 2-dimensional representation in terms of Pauli matrices where

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.12}$$

The computational basis states of a qubit are

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.13}$$

so Z can be thought of as a "phase flip" and X a "bit flip".

There are several important properties of the group and its elements to keep in mind. The order $|\mathcal{G}_1|$ of the Pauli group is 16. The matrices and their products are unitary so they can be applied as quantum gates. Those that have eigenvalues ± 1, such as $\pm X$, $\pm Y$, and $\pm Z$, are Hermitian, so they are quantum mechanical observables. The non-identity elements are traceless. Furthermore, the elements are orthogonal in the Hilbert-Schmidt inner product, meaning that $\mathrm{Tr}(P_i P_j) = \delta_{ij}$ for elements P_i and P_j, and the complex span of them equals the set of 2×2 complex matrices.

The n-**qubit Pauli group** $\mathcal{G}_n$ is the n-fold tensor product of $\mathcal{G}_1$,

$$\mathcal{G}_n := \mathcal{G}_1^{\otimes n} = \{\pm 1, \pm i\} \otimes \underbrace{\{I, X, Y, Z\} \otimes \cdots \otimes \{I, X, Y, Z\}}_{n \text{ times}}. \tag{2.14}$$

For example, $iZXZII$ is an element of $\mathcal{G}_5$ written in shorthand notation where two Pauli elements, written side by side, mean $P_1 P_2 := P_1 \otimes P_2 = (P_1 \otimes I)(I \otimes P_2)$ for $P_1, P_2 \in \mathcal{G}_1$. The phase is usually factored to the left of the product of Pauli matrices. The intended operation, group multiplication or tensor product, is usually clear from context. The order of this group is 4^{n+1} since each of the n terms in the tensor product can take 4 values and the complex phase can take 4 values. An element with ± 1 phase is self-inverse (Hermitian and unitary). An element $\pm iP$ generates a subgroup $\{iP, -I, -iP, I\}$ and is not a quantum mechanical observable. From the fact $\mathrm{Tr}(P \otimes Q) = \mathrm{Tr}(P)\,\mathrm{Tr}(Q)$, it follows that the n-qubit Pauli matrices modulo their phase are an orthonormal basis for the $2^n \times 2^n$ matrices over the complex field.

The n-qubit Pauli group is generated by $2n + 1$ elements

$$\mathcal{G}_n = \langle iI, X_1, X_2, \ldots, X_n, Z_1, Z_2, \ldots, Z_n \rangle \tag{2.15}$$

that we will call the **standard generating set** of $\mathcal{G}_n$. Here, P_i denotes the element with a Pauli P at the ith term and identity elsewhere, e.g., $X_1 := XII \ldots I$. As a shorthand, let

$$[n] := \{1, 2, \ldots, n\} \tag{2.16}$$

denote the positive integers up to and including n, since this set will appear in several places. Recall that the commutator bracket of two elements is $[P_a, P_b] := P_aP_b - P_bP_a$ and the anticommutator bracket is $\{P_a, P_b\} := P_aP_b + P_bP_a$. Two elements commute if $[P_a, P_b] = 0$ and anticommute if $\{P_a, P_b\} = 0$. The generators of $\mathcal{G}_n$ satisfy $[X_i, X_j] = 0$ and $[Z_i, Z_j] = 0$ for all $i, j \in [n]$. They also satisfy $[Z_i, X_j] = 0$ for all $i \neq j$ in $[n]$ and $\{Z_i, X_i\} = 0$ for all $i \in [n]$. This makes clear that two elements $P_a, P_b \in \mathcal{G}_n$ either commute or anticommute, since $[AB, C] = 0$ if $[A, C] = [B, C] = 0$ or $\{A, C\} = \{B, C\} = 0$, and $\{AB, C\} = 0$ if $[A, C] = \{B, C\} = 0$ or $\{A, C\} = [B, C] = 0$.

The Pauli group elements can be expressed in a way that is amenable to binary matrix computations. There is a map binary$(g) : G_n \to \{0, 1\}^{2n}$ given by binary(I) = [0|0], binary(X) = [1|0], binary(Z) = [0|1], and binary(Y) = [1|1] and binary$(P_1 \otimes P_2)$ = (binary(P_1), binary(P_2)). For example binary$(XZYI)$ = [1010|0110]. The map is a **homomorphism**, meaning that it preserves the group operation binary(P_1P_2) = binary(P_1) + binary(P_2), i.e., matrix multiplication becomes addition modulo 2. The phases of Pauli elements are dropped, so we can write binary$(X^{\mathbf{a}}Z^{\mathbf{b}}) = [\mathbf{a}|\mathbf{b}]$ where "|" just separates the two halves of the binary string. This map is called the **homomorphism from a Pauli group to a binary vector space**. $cX^{\mathbf{a}}Z^{\mathbf{b}}$ and $dX^{\mathbf{a}'}Z^{\mathbf{b}'}$ commute iff $(\mathbf{a}|\mathbf{b}) \odot (\mathbf{a}'|\mathbf{b}') := \mathbf{a}\mathbf{b}' - \mathbf{b}\mathbf{a}' = 0$. This inner product is called the **symplectic inner product**.

The Clifford Group

The quantum gates in a group called the Clifford group are very important within the stabilizer formalism. The **Clifford groups** are defined in the following way in quantum computing literature [Got97, BRW61, Bol61]

$$\mathcal{C}_2^{(n)} := \{U \in U(2^n) \mid UgU^\dagger \in \mathcal{G}_n \text{ for all } g \in \mathcal{G}_n\}. \tag{2.17}$$

This is the normalizer subgroup of the Pauli group in the unitary matrices on n qubits, i.e., the subgroup that maps a group into itself under conjugation $UgU^\dagger$. If U is in a Clifford group, then so is $e^{i\theta}U$ for all angles θ. If the superscript (n) is not present, the particular Clifford and Pauli groups should be determined by context and "the Clifford group" or "the Pauli group" refer to the appropriate group.

The action of a Clifford group element on the Pauli group is completely determined by its action on the standard generators $\{X_i, Z_i, i \in [n]\}$ of the Pauli group. Furthermore, an element of the Clifford group modulo global phases $\mathcal{C}_2/\{e^{i\phi}\}$ is also completely determined by this action since the

n-qubit Paulis are an orthonormal basis for the $2^n \times 2^n$ complex matrices. Therefore, it is enough to specify $UX_iU^\dagger$ and $UZ_iU^\dagger$ for $i \in [n]$ to specify a Clifford group element U up to a phase.

For example, the Hadamard gate H conjugates (X, Y, Z) to $(Z, -Y, X)$, and the Phase gate K conjugates (X, Y, Z) to $(Y, -X, Z)$. $CNOT$ conjugates IX to IX, XI to XX, IZ to ZZ, and ZI to ZI.

By construction, the Pauli group is a normal subgroup of the Clifford group. When a Pauli P acts on another Pauli Q by conjugation, the action is merely $PQP = (-1)^{\omega(P,Q)}Q$ where

$$\omega(P, Q) = \begin{cases} 0 & \text{if } [P, Q] = 0 \\ 1 & \text{if } \{P, Q\} = 0 \end{cases} . \tag{2.18}$$

The group $C_2^{(n)}/\{e^{i\phi}G_n\}$ is the Clifford group, ignoring global phases and how the action of a group element might change the sign of a Pauli element.

The Clifford group is generated by $CNOT$, H, and K. This can be proven directly by induction. An alternate proof shows that $C_2^{(n)}/\{e^{i\phi}G_n\}$ is isomorphic to the symplectic group $Sp(2n, 2)$, and that any element of the group can be reduced to the identity matrix by a sequence of operations corresponding to $CNOT$, H, and K [Got07]. The **symplectic group** $Sp(2n, 2)$ is the group of $2n \times 2n$ binary matrices M satisfying $M^T \Gamma M = \Gamma$ where

$$\Gamma = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}, \tag{2.19}$$

where I_n is the $n \times n$ identity matrix. We do not present the proof here, but the idea is to first show that $C_n/\{e^{i\phi}G_n\} \sim Sp(2n, 2)$ since the symplectic group preserves the symplectic inner product. $CNOT$, H, and K have representations as matrices in the symplectic group. We can use those to reduce $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in Sp(2n, 2)$ to $\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$, showing that M is a product of these gates.

The Clifford group is the largest finite subgroup of the unitary group, as is captured by the following theorem

Theorem 2 ([NRS01]) *Let $n \geq 1$ and let G be a finite group such that $C_2^{(n)} \subseteq G \subseteq U(2^n)$. Then there exists a root of unity ξ such that $G = \langle C_2^{(n)}, \xi I_{2^n} \rangle$.*

2.3.2 Stabilizer states

In the stabilizer formalism, a **stabilizer** S is an abelian subgroup of $\mathcal{G}_n$ that does not contain $-I$. The reason for excluding some subgroups will become clear in a moment. $\mathcal{G}_n$ is a finite group, so S can be described concisely by a smallest generating set $\hat{S} = \{g_i \mid i \in [m]\}$. S is abelian, so its generators g_i must commute, and a general element of S can be written $\prod_{i\in[m]} g_i^{b_i}$ where each bit $b_i \in \{0,1\}, i \in [m]$ indicates the presence or absence of the corresponding (order 2) generator. Therefore, $|S| = 2^m$. Since S is an abelian subgroup of $\mathcal{G}_n$, S cannot be generated by more than n elements upon consideration of the generating set for $\mathcal{G}_n$.

Let $\mathcal{H}$ denote the n-qubit Hilbert space. Fix a stabilizer S and consider the subspace of all $+1$ eigenvectors of the first generator $g_1 \in \hat{S}$

$$\mathcal{H}_{g_1} := \{|\psi\rangle \in \mathcal{H} \mid g_1|\psi\rangle = |\psi\rangle\}. \tag{2.20}$$

$\mathcal{H}_{g_1}$ must be nonempty, otherwise g_1 must have a $\pm i$ phase factor (so have no $+1$ eigenvectors) and generates a subgroup containing $-I$. This is explicitly ruled out by the definition of S. Precisely half of g_1's eigenvectors have eigenvalue $+1$, so $\mathcal{H}_{g_1}$ is a 2^{n-1}-dimensional subspace of $\mathcal{H}$. This bisection procedure can continue $m-1$ more times, for each generator in $\hat{S}$, until we obtain

$$C(S) := \mathcal{H}_{g_1,g_2,\ldots,g_m} = \{|\psi\rangle \in \mathcal{H} \mid g|\psi\rangle = |\psi\rangle \;\forall g \in S\}. \tag{2.21}$$

This 2^{n-m}-dimensional subspace of $\mathcal{H}$ is the **stabilizer subspace** $C(S)$ associated with S. A generic 2^{n-m}-dimensional subspace can be described by 2^{n-m} basis vectors, but stabilizer subspaces of the same dimension can be described economically by $m \leq n$ elements of the n-qubit Pauli group.

It may help at this point to keep in mind a simple example. The state vector

$$|\text{cat}\rangle := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.22}$$

is the famous "cat" state – the first qubit represents the internal state of the nucleus whose decay triggers the cat's demise and the second qubit represents the cat's beingness. This state is a 1-dimensional stabilizer subspace, also called a stabilizer state. Indeed, $XX \in \mathcal{G}_2$ exchanges the first and second term in the sum and ZZ acts trivially on both terms. This identifies two generators of S so there are no more, and $S = \{XX, ZZ, -YY, II\}$ stabilizes the cat state whose dimension is

2^0, i.e. $C(S) = \{|\text{cat}\rangle\}$. Using the binary notation, the generating set can be written

$$\hat{S} \mapsto \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right]. \tag{2.23}$$

For this simple example, the binary notation for the state is less concise than writing the state directly. For larger number of qubits, however, the stabilizer notations are exponentially more concise than the state vector notation.

2.3.3 Stabilizer circuits

There is a set of circuits, stabilizer circuits, that are easy to simulate and appear frequently in the study of stabilizer states and subspaces. The gates in these circuits are intimately related to the Pauli group. A **stabilizer circuit** is a quantum circuit consisting of gates in the Clifford group, measurements of Pauli observables, and Clifford group gates that are conditioned on past measurement outcomes. Clifford group elements map stabilizer generators to stabilizer generators under conjugation, so gates in the Clifford group map between stabilizer subspaces. In fact, any two stabilizer subspaces of the same dimension can be mapped to each other by Clifford group gates.

The next theorem is very important, both as a statement of the fundamental limitations of stabilizer circuits, and as a statement of hope for designing systems built from stabilizer circuits.

Theorem 3 (Gottesman-Knill [Got98a]) *A stabilizer circuit whose input is a stabilizer state can be efficiently and faithfully simulated using a classical probabilistic computer. A classical probabilistic computer is a classical computer that also has the ability to generate uniformly random bits. The simulation is faithful if it samples from the correct probability distribution over measurement outcomes and produces the correct output stabilizer state.*

Proof The Gottesman-Knill theorem is proven if we give one way this simulation can be implemented. Consider a stabilizer circuit and let a stabilizer with 2^n elements be associated with the input state of the stabilizer circuit. The proof proceeds in two pieces. The first piece shows that Clifford gates can be simulated, and the second shows that measurements can be simulated.

Section 2.3.1 reviewed that any gate in the Clifford group can be decomposed into a product of $O(n^2)$ $CNOT$, H, and K gates. Any one of $CNOT$, H, or K can be simulated by applying an update rule to each of the n stabilizer generators. The update rules correspond to conjugation by the Clifford. Each of the three possible update rules requires changing at most 2 single qubit Paulis and the phase of each generator, so applying the update rules take constant time. Simulating the original Clifford gate therefore takes $O(n^2)$ time.

The measurement of an n-qubit Pauli operator M has elements $P_k := P((-1)^k M)$. The stabilizer state $|\psi\rangle$ is stabilized by $S = \langle g_1, g_2, \ldots, g_n\rangle$ so

$$\rho = |\psi\rangle\langle\psi| = \prod_{i=1}^{n} P(g_i). \tag{2.24}$$

The probability of outcome $k \in \{0,1\}$ is

$$p_k := \mathrm{Tr}(P_k\rho) = \frac{1}{2} + (-1)^k \frac{1}{2^{n+1}} \mathrm{Tr}\left[\prod_{i=1}^{n}(M + Mg_i)\right]. \tag{2.25}$$

The trace is zero if $\pm M \notin S$ and otherwise it is $2^n(-1)^\ell$ if $(-1)^\ell M \in S$ so

$$p_k = \begin{cases} \frac{1}{2} \text{ if } \pm M \notin S \\ \frac{1}{2} + (-1)^{k+\ell}\frac{1}{2} \text{ if } (-1)^\ell M \in S. \end{cases} \tag{2.26}$$

$\pm M \notin S$ iff M anticommutes with some element of S. The post-measurement state when outcome k is obtained is (provided that $p_k \neq 0$)

$$\rho_k := p_k^{-1} P_k \rho P_k = \frac{p_k^{-1}}{2^{n+2}}\left[\Sigma + (-1)^k \{M, \Sigma\} + M\Sigma M\right] \tag{2.27}$$

where $\Sigma := \sum_{g\in S} g$. If $(-1)^k M \in S$ for some $k \in \{0,1\}$ then $P_k\rho = \rho$ and $p_k = 1$, so $\rho_k = \rho$. Otherwise, if $\pm M \notin S$ then M anticommutes with some generators $g_1, g_2, \ldots, g_j$, $1 \leq j \leq n$, and $p_k = \frac{1}{2}$ for $k \in \{0,1\}$. Let S_0 denote the subgroup of S that commutes with M. We can form new generators $g_1g_2, g_1g_3, \ldots, g_1g_j$ that commute with M so that S_0 is generated by these new generators together with $g_{j+1}, g_{j+2}, \ldots, g_n$. Therefore, $|S_0| = |S|/2$. The expression in square brackets in (2.27) becomes

$$2(I + (-1)^k M)\sum_{g\in S_0} g. \tag{2.28}$$

Therefore, ρ_k is the pure state stabilized by $S' := S_0 \cup (-1)^k M S_0$.

The measurement of any Pauli observable M can also be simulated efficiently. To simulate the measurement of M, it suffices to either compute the measurement outcome or flip a fair coin, depending on the stabilizer state. Which action to take can be determined in polynomial time by checking whether or not M commutes with all the elements of S. When the outcome is predetermined, it can be computed by putting the stabilizer generators into a standard form through Gauss-Jordan elimination and selecting those generators whose product is $\pm M$. This can be done in polynomial time also. □

2.4 Open quantum systems

The quantum states and gates described in Section 2.2 have matrix elements that are assumed to be chosen accurately and precisely. The reality, however, is that noise processes and systematic flaws tend to damage fragile states and introduce uncertainty and inaccuracy in gates. A general description of these processes can be obtained by considering an **open quantum system**. In an open system, the **system** $\mathcal{H}_S$ is part of a larger Hilbert space $\mathcal{H}_{SB} = \mathcal{H}_S \otimes \mathcal{H}_B$, where $\mathcal{H}_B$ is a **bath** Hilbert space. The system $\mathcal{H}_S$ can be observed and manipulated, but the bath is assumed to contain a large number of degrees of freedom that are all inaccessible. Interactions between the system and bath describe some types of noise. This section reviews the basic theory of such open quantum systems.

2.4.1 Ensembles of quantum states

A **probabilistic mixture** of quantum states is described by an ensemble $\{p_i \in [0,1], |\psi_i\rangle \in \mathcal{H}_S\}$, where $\sum_i p_i = 1$, and this ensemble is represented as a linear operator on $\mathcal{H}_S$ called the **density matrix** $\rho := \sum_i p_i |\psi_i\rangle\langle\psi_i| \in \mathcal{L}(\mathcal{H}_S)$. A density matrix with $\mathrm{Tr}(\rho^2) = 1$ is called a **pure state** (the condition means it is rank one), whereas any other density matrix is called a **mixed state**. A pure state $|\psi\rangle$ has a density matrix $|\psi\rangle\langle\psi|$. A unitary gate U on the system acts like $U\rho U^\dagger$. A measurement on the system, described by $\{M_m\}$, obtains outcome m with probability $p(m) = \mathrm{Tr}(M_m^\dagger M_m \rho)$ and the post-measurement state is $M_m \rho M_m^\dagger / p(m)$.

Probabilistic mixtures arise naturally when the system is entangled with the bath, and the bath is "averaged out", "lost", or "measured and forgotten". Suppose the joint state of the system and bath is ρ_{SB}. The system's state can be described by a **reduced density matrix**

$$\rho_S := \mathrm{Tr}_B(\rho_{SB}) \tag{2.29}$$

where Tr_B is known as the partial trace over B. The **partial trace** is defined by

$$\mathrm{Tr}_B(|s_1\rangle\langle s_2| \otimes |b_1\rangle\langle b_2|) := |s_1\rangle\langle s_2| \,\mathrm{Tr}(|b_1\rangle\langle b_2|) \tag{2.30}$$

where $|s_1\rangle, |s_2\rangle \in \mathcal{H}_S$, $|b_1\rangle, |b_2\rangle \in \mathcal{H}_B$, and the trace on the right hand side is the usual trace operation $\mathrm{Tr}(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle$.

The partial trace is actually the unique operation that gives rise to consistent values of observables on subsystems [NC00]. An observable M_S on the system is necessarily equal to $M_S \otimes I_B$ on $\mathcal{H}_{SB}$. The expected value of M must be the same whether we observe the system or the system-bath, so $\mathrm{Tr}(M_S\rho_S) = \mathrm{Tr}((M_S \otimes I_B)\rho_{SB})$. This is certainly true if ρ_S is taken to be the reduced density matrix. Suppose that $f(\rho_{SB})$ is another matrix operator describing the reduced state on $\mathcal{H}_S$. Expanding this function in an orthonormal basis of operators $\{B_i\} \subseteq \mathbf{L}(\mathcal{H}_S)$,

$$f(\rho_{SB}) = \sum_i B_i \,\mathrm{Tr}(B_i f(\rho_{SB})) = \sum_i B_i \,\mathrm{Tr}((B_i \otimes I_B)\rho_{SB}). \tag{2.31}$$

Therefore, there is only one matrix operator $f(\rho_{SB})$ if the expected value of each B_i given ρ_S must be consistent with $\mathrm{Tr}((B_i \otimes I_B)\rho_{SB})$. The partial trace is the unique operation that correctly describes observables for subsystems of a composite system.

2.4.2 Phenomenology of single qubit noise

Interaction with an environment can lead to relaxation and dephasing. We review the concepts of relaxation, dephasing, and how they are phenomenologically modeled by so-called T_1 and T_2 times.

Relaxation is a damping process that occurs when excited states, say $|1\rangle$, transition to other available modes, such as vibrations of surrounding atoms or emissions of photons. The excited state to transitions to a lower energy state, say from $|1\rangle$ to $|0\rangle$, for example. The energy difference is transferred to different quantum systems that are considered part of the environment and may not be experimentally accessible.

Dephasing destroys coherence of quantum information stored in local degrees of freedom. Dephasing processes can transform superpositions into probabilistic mixtures of orthogonal states. These mixtures may no longer exhibit important quantum behavior such as interference and instead can behave like classical biased coins when they are measured. Dephasing usually happens more quickly than relaxation since it can be caused by elastic collisions that do not add or remove energy.

Relaxation and dephasing can be roughly characterized by rates T_1 and T_2, respectively. For a single qubit, the noise process acts on a mixed state as

$$\rho = \begin{bmatrix} a & b \\ b^* & 1-a \end{bmatrix} \mapsto \rho' = \begin{bmatrix} (a-a_0)e^{-t/T_1} + a_0 & be^{-t/(2T_2)} \\ b^* e^{-t/(2T_2)} & (a_0 - a)e^{-t/T_1} + 1 - a_0 \end{bmatrix}.$$

As $t \to \infty$, the system relaxes to a thermal mixture $\mathrm{diag}(a_0, 1-a_0)$.

The thermal equilibrium state for a system whose dynamics are described by a Hamiltonian H is

$$\rho = e^{-\beta(T)H}/\mathcal{Z} \tag{2.32}$$

where $\beta(T) = (k_B T)^{-1}$, T is the temperature, k_B is Boltzmann's constant, and $\mathcal{Z} = \mathrm{Tr}\, e^{-\beta(T)H}$.

These processes are often called **decoherence** and are attributed in some models to interactions between system and environment. The nature of the coupling, as described by a Hamiltonian H_{SB} on the system and bath, can be used to derive properties of the decoherence process [Zur81, Zur82].

One way to measure the quality of a quantum gate is the average or worst-case fidelity, over all inputs, between ideal and actual outputs. The **fidelity** of states ρ and σ is $F(\rho, \sigma) := \mathrm{Tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}}$. Fidelity is not a metric in the mathematical sense but the angle $A(\rho,\sigma) := \cos^{-1} F(\rho,\sigma)$ between two states is. The fidelity between pure and mixed states is $F(|\psi\rangle, \rho) = \sqrt{\langle\psi|\rho|\psi\rangle}$, i.e., the square root of the overlap between ρ and $|\psi\rangle$. It is possible to show that $F(\rho,\sigma) = \min_{\{E_m\}} \sum_m \sqrt{p_m, q_m}$ where $\{E_m\}$ is a set of operators $E_m = M_m^\dagger M_m$ defined in terms of measurement operators M_m, $p_m := \mathrm{Tr}(\rho E_m)$, and $q_m := \mathrm{Tr}(\sigma E_m)$ [NC00]. The fidelity is, in this sense, the largest possible distance between the probability distributions over measurement outcomes for ρ and σ.

2.4.3 Quantum operations

One advantage of the density matrix formalism is that dynamics in $\mathcal{H}_S$ can be described without describing dynamics in $\mathcal{H}_{SB}$. Let ρ be a density matrix on $\mathcal{H}_S$. A **quantum operation** is a map $\mathcal{E}(\rho)$ that satisfies (i) $\mathcal{E}(\rho)$ is a density matrix, (ii) $\mathcal{E}(\sum_i p_i\rho_i) = \sum_i p_i \mathcal{E}(\rho_i)$ for probabilities p_i that sum to one, and (iii) $(I_R \otimes \mathcal{E})(A)$ is positive semidefinite for any positive semidefinite A on the composite system $\mathcal{H}_R \otimes \mathcal{H}_S$.

Theorem 4 (Kraus representation) *[NC00] Let $\{E_k\}$ be a set of operators $E_k \in \mathbf{L}(\mathcal{H}_S)$ such* *that* $\sum_k E_k^\dagger E_k = I$. *Then*

$$\mathcal{E} : \rho \mapsto \sum_k E_k \rho E_k^\dagger \tag{2.33}$$

defines a quantum operation. Furthermore, if $\mathcal{E}$ *is a quantum opration then there exists a set of operators* $\{E_k\}$ *such that* $\mathcal{E}$ *can be written in the form of equation 2.33. The operators* $\{E_k\}$ *are known as* **Kraus operators**.

2.4.4 Models of noise processes

Suppose that a system in the state ρ interacts with a bath in state $|0\rangle_B$ by way of a unitary gate U on the composite system. Let $\{|i\rangle_B\}$ be an orthonormal basis for $\mathcal{H}_B$, then

$$\mathrm{Tr}_B\left[U(\rho \otimes |0\rangle\langle 0|_B)U^\dagger\right] = \sum_{i,j} \mathrm{Tr}_B(E_i \rho E_j^\dagger \otimes |i\rangle\langle j|_B) = \sum_i E_i \rho E_i^\dagger, \tag{2.34}$$

and $E_i = {}_B\langle j|U|0\rangle_B$. The resulting quantum operation can be interpreted as a process where the system interacts with the bath, the bath is measured in the basis $\{|i\rangle_B\}$, and the outcome is unknown. The resulting state is a weighted average over the possible states $E_i \rho E_i^\dagger / p(i)$ associated with outcome i and weighted by their respective probabilities $p(i)$.

Quantum operations that have a Kraus operator $E_0 = \sqrt{1-p}I$, $p \in [0,1]$, may be interpreted as **stochastic noise processes** by which an error of some kind occurs with probability p. Such an operation can be written

$$\mathcal{E}(\rho) = (1-p)\rho + p\mathcal{E}'(\rho) \tag{2.35}$$

since the non-identity Kraus operators satisfy $\sum_{k,k\neq 0} E_k^\dagger E_k = pI$. The models used in Chapter 8 are stochastic.

The **depolarizing noise model on** n **qubits** is defined by

$$\mathcal{E}(\rho) = (1-p)\rho + pI_{2^n}/2 = (1-p)\rho + p \sum_{E \in G_n \backslash \{I\}} E\rho E^\dagger. \tag{2.36}$$

The n-qubit system is replaced by the completely mixed state $I_{2^n}/2$ with probability p. This means that the qubits have become maximally entangled with the environment and a uniform mixture over all possible pure states. Depolarization is, in this sense, a rather catastrophic error.

Typically, $T_2 \ll T_1$ so it is physically realistic to have a model where phase errors occur with greater likelihood than amplitude errors. A **biased stochastic noise model on one qubit** is defined by

$$\mathcal{E}(\rho) = (1 - p_x - p_z - p_x p_z)\rho + p_x X \rho X + p_z Z \rho Z + p_x p_z Y \rho Y, \tag{2.37}$$

where $p_x \ll p_z$. When $p_x = 0$, the model is a **dephasing noise model**. Of course, the probability of each Pauli operator can be different,

$$\mathcal{E}_p(\rho) = (1-p)\rho + p \sum_{E \in G_n \setminus \{I\}} p(E) E \rho E^\dagger, \tag{2.38}$$

where $p : G_n \to [0,1]$ assigns a probability to each element of G_n such that $\sum_g p(g) = 1$.

One kind of relaxation, spontaneous emission, can sometimes be described as **single qubit amplitude damping noise**. Amplitude damping is not a stochastic noise process since no linear combination of its Kraus operators is proportional to the identity. Amplitude damping has Kraus operators $E_0 = |0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1|$ and $E_1 = \sqrt{\gamma}|0\rangle\langle 1|$. The operator E_1 corresponds to relaxation from $|1\rangle$ to $|0\rangle$ with strength γ, and $E_0^\dagger E_0 = I - E_1^\dagger E_1$ is chosen so that the operators are complete.

Timing inaccuracies in gates can lead to another noise process that is not stochastic. If the inaccuracy is slowly varying compared to the computation's duration, then a constant angle δ may be added to each single qubit rotation. This is an example of a coherent **systematic error**. For example, a single qubit X rotation by angle θ may actually be implemented by $U = e^{-i(\theta+\delta)X/2}$.

2.5 Conclusion

This chapter has introduced the quantum circuit model as a foundation for the rest of the dissertation. As a special case of the quantum circuit model, a stabilizer circuit model has also be reviewed. The stabilizer formalism is an essential formalism for quantum information and computation, and we make constant use of it throughout the dissertation. Finally, we reviewed the essential elements of the theory of open quantum systems, so that the concept of decoherence and the need for techniques to combat decoherence is properly motivated.

Chapter 3

Models of physical systems

3.1 Introduction

This chapter reviews, at a high level, a model of qubits, gates, and measurements in one physical system, and how these may be combined to model larger quantum circuits in that physical system. The discussion is not meant to introduce the physics of this system in great depth, nor is it meant to discuss how to realize quantum computing devices. The chapter is meant to provide perspective on physical systems from the standpoint of system architecture, so that Chapters 7 and 8 may be read with a high level understanding of architectural constraints of model systems.

The **DiVincenzo criteria** [DiV00] for a circuit model quantum computer are criteria that a quantum system satisfies to be considered a circuit model quantum computer. They provide a frame for our review of circuit model elements implemented by a physical system. The DiVincenzo criteria are

1. The physical system must be physically scalable to an arbitrary number of well-defined qubits.
2. Qubits must be initializable to a well-defined starting state.
3. Qubits must have long coherence times relative to gate times, and it must be possible to implement a universal set of gates.
4. The system must permit high quantum efficiency measurements on arbitrary single qubits.

In this chapter, we review a model for a physical system where the criteria are all met in principle. This model is an example of a **device model** above and with which components, in Chapters 4, 5, and 6, and systems, in Chapters 7 and 8, can be studied. We introduce this model to provide a concrete example that supports a vision: the problem of studying a large system architecture can be partitioned into managable strata with well-defined interfaces between layers. We make implicit use of this model architecture throughout the dissertation, whenever the concepts of locality, noise, and systematic inaccuracy are discussed. Together with the quantum circuit model and stabilizer formalism reviewed in Chapter 2, this is our example of an abstract device-level model.

3.2 Atomic ions in radio-frequency traps

Trapped atomic ions behave like elementary quantum systems that are well isolated from their environment, yet they can be precisely controlled. Controlled state transitions in trapped ions have been used in precision timekeeping applications for decades. Laser cooling techniques can bring trapped atoms nearly to rest, creating some of the coldest known matter. More recently, experiments have demonstrated fundamental quantum logic gates and some elements necessary to scale experiments to larger numbers of qubits.

3.2.1 Ion-qubits and radio-frequency traps

Trapped-ion quantum computation, as initially proposed by Cirac and Zoller [CZ95], uses a number of atomic ions trapped in a linear radio-frequency (RF) trap that can interact with laser fields to quantum compute. Each qubit is identified with two internal electronic or nuclear states of an ion. For example, the $^{40}\mathrm{Ca}^{+}$ ions used in experiments at Innsbruck identify $|0\rangle$ with the $4^2S_{1/2}$ ground state $|g\rangle$ and $|1\rangle$ with the $3^2D_{5/2}$ metastable excited state $|e\rangle$; see Figure 3-1 [Mon08]. Two or more ions can be contained in a single trap, where they couple to each other through Coulomb repulsion, forming a linear chain of ions called a Coulomb crystal. The vibrational modes of this chain provide a qubit-qubit interaction. Single qubit rotations and the qubit-qubit interaction yield a universal set of quantum gates for quantum computation.

The Cirac-Zoller proposal does not scale to large numbers of qubits. As the length of the ion chain increases, the vibrational modes become progressively harder to identify, decreasing gate fidelities. These modes also couple more strongly to ambient fields, increasing the heating rate and, hence, the dephasing rate.

Kielpinski, Wineland, and Monroe propose using a network of interconnected ion traps. Multiple traps allow for smaller linear ion chains, and thus greater control over logic operations. Furthermore,

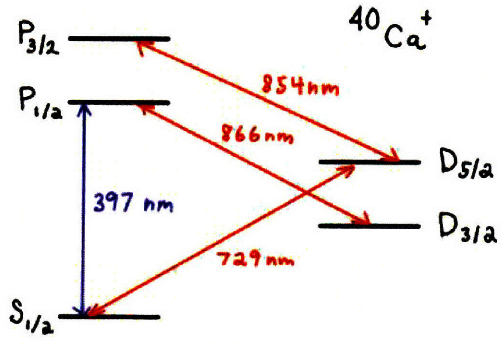


Figure 3-1: Energy levels of the $^{40}Ca^+$ system.

modulated electrode voltages move ions between traps within the network, potentially allowing coherent manipulation of a large number of ions. This kind of trap network is called a quantum charge-coupled device (QCCD).

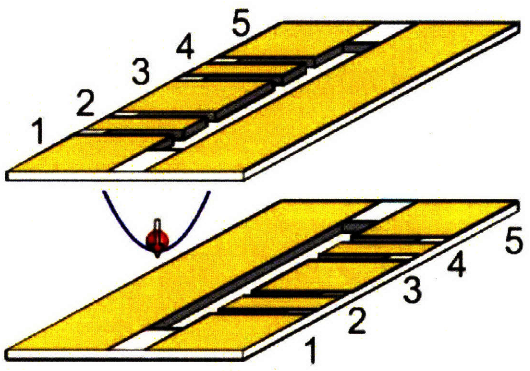


Figure 3-2: Schematic of the ion trap used in ion shuttling experiments at NIST Boulder, courtesy of David Wineland. Individual ions are trapped near electrodes 2 and 4. Ions can be moved by adjusting the static potentials on electrodes 1, 3, and 5.

Figure 3-2 is a schematic of a dual trap constructed at NIST Boulder. Individual ions are trapped in regions near electrodes 2 and 4. Slowly varying potentials on the other electrodes move ions between the two trapping regions.

There are ion-traps that constructed using semiconductor materials [SHO$^+$06] and traps that are planar [SCR$^+$06]. Quantum information processing with semiconductor traps may be more experimentally challenging but has the benefit of combining the scalability of semiconductor fabrication processes with the quantum control techniques of atomic physics. There are proposals for three-layer T-junctions to allow trapped ions to move throughout a large network of traps [HOS+06].

3.2.2 Gates and Measurement

Our discussion in this section is based on [Jam98]. The trapping electrodes establish a potential that confines the ions. The trap potential is typically tight in the radial direction, so N ions are trapped in a linear configuration, and can be indexed from left to right by integers. Since the ions are cold, the amplitude of each ion's motion is sufficiently small that the trapping potential is approximately harmonic. The quantized vibrational motion of the ions in the trap is modeled by $3N$ uncoupled harmonic oscillators, one for each Cartesian direction of each ion,

$$H_{\text{bus}} = \sum_{\alpha=1}^{3N} \hbar\nu_\alpha \left(\hat{a}_\alpha^\dagger \hat{a}_\alpha + \frac{1}{2} \right), \tag{3.1}$$

where $\hbar$ is Planck's constant, ν_α is the normal frequency of the mode labeled α, and $\hat{a}_\alpha$, $\hat{a}_\alpha^\dagger$ are the annihilation and creation operators, respectively. The $\{|n\rangle, |n+1\rangle\}$ manifold of one of the low-order modes is selected as a "bus" qubit to mediate interactions between ions, where $n = 0$ in the Cirac-Zoller scheme.

Each ion's internal qubit $\{|e\rangle, |g\rangle\}$ is modeled by the Hamiltonian

$$H_{\text{int}} = \frac{\hbar\omega_0}{2} Z + \frac{E_e + E_g}{2} I, \tag{3.2}$$

where Z is the Pauli Z-operator, I is the identity operator, and $\omega_0 = \frac{E_e - E_g}{\hbar}$ is the angular frequency of the qubit transition, given by the difference in energy between the ground and excited state. Shifting the energy origin, the total Hamiltonian for an ion qubit (indexed by integer j) and a single motional "bus" mode with frequency ν is

$$H_0 = \frac{\hbar\omega_0}{2} Z_j + \hbar\nu \hat{a}^\dagger \hat{a}. \tag{3.3}$$

Laser fields of specific duration, power, and phase are single qubit quantum gates when applied on resonance to a particular ion. In some experiments, beams can be steered to individually address ions at any location in the trap. The laser's electric field is modeled as a plane wave,

$$\mathbf{E}(t, \mathbf{q}) = E_0 \boldsymbol{\epsilon} \cos(\omega_L t - \boldsymbol{\kappa} \cdot \mathbf{q} + \phi), \tag{3.4}$$

that interacts with the ion through dipolar coupling, in the simplest case. In this expression, E_0 is the amplitude of wave, $\boldsymbol{\epsilon}$ is the polarization vector of the electric field, ω_L is the angular frequency of wave, t is time, $\boldsymbol{\kappa}$ is the wave vector which has magnitude $|\boldsymbol{\kappa}| = \frac{2\pi}{\lambda_L}$, λ_L is the free-space wavelength, $\mathbf{q}$ is the position vector, and ϕ is a phase shift. The interaction Hamiltonian for dipolar coupling is

$$V = -q_e \mathbf{r}_j \cdot \mathbf{E}(t, \mathbf{q}_j) \tag{3.5}$$

where q_e is the magnitude of an elementary charge, $\mathbf{r}_j$ is the (internal) position operator of the valence electron of the jth ion, and $\mathbf{q}_j$ is the (external) position operator of the jth ion.

The ion vibrational motion is typically cooled to the *Lamb-Dicke regime* as a prerequisite for applying quantum gates. The Lamb-Dicke regime corresponds to the physical situation where the spatial extent of the ion motion is much smaller than the laser wavelength. The ion spontaneously emits mostly on the carrier frequency because the recoil energy is much smaller than the energy of a vibrational quanta. Sideband cooling techniques applied in the Lamb-Dicke regime can cool the ion motion to the ground state. The formal conditions, collectively called the *Lamb-Dicke limit*, are

$$\eta(\langle n \rangle + \frac{1}{2})^{1/2} \ll 1, \tag{3.6}$$

$$\eta^2/2 \ll 1, \tag{3.7}$$

where $\eta = \kappa k z_0$ is the *Lamb-Dicke parameter*, $\langle n \rangle$ is the average number of phonons in the selected bus mode, κ is a parameter that depends the selected bus mode, and k is the magnitude of the laser wave vector. The distance $z_0 = \left(\frac{\hbar}{2m\omega_z}\right)^{1/2}$ is the extent of the ion's ground state wave function found from the expectation of the position operator, where m is the ion mass and ω_z is the trap frequency (which corresponds to ν in equation 3.3). One calculation for $^{40}\mathrm{Ca}^+$ ions gives $\eta \approx 0.06$, for which the Lamb-Dicke limit implies that $\langle n \rangle \ll 200$. For $\langle n \rangle > 200$, the ion is too warm for quantum computation.

The ion-laser interaction Hamiltonian V is not obviously useful for implementing quantum gates until it is written in a different form. This formal manipulation involves a sequence of approximations such as the rotating wave approximation, the weak-coupling approximation, and application of the Lamb-Dicke limit. When the laser is on-resonance, i.e., $\omega_L = \omega_0$, the final interaction Hamiltonian is

$$\mathcal{A} = \sum_{n=0}^{\infty} \cos(\frac{|A_n|t}{2})(|e\rangle\langle e| \otimes |n\rangle\langle n| + |g\rangle\langle g| \otimes |n\rangle\langle n|) \tag{3.8}$$

$$- i\sum_{n=0}^{\infty} \sin(\frac{|A_n|t}{2})(|e\rangle\langle g| \otimes |n\rangle\langle n| e^{-i\phi} + |g\rangle\langle e| \otimes |n\rangle\langle n| e^{i\phi}). \tag{3.9}$$

The laser intensity and dipole matrix elements in equation 3.5 determine the Rabi frequencies A_n, and ϕ is the phase of the laser. The effect of *carrier* excitation on the level populations is illustrated in Figure 3-3.

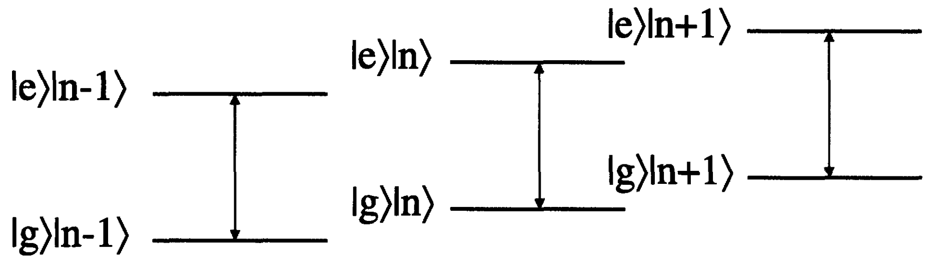


Figure 3-3: Energy levels of the atom-oscillator system together with carrier-driven transitions ($\omega_L = \omega_0$). Carrier-driven transitions rotate the qubit to desired superposition states.

Similarly, when the laser is tuned to $\omega_L = \omega_0 - \nu$, the interaction Hamiltonian becomes

$$\mathcal{B} = \sum_{n=0}^{\infty} \cos(\frac{|B_n|t}{2})(|e\rangle\langle e| \otimes |n\rangle\langle n| + |g\rangle\langle g| \otimes |n+1\rangle\langle n+1|) \tag{3.10}$$

$$- i\sum_{n=0}^{\infty} \sin(\frac{|B_n|t}{2})(|e\rangle\langle g| \otimes |n\rangle\langle n+1| e^{-i\phi} + |g\rangle\langle e| \otimes |n+1\rangle\langle n| e^{i\phi}). \tag{3.11}$$

This causes transitions on the first *red sideband*, as illustrated in Figure 3-4. Finally, tuning to $\omega_L = \omega_0 + \nu$ causes transitions on the first *blue sideband*, shown in Figure 3-5. Again, B_n is determined by laser intensity, the dipole matrix elements, and the Lamb-Dicke parameter.

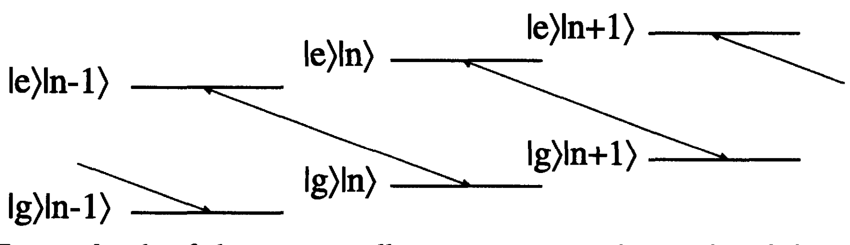


Figure 3-4: Energy levels of the atom-oscillator system together with red-driven transitions. Red-driven transitions change the populations of the qubit energy levels and the oscillator energy levels simultaneously. The oscillator loses one quanta of vibrational energy in a transition on the red sideband.

Single qubit rotations of the ion's internal state in the $\{|g\rangle, |e\rangle\}$ basis are represented by oper-

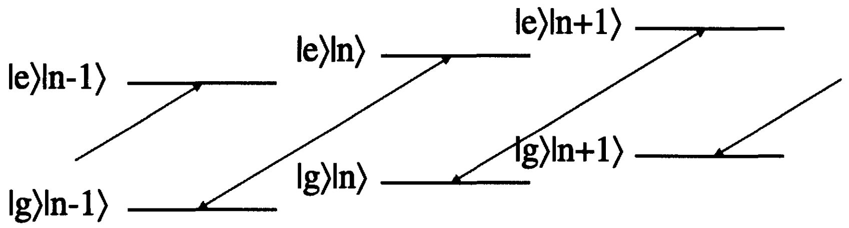


Figure 3-5: Energy levels of the atom-oscillator system together with blue-driven transitions. Blue-driven transitions change the populations of the qubit energy levels and the oscillator energy levels simultaneously. The oscillator gains one quanta of vibrational energy in a transition on the blue sideband.

ators

$$\mathcal{R}(\theta,\phi) = \begin{pmatrix} \cos(\frac{\theta}{2}) & e^{i\phi}\sin(\frac{\theta}{2}) \\ -e^{-i\phi}\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}, \tag{3.12}$$

where θ is the angle of rotation and ϕ is the relative phase shift of the ground and excited states. Equation 3.8 directly implements a single qubit rotation. Laser intensity, phase, and duration determine the angles θ and ϕ.

Universal quantum computation requires a gate such as the controlled-NOT (CNOT). There are several ways to perform CNOT gates with trapped-ions, and they can be implemented using fields tuned to the carrier, red, and blue sideband with varying intensity, duration, and relative phase [CZ95, SM99, LDM$^+$03]. An important engineering challenge appears to be how to integrate lasers into a system, and alternative proposals attempt to address this issue [LKOW07]. Gates have been implemented with $O(1-10^{-3})$ fidelity [BKRB08].

The electron shelving method accomplishes reliable single qubit projective measurement in the computational basis $\{|g\rangle, |e\rangle\}$. An intense laser field excites a transition that transfers population between $|g\rangle$ and a fast-decaying auxiliary level $|aux\rangle$. If the qubit collapses to the ground state, that population is transferred to the $|aux\rangle$ level. The auxiliary level spontaneously emits a photon, decaying back to $|g\rangle$. As long as the cycling transition is driven, the presence or absence of fluorescence at $\omega_c = (E_{aux} - E_g)/\hbar$ determines if the post-measurement state is $|g\rangle$ or $|e\rangle$, respectively. However, if the qubit collapses to $|e\rangle$, the qubit remains in that state during the readout process and no photons are emitted.

3.2.3 Moving Ions between Traps

Quantum computation may interact and entangle a majority of the qubits in the system. Ion-trap quantum computing differs from other physical systems because qubit swapping can be implemented using **ballistic transport** [RBKD$^+$02, RLB$^+$07], in which ions are electromagnetically pushed from trap to trap. This method is a critical feature of the QCCD proposal [KMW02] because ions no longer have to be in the same trap for the duration of the computation.

Ballistic transport is not only advantageous for scaling, but also for reducing noise. Ballistic transport may be more reliable than swap gates. Ions can be separated from one another during measurement to reduce scattering errors induced by fluorescing ions. Twice the number of qubits may be necessary in some architectures [Got00] than are necessary here.

Ballistic ion transport experiments at NIST Boulder have used the trap illustrated in Figure 3-2 [RBKD+02]. Modulated static voltages on trapping electrodes 1, 3, and 5 shuttled a single $^9Be^+$ ion between traps 2 and 4. The data corresponds to 10^6 transfers over the 1.2mm distance. The transfer duration was 50μs and, in one particular experiment, occurred between each pulse of a spin-echo experiment. The resulting interference fringe contrast for two transfers, $96.8 \pm 0.5\%$, was due to imperfections in state preparation, detection, and the spin-echo pulses, rather than from environmental influences. No ion loss was ever observed as a result of the transfer.

Ions shuttled by ballistic transport in the NIST experiment with average velocity 0.024mm/μs heated at a rate of about 8×10^{-6}quanta/μm. This heating presents a challenge because qubit-qubit gates such as the controlled-NOT degrade when acting on hot ions. Hence, ions must be recooled periodically by laser fields applied on the blue sideband transitions. Moving to cryogenic traps reduces ion heating rates by several orders of magnitude [LGA+08].

The cooling laser can change the state of the qubit, but ions can be **sympathetically cooled** [KKM+00, BDS+03]. Sympathetic cooling couples an ion of a different species to the target ion. Fields applied to the sympathetic cooling ion do not strongly interact with the target ion because the optical transitions are at different frequencies.

The NIST experiment also studied separating and joining linear chains of ions between two traps, A and B, 1.2mm apart. Two ions confined in trap A were separated into traps A and B, then brought back together into trap A. This required several steps: laser cooling in trap A, trap parameter adjustment, and changing electrode DC voltages. The entire process required 10ms. Discrete voltage steps during the ramping process, as well as other imprecisions in control, caused motional mode heating. The splitting process left an ion in each trap 95% of the time, and a mean on-axis motional mode population of 140 ± 70 quanta assuming a thermal distribution. More recent experiments have a success rate greater than 99% and heating of about 1 quanta in the center of mass mode and 0 in the next higher mode. The separation time is also reduced to around 1ms.

3.3 Conclusion

We have reviewed a model for a trapped-ion quantum charge-coupled-device architecture. This model is an example of a basic device level model that adds notions of geometry and specific noise parameters to the abstract circuit model reviewed in Chapter 2. Such a model can be constructed for any candidate physical system, and thorough models make strong connections with the physical theory of the corresponding system. We do not make explicit use of such a model in the remainder of the dissertation, given the variety of models that can be created and the knowledge and data required to build a model that accurately matches current experimental observations. However, the concept of a device-level model is important to recall whenever locality and noise are mentioned at the component-level, in Chapters 4, 5, and 6, and the system-level, in Chapters 7 and 8.

Part II

Components

Chapter 4

Quantum codes in the stabilizer and codeword stabilized frameworks

4.1 Introduction

Quantum codes are powerful tools against noise and inaccuracy. Codes can be designed to correct specific noise operators arising in an experiment. They can also be selected to correct arbitrary errors on a small number of qubits. Their efficiency and error correcting power can be tuned based on the application.

Quantum codes can be viewed as components of a quantum computer in the following sense. If a particular quantum code is chosen, for whatever reason, as a means to combat noise, the structure of that code is instantiated as a collection of circuits and algorithms that may operate continuously to locate and purge errors from the system. By some estimates, the vast majority of energy that a quantum computer consumes will be spent on this process alone. Choosing a "good" set of codes is crucial, where goodness will be measured in several different ways throughout this dissertation. Goodness can mean that a code encodes the largest amount of information for a given error correction capability. It can also mean that the circuitry implementing the error correction procedure is small and efficient. Goodness may mean that the code belongs to a family where the error correction capability "keeps up" as the size of the code grows.

The first section, Section 4.2, reviews classical codes and gives an example of one of the first quantum error-correcting codes to be discovered. The example illustrates the main ideas of quantum error-correction. Next, the general theory of quantum codes is quickly reviewed, culminating in a statement of the error-correction conditions.

Section 4.3 reviews stabilizer codes. These codes are important for much of the dissertation, so in addition to reviewing the theory, we record several important stabilizer code examples. These examples make their debut here and return again in Chapter 8, where they are used for fault-tolerant computing.

Finally, in Section 4.4, we present the first new result of this dissertation – a framework for quantum codes that are analogous to classical nonlinear codes. The framework includes all of the stabilizer codes but goes beyond them to include all known nonstabilizer quantum codes with optimal parameters. The framework restates the quantum error-correcting conditions as conditions for a classical code to detect an error model induced by a graph, creating a simple way to understand many previously opaque quantum codes with good parameters. Searching for optimal codes in this framework is equated to an iteration over inequivalent graphs on a fixed number of vertices, where the NP-hard problem of finding maximum cliques on an induced graph is solved on each iteration. Such a search can be carried out for small codes, offering new examples of quantum codes.

The chapter ends in Section 4.5 with several open questions that are revisited and refined in later chapters.

4.2 Quantum codes and error-correcting conditions

This section begins by quickly reviewing classical binary codes then presenting an example of a quantum code. Next, definitions for general quantum codes and necessary and sufficient conditions for a quantum code to correct a set of errors are given [BDSW96, KL00, NC00, KSV02].

4.2.1 Classical binary codes

An $(n, K, d)_2$ **binary code** C on n-bits is a set of K distinct vectors in $\{0,1\}^n$ such that the minimum distance of C is d. C **encodes** K symbols into n bits. The **minimum distance** of C is $\min_{\mathbf{c},\mathbf{c}'\in C}\text{wt }(\mathbf{c}+\mathbf{c}')$, where wt $\mathbf{c}$ is the **Hamming weight**, or weight, of a binary vector c and equals the number of nonzero coordinates. C can **detect** errors of weight less than d and **correct** errors of weight no more than $t := \lfloor (d-1)/2 \rfloor$.

The **dual code** of C, denoted $C^{\perp}$, is the code

$$C^{\perp} = \{\mathbf{x} \in \{0,1\}^n \mid \mathbf{x}\cdot\mathbf{c} = 0\ \forall \mathbf{c} \in C\} \tag{4.1}$$

where $\mathbf{x} \cdot c$ is the usual dot product and the operations are modulo 2. $C^{\perp}$ is a linear code.

An $[n,k,d]_2$ **binary linear code** C on n-bits is a k-dimensional subspace of $\{0,1\}^n$ with minimum distance d. For a linear code, the minimum distance equals $\min_{\mathbf{c}\in C,\mathbf{c}\neq 0}\mathrm{wt}(\mathbf{c})$. If C is linear then $(C^{\perp})^{\perp} = C$. A linear code is **self-orthogonal** if $C \subseteq C^{\perp}$ and **self-dual** if $C^{\perp} = C$.

A linear code is defined by a $k \times n$ **generator matrix** G whose rows are a basis of C. The codewords of C are $\mathbf{x}G$ where $\mathbf{x}$ is a k-bit row vector representing the message to be encoded. A vector $\mathbf{x}$ is in $C^{\perp}$ iff $\mathbf{x}G = 0$. Therefore, $C^{\perp}$ consists of the vectors in the null space of G and has dimension $n-k$. The generator matrix can be brought into **standard or systematic form**

$$G = \left(\begin{array}{c|c} I_{k\times k} & A_{k\times(n-k)} \end{array} \right) \tag{4.2}$$

by Gauss-Jordan elimination, since it has rank k.

The **check matrix** of C is an $(n-k)\times n$ matrix H that is the generator matrix for $C^{\perp}$. If the generator matrix G of C is in standard form, then a check matrix for C is given by

$$H = \left(\begin{array}{c|c} -A^T_{(n-k)\times k} & I_{(n-k)\times(n-k)} \end{array} \right). \tag{4.3}$$

The rows of H are parity checks for C since $H\mathbf{c}^T = 0$ for all $\mathbf{c} \in C$. For $\mathbf{e} \in \{0,1\}^n$, the vector $H\mathbf{e}^T$ is called the **syndrome** of $\mathbf{e}$.

For example, consider the repetition code $C = \{000, 111\}$. The parameters are $[3,1,3]$. The generator matrix is (111) and the check matrix is $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$. The bit value 0 is encoded as 000, and a single error gives one of 100, 010, or 001. The corresponding syndromes are 10, 11, and 01, respectively, and each locates an error. The decoding algorithm that maps each syndrome to an error pattern is called **syndrome decoding**.

4.2.2 Shor's 9 qubit quantum code

The classical repetition code encodes a single bit into a trio of bits with the same value, $0 \mapsto 000$ and $1 \mapsto 111$. Shor constructed the first error-correcting quantum code using two classical repetition codes [Sho95]. A brilliant insight is that, to correct an arbitrary error on a single qubit, it is sufficient to correct bit-flips or phase-flips on single qubits, as we will see. This can be done using one repetition code to correct bit-flip errors and another repetition code to correct phase flip errors.

For the moment, consider a quantum code to be a procedure describing how to encode infor-

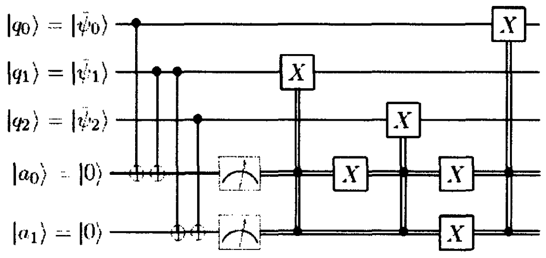

$|q_0\rangle = |\psi_0\rangle$
$|q_1\rangle = |\psi_1\rangle$
$|q_2\rangle = |\psi_2\rangle$
$|a_0\rangle = |0\rangle$
$|a_1\rangle = |0\rangle$
X
X
X
X
X
X

Figure 4-1: This quantum circuit couples a 3-qubit quantum codeword to two ancillary qubits. Upon measurement, these ancilla reveal the parity of adjacent bits and simultaneously project the quantum codeword onto a state with that parity. The measurement results locate bit-flip errors which can then be corrected by classically controlled correction operations. For example, the 3-qubit input state $|\psi\rangle = \sqrt{\epsilon}(a|010\rangle + b|101\rangle) + \sqrt{1-\epsilon}(a|000\rangle + b|111\rangle)$ leads to parity measurement 11 with probability ϵ and parity measurement 00 with probability $1-\epsilon$. These measurements indicate no error with probability $1-\epsilon$ and a bit-flip on the second qubit with probability ϵ.

mation and correct errors. We will review Shor's code by giving this procedure and demonstrating that single errors can be corrected. If the encoder for a classical repetition code is reversibly applied to a basis state $|b\rangle$ and two ancillary qubits $|00\rangle$, the basis state is mapped to $|bbb\rangle$. A single qubit $a|0\rangle + b|1\rangle$ is mapped to $a|000\rangle + b|111\rangle$. Let us call this code C_X. The encoder is a simple circuit using two CNOT gates. This code corrects a single X error on any qubit. Indeed, the circuit shown in Figure 4-1 measures the parity of pairs of adjacent bits and uses the outcomes, which indicate the location of any single bit-flip error, to correct the error. The parity measurements correspond to measuring the two-qubit operators Z_1Z_2 and Z_2Z_3.

However, a phase-flip error Z_i will invert the phase between $|000\rangle$ and $|111\rangle$. If we apply a Hadamard gate to each qubit, the code, call it C_Z, is now a repetition code in the conjugate basis, taking $a|+\rangle + b|-\rangle$ to $a|+\rangle|+\rangle|+\rangle + b|-\rangle|-\rangle|-\rangle$. The conjugate code can correct a single phase-flip error now, but it cannot correct a bit-flip error.

Shor's solution is to compose C_X and C_Z by encoding into C_Z then encoding each bit into C_X. This procedure encodes a qubit as

$$a|+\rangle + b|-\rangle \xrightarrow{C_Z} a\,|+\rangle\,|+\rangle\,|+\rangle + b\,|-\rangle\,|-\rangle\,|-\rangle \tag{4.4}$$

$$\propto a(|0\rangle + |1\rangle)^{\otimes 3} + b(|0\rangle - |1\rangle)^{\otimes 3} \tag{4.5}$$

$$\xrightarrow{C_X} a(|000\rangle + |111\rangle)^{\otimes 3} + b(|000\rangle - |111\rangle)^{\otimes 3} \tag{4.6}$$

Figure 4-2 shows the circuit that carries out this encoding. Qubits $1-3$, $4-6$, and $7-9$ are encoded into C_X, so the outcomes of measuring Z_iZ_{i+1} for $i = 1, 2, 4, 5, 7, 8$ indicate the location of any

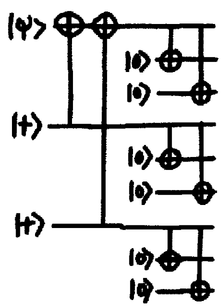


Figure 4-2: Encoding circuit for Shor's 9 qubit code.

single bit-flip error. A single phase-flip error flips the relative phase of $|000\rangle$ and $|111\rangle$ in the trio of qubits where the error occurs. The trio can be located by measuring $X_{1,2,\dots,6}$ and $X_{4,5,\dots,9}$. One way to derive these measurements is to observe that the conjugate measurements to Z_iZ_{i+1} are X_iX_{i+1}, and that $X \mapsto XXX$ by the encoding circuit for C_X, since it is a stabilizer circuit. Importantly, the measurement operators act trivially on an undamaged encoded state, so measuring them does not damage the relative phase between an encoded $|0\rangle$ and an encoded $|1\rangle$.

Suppose that the first qubit suffers a continuous rotation by the angle θ along some axis $\hat{n}$ on the Bloch sphere. Such a rotation can be written as a superposition of identity I, bit-flips X, phase-flips Z, and a bit-phase-flip Y, since we know the Paulis are an orthonormal basis for the 2×2 matrices. In this case, the measurement procedure for locating errors has four possible outcomes: either all of the outcomes are $+1$, only $X_{1,2,\dots,6}$ is -1, only Z_1Z_2 is -1, or only $X_{1,2,\dots,6}$ and Z_1Z_2 are -1. The post-measurement state must be an eigenstate of the measured operators, so the measurement actually collapses the continuous rotation error onto "no error" or one of the Pauli errors X_1, Z_1, or Y_1. In this sense, linear combinations of single qubit Pauli errors can be corrected because the correction procedure "digitizes" errors [NC00].

4.2.3 Error-correction conditions

A **quantum code** is a subspace $\mathcal{C}$ of a Hilbert space $\mathcal{H}$. $\mathcal{C}$ **detects** an error $E \in \mathbf{L}(\mathcal{H})$ if there exists some $c = c(E) \in \mathbb{C}$ such that

$$\forall|\psi_1\rangle, |\psi_2\rangle \in \mathcal{C},\ \langle\psi_2|E|\psi_1\rangle = c(E)\langle\psi_2|\psi_1\rangle. \tag{4.7}$$

Any state in $\mathcal{C}$ is called a **codeword**. This error-detecting condition is a precise statement of the fact that a detectable error does not deform the code. If an error rescales non-orthogonal codewords, it should do so in a way that depends only on the error operator. So, a detectable error cannot scale the codespace in a nonuniform way.

The **minimum distance** of $\mathcal{C}$ is the smallest number $d = d(\mathcal{C})$ for which the code does not detect errors from the space $\mathcal{E}(n,d)$. The error operators $\mathcal{E}(n,m)$ are defined as follows. For each subset of qubits $A \subseteq [n]$, let $\mathcal{E}[A]$ be the set of linear operators that only act on qubits in A. Let $\mathcal{E}(n,m) = \sum_{A:|A|\leq m} \mathcal{E}[A]$ denote the set of all linear operators that are sums of linear operators acting on m qubits or less. This set of error operators is analogous to the set of all classical errors of weight less than m.

Suppose that the Hilbert space $\mathcal{H}$ is a 2^n-dimensional space. A quantum code $\mathcal{C} \subseteq \mathcal{H}$ is parameterized by the block size n, the dimension $k = \dim \mathcal{C}$, and the minimum distance d. These data are usually written $((n,K,d))_2$. The subscript indicates that the distance and block size are defined with respect to qubits. Since this is the only case we consider in this dissertation, with little exception, we usually drop the subscript and write $((n,K,d))$.

The error-correction theorem can be stated succinctly in terms of error detection:

Theorem 5 (Error-correction conditions) *A quantum code $\mathcal{C} \subseteq \mathcal{H}$ corrects errors from $\mathcal{E} \subseteq \mathbf{L}(\mathcal{H},\mathcal{H}')$ iff it detects errors from the space $\mathcal{E}^\dagger\mathcal{E} = \{\sum_p A_p^\dagger B_p \mid A_p, B_p \in \mathcal{E}\}$. A code corrects t errors iff $d(\mathcal{C}) > 2t$. Usually, t is defined to be $\lfloor (d-1)/2 \rfloor$.*

Proof See one of [KSV02, NC00]. □

4.3 Stabilizer codes

Chapter 2 introduced the stabilizer formalism. The stabilizer formalism is a way to describe subspaces of states that are somehow easier to manipulate and comprehend than general states. These subspaces often make excellent quantum error-correcting codes. In fact, Shor's code is a stabilizer code.

4.3.1 Construction and properties

The stabilizer formalism for quantum error-correcting codes was invented by Gottesman [Got97] and, simultaneously, a formalism in which stabilizers are replaced by classical additive codes was invented by Calderbank, Rains, Shor, and Sloane [CRSS98].

Recall from Chapter 2 that a stabilizer subspace is specified by an abelian subgroup S of the n-qubit Pauli group. S has a minimal set of independent generators $|\hat{S}|$ such that $1 \leq |\hat{S}| \leq n$. A quantum code is a subspace of a Hilbert space, so we now refer to $C(S)$, the joint $+1$ eigenspace of the Pauli operators in S, as a **stabilizer code**. The dimension of $C(S)$ is $2^{n-|\hat{S}|}$, so the code encodes $k := n - |\hat{S}|$ qubits. A stabilizer code encoding zero qubits is just a stabilizer state.

For example, Shor's code is stabilized by a stabilizer S generated by

$$\hat{S} := \{X_{1,2,\dots,6}, X_{4,5,\dots,9}, Z_i Z_{i+1}, i = 1, 2, 4, 5, 7, 8\}, \tag{4.8}$$

which can be verified directly by applying these operators to the encoded qubit. Since $|\hat{S}| = 8$, the code encodes a single qubit.

The **projector onto a stabilizer code** $C(S)$ has an expression in terms of the stabilizer and/or its generators. The projector onto $C(S)$ is

$$P_{C(S)} = \frac{1}{2^m} \sum_{g \in S} g, \tag{4.9}$$

when S is generated by m elements. First, we can confirm that the rank of $P_{C(S)}$ is 2^{n-m} by observing that every element of S is traceless except the identity element which has trace 2^n, so $\mathrm{Tr}\, P_{C(S)} = 2^{n-m} = 2^k$. Consider the projector $P_g = \frac{1}{2}(I + g)$ associated with an observable $g \in \mathcal{G}_n$. It is easy to check that $P_g^2 = P_g$ and that $gP_g = P_g$, so P_g projects onto the $+1$ eigenspace of g. Therefore,

$$P_{C(S)} = \prod_{h \in \hat{S}} \frac{1}{2}(I + h). \tag{4.10}$$

Suppose we have a state in $C(S)$. If we measure a generating set of the stabilizer, each outcome will be zero, and the state will be unchanged. However, any Pauli error E that does not commute with the stabilizer can be detected by this set of measurements. Since there is a $g \in S$ such that $EgE^\dagger = -g$, one or more of the outcomes will be nonzero. If it is possible to infer E from the measurement outcomes, then E can be corrected. This motivates the consideration of the **normalizer in the Pauli group** of a stabilizer,

$$N(S) := \{g \in G_n \mid gsg^\dagger \in S\}. \tag{4.11}$$

For the Pauli group, the normalizer coincides with the centralizer

$$Z(S) := \{g \in G_n \mid [g, s] = 0 \text{ for all } s \in S\} \tag{4.12}$$

since elements of the Pauli group either commute or anticommute.

Theorem 6 (Error-correction conditions for stabilizer codes) *Let S be the stabilizer for a stabilizer code $C(S)$. Suppose $\{E_j\}$ is a set of operators in G_n such that $E_j^\dagger E_k \notin N(S) - S$ for all j and k. Then $\{E_j\}$ is a correctable set of errors for the code $C(S)$. The minimum distance of $C(S)$ is the minimum weight of an element of $N(S) - S$.*

Proof See [NC00]. □

The usual notation for stabilizer quantum code parameters is different than general quantum codes. The parameters of stabilizer codes are the block size n, the number of encoded qubits k, and the minimum distance d. They are usually grouped using double square brackets $[[n, k, d]]_2$. As before, the subscript is dropped if the dimension of each subsystem is already understood.

A stabilizer code is said to be **degenerate** if the minimum weight non-identity element of S has weight less than d. This means that some low weight errors act trivially on the code space and need not be corrected actively. Shor's code is degenerate, for example, since Z_1Z_2 is in the stabilizer and the code has distance 3.

The group $Z(S)/S$ is **isomorphic** to the k-qubit Pauli group G_k, meaning that cosets and group elements are in $1-1$ correspondence and have the same multiplication table. By a counting argument, there can only be $2k$ additional independent Pauli elements that commute with S. These elements can be chosen in pairs of **logical Pauli operators** $\{\bar{X}_i, \bar{Z}_i\}$ for $i \in [k]$ such that $\{\bar{X}_i, \bar{Z}_i\} = 0$ and all others commute. The logical Pauli operators act on the k encoded qubits in the same way that G_k acts on k-qubits. The isomorphism is explicitly given by an encoding circuit for the code; see Section 4.3.2. Each logical Pauli is derived by conjugating the single qubit Pauli X_i or Z_i by the encoding circuit. The choice is not unique of course – it amounts to a choice of basis for the code space since $\{|\bar{a}_1\bar{a}_2 \dots \bar{a}_k\rangle := \left(\prod_{j=1}^{k} \bar{X}_j^{a_j}\right) |\bar{0}\rangle\}$ spans the code space, where $|\bar{0}\rangle$ is stabilized by S and each of the $\bar{Z}_i$.

Stabilizer codes are analogs of classical linear codes – there is a sense in which the stabilizer is exactly a classical code [CRSS98]. A **finite field** is a set of elements together with an identity element, inverse elements, and addition and multiplication operations that satisfy commutativity, associativity, distributivity. $GF(4)$ is a finite field with elements $\{0, 1, \omega, \bar{\omega}\}$ that satisfy $\bar{\omega} := \omega^2 = 1 + \omega$ and $\omega^3 = 1$. The field elements correspond to binary vectors by writing them over the basis $\{1, \omega\}$, so $0 = [0|0]$, $1 = [0|1]$, $\omega = [1|0]$, and $\omega^2 = 1 + \omega = [1|1]$. Field elements can be written multiplicatively as a power of ω or additively as vectors with basis $\{1, \omega\}$ over the binary numbers. The **trace map** $\mathrm{Tr} : GF(4) \to \{0, 1\}$ takes x to $x + \bar{x}$.

A stabilizer S can be mapped, by the homomorphism binary(g) given in Chapter 2, to an additive subgroup binary(S) of $GF(4)^n$, where each binary tuple is $[x|y] = x + y\omega \in GF(4)$. Multiplication in S becomes addition in binary(S). The fact that S is abelian becomes the condition that the **trace inner product** $\mathrm{Tr}(v\bar{w})$ vanishes for all $v, w \in$ binary(S). This condition is equivalent to the symplectic inner product vanishing. So, a 2^k-dimensional stabilizer code is naturally associated to a **trace self-orthogonal additive code** of dimension $n - k$ over $GF(4)$. This additive code is spanned by the rows of an $(n - k) \times 2n$ check matrix

$$H_S = \begin{pmatrix} \mathrm{binary}(g_1) \\ \dots \\ \mathrm{binary}(g_{n-k}) \end{pmatrix} \tag{4.13}$$

where $S = \langle g_1, \dots, g_{n-k} \rangle$. Furthermore, if binary$(S)$ is a vector space, closed under multiplication by scalars in $GF(4)$, then it is called a $GF(4)$ linear code.

The theory of stabilizer codes includes the concept of a subsystem code that we now review to prepare for Chapters 6 and 8. Some subsystem codes have compact circuitry for error-correction [AC07] which motivates their study. In the usual **subspace codes** we have discussed, $k = \log \dim C(S)$. A **subsystem code** defines a partition of $C(S)$ into a **logical subsystem** $\mathcal{H}_L$ where information is encoded and a **gauge subsystem** $\mathcal{H}_G$ that can be "ignored". The partition is such that $C(S) = \mathcal{H}_L \otimes \mathcal{H}_G$ and two states $\rho_L \otimes \rho_G$ and $\rho_L \otimes \rho'_G$ in $C(S)$ are equated with one another even though $\rho_G \neq \rho'_G$ [Pou05].

Identify a subgroup $\mathcal{G}$ of $Z(S)$ called the **gauge group** that defines an equivalence relation $\rho \equiv \rho' \iff \exists g \in \mathcal{G} : \rho = g\rho' g^\dagger$. The remaining elements $\mathcal{L} = Z(S)/\mathcal{G}$ are the logical Pauli operators on the logical subsystem, and we must have $[\mathcal{L}, \mathcal{G}] = 0$. Therefore, the k encoded qubits of the original subspace code $C(S)$ have been partitioned into k' logical qubits on $\mathcal{H}_L$ and r **gauge qubits** on $\mathcal{H}_G$ such that $k = k' + r$. It can be seen that abelian gauge groups produce subspace codes and non-abelian gauge groups produce subsystem codes.

Calderbank-Shor-Steane (CSS) codes

The Calderbank-Shor-Steane (CSS) codes are stabilizer codes constructed from a pair of classical linear codes [CS96, Ste96]. They are a very important class of codes since they can be constructed from known classical codes and they have other properties we will see in Chapter 5. Many of the codes studied in Chapter 8 are CSS codes.

Suppose C_1 and $C_2^{\perp}$ are classical linear codes with parameters $[n, k_1, d_1]$ and $[n, k_2, d_2]$ such that $C_2^{\perp} \subseteq C_1$ and $d = \min\{\text{wt } \mathbf{x} \mid \mathbf{x} \in (C_1 - C_2^{\perp}) \cup (C_2 - C_1^{\perp})\} \geq \min(d_1, d_2)$. The **CSS code** $CSS(C_1, C_2)$ is the $[[n, k = k_1 - k_2, d]]$ quantum code spanned by

$$|\bar{\mathbf{x}}\rangle := |\mathbf{x} + C_2\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{\mathbf{c} \in C_2^{\perp}} |\mathbf{x} + \mathbf{c}\rangle \tag{4.14}$$

for all $\mathbf{x} \in C_1/C_2^{\perp}$. The number of cosets of $C_2^{\perp}$ in C_1 is $|C_1|/|C_2^{\perp}| = 2^{k_1}/2^{k_2} = 2^{k_1 - k_2}$. Each row $\mathbf{r}$ of the parity check matrix of C_2 gives an **X-type** stabilizer generator $X(\mathbf{r})$, and each row $\mathbf{s}$ of the parity check matrix of C_1 gives a **Z-type** stabilizer generator $Z(\mathbf{s})$, where $U(\mathbf{r}) = U_1^{r_1} \otimes \cdots \otimes U_n^{r_n}$. Therefore, The $(n-k) \times 2n$ check matrix of a CSS code as an additive quantum code is

$$H_S = \left(\begin{array}{c|c} H_{C_2} & 0 \\ \hline 0 & H_{C_1} \end{array}\right), \tag{4.15}$$

and the generators can be seen to separate into X-type generators H_{C_2} and Z-type generators H_{C_1}.

Indeed, $|\bar{\mathbf{a}}\rangle$ is a simultaneous eigenstate of these stabilizer generators: a row $\mathbf{r}$ of the parity check matrix of C_2 must be an element of $C_2^{\perp}$, so adding it to each codeword in the superposition $|\bar{\mathbf{a}}\rangle$ leaves the state unchanged, and every codeword in the superposition $|\bar{\mathbf{a}}\rangle$ is an element of C_1, so it must pass the parity checks of C_1.

A collection of representatives of the 2^k different cosets of $C_2^{\perp}$ in C_1 corresponds to logical $\bar{X}$ operations $X(\mathbf{a})$, $\mathbf{a} \in C_1$, because $X(\mathbf{a})\,|\bar{0}\rangle = |\bar{\mathbf{a}}\rangle$. Similarly, a collection of representatives of the 2^k different cosets of $C_1^{\perp}$ in C_2 corresponds to logical $\bar{Z}$ operations $Z(\mathbf{b})$, $\mathbf{b} \in C_2$, since $Z(\mathbf{b})\,|\bar{\mathbf{a}}\rangle = (-1)^{\mathbf{b}\cdot\mathbf{a}}\,|\bar{\mathbf{a}}\rangle$. We can choose these representatives such that the logical operators obey the commutation relations of the k-qubit Pauli group.

A special case of the CSS construction occurs when $C_1 = C_2$ is dual-containing. The X and Z stabilizer generators have identical supports. If in addition the weight of each stabilizer generator is a multiple of 4, $C_1^{\perp}$ is called **doubly-even**.

4.3.2 Encoding circuits

Gottesman has given an algorithm for encoding any stabilizer code given its stabilizer S [Got97]. We do not review this algorithm here because the encoding circuits described in Section 4.4.2 apply to stabilizer codes as well and produce circuits of the same size.

4.3.3 Examples of important stabilizer codes

The $[[5, 1, 3]]$ code

The $[[5, 1, 3]]$ code is the smallest quantum code encoding one qubit and correcting a general single qubit error [LMPZ96, BDSW96]. This can be seen from the quantum Singleton bound, $[[n, k, d \leq \frac{n-k}{2} + 1]]$ [KL00, Rai99a]. The stabilizer is generated by $XZZXI$ and its cyclic shifts. Only four of the shifts are independent of each other. The centralizer is generated by $e^{i\phi}$, S, $XXXXX$, and $ZZZZZ$. The code is a stabilizer code, but is not CSS, and it is a **perfect** quantum code, meaning that the number of syndrome vectors corresponds to the number of distinct correctable errors.

The $[[7, 1, 3]]$ Steane code

The $[[7, 1, 3]]$ code is the smallest CSS code correcting a general single qubit error [Ste96]. It is constructed from the Hamming code and is generated by

$$G_{C_2^\perp} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{4.16}$$

and $C_1 = C_2$. The normalizer is generated by S, $e^{i\phi}$, the string of all X's, and the string of all Z's. This Hamming code is a perfect classical code and it is also doubly-even. C_1 is dual-containing.

The surface codes

Surface codes are CSS codes that can be defined for many different kinds of surfaces and grids [BK98, FM01]. One family of surface codes, the $[[\ell^2 + (\ell - 1)^2, 1, \ell]]$ surface codes, are defined on the $\ell \times \ell$ grid shown in Figure 4-3. Surface codes have a stabilizer generated by A_s and B_p for all sites s and plaquettes p on the grid. There are $\ell(\ell - 1)$ site operators $A_s = \bigotimes_{j\in N(s)} X_j$ consisting of X's on the edges connecting to the site. Similarly, there are $\ell(\ell - 1)$ plaquette operators $B_p = \bigotimes_{j\in\partial(p)} Z_j$ consisting of Z's on the edges on the boundary $\partial(p)$ of the plaquette.

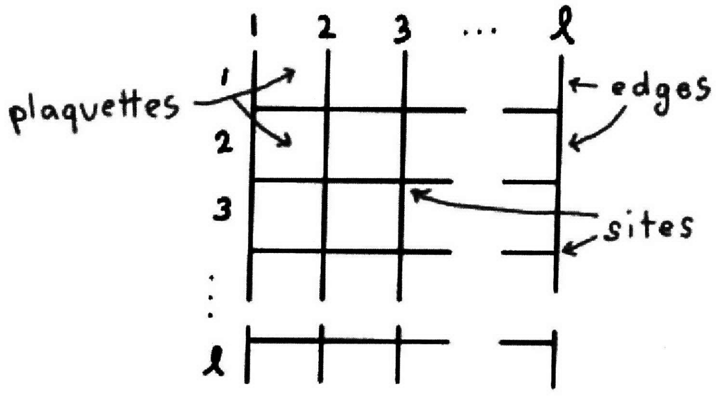


Figure 4-3: Grid for defining the surface code.

Any connected chain of X operators extending between the west and east boundaries of the grid commutes with the plaquette operators but is not a product of them. Such a chain represents $\bar{X}$. Similarly, any chain of Z errors extending between the north and south boundaries represents $\bar{Z}$. Since each of the stabilizer generators has constant weight four, if $\ell > 4$ then the surface code is highly degenerate.

One reason that the surface codes are interesting is that they have *constant weight local check operators.* Each generator of the stabilizer only involves qubits adjacent to a site or plaquette. This property allows measuring the generators using only a constant number of local interactions.

The Bacon-Shor codes

The Bacon-Shor codes are subsystem CSS codes due to Bacon [Bac06] that are closely related to Shor's code. There is a Bacon-Shor code for every integer $\ell \geq 2$ encoding one qubit into ℓ^2 qubits with distance ℓ. Imagine placing ℓ^2 qubits on the vertices of an $\ell \times \ell$ square grid. The Bacon-Shor code stabilizer is generated by

$$\langle X_{j,*}X_{j+1,*}, Z_{*,j}Z_{*,j+1}, j \in [n-1]\rangle, \tag{4.17}$$

where $P_{j,*}$ and $P_{*,j}$ denotes the Pauli P acting on all of the qubits in the jth row or column of the grid, respectively. The gauge group is

$$\mathcal{G} = \langle X_{j,i}X_{j+1,i}, Z_{i,j}Z_{i,j+1} \mid i \in [n], j \in [n-1]\rangle, \tag{4.18}$$

and $Z(S)/\mathcal{G} = \langle Z_{1,*}, X_{*,1}\rangle$.

One reason the Bacon-Shor codes are interesting is that the stabilizer generators can be written as products of gauge group generators

$$X_{j,*}X_{j+1,*} = \otimes_{k=1}^{n}(X_{j,k}X_{j+1,k}) \tag{4.19}$$

$$Z_{*,j}Z_{*,j+1} = \otimes_{k=1}^{n}(Z_{k,j}Z_{k,j+1}). \tag{4.20}$$

The gauge group generators act on pairs of adjacent qubits in the grid, so they can be measured by local operators to determine the syndrome [AC07].

Bacon-Shor codes can be obtained from surface codes by measuring all horizontal edges of the grid in the computational basis. This removes the horizontal edges and leaves an $[[n^2, 1, n]]$ Bacon-Shor code whose gauge qubits are initialized in an encoded computational basis state chosen uniformly at random.

The $[[23, 1, 7]]$ Golay code

The Golay code can be constructed from a dual-containing $[23, 12, 7]$ cyclic binary linear code with generator polynomial $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$. The generators are derived from the 11 classical codewords obtained from right cyclic shifts of 10100100111110000000000 by replacing 1 by X or Z and 0 by I. The normalizer is generated by the string of all X's and the string of all Z's. The Golay code is a perfect classical code and its group of symmetries is a famous example of a so-called sporadic simple group. The code is also doubly-even.

A $[[47, 1, 11]]$ quadratic residue code

This code is the next largest CSS code constructed from a dual-containing cyclic classical code that encodes one qubit with large minimum distance. It is constructed from a dual-containing $[47, 24, 11]$ with generator polynomial $g(x) = x^{23}+x^{19}+x^{18}+x^{14}+x^{13}+x^{12}+x^{10}+x^9+x^7+x^6+x^5+x^3+x^2+x+1$. The generators are derived from the 23 classical codewords obtained from right cyclic shifts of 10001100110110010010100110000000000000000000000 by replacing 1 with X or Z and 0 with I. The normalizer is generated by the string of all X's and the string of all Z's. This code is the smallest known CSS code encoding one qubit and constructed from a BCH code such that the minimum distance is larger than the Golay code.

The $[[15,1,3]]$ even-subcode Reed-Muller code

This code is a CSS(C_1, C_2) code constructed from a classical first order punctured Reed-Muller code $C_1 = RM^*(1,4)$ and its even subcode $C_2^\perp = \text{even}(RM^*(1,4)) \subseteq C_1$ [KLZ96]. The dual code of both $C_2^\perp$ and C_1 is a classical second order Reed-Muller code $RM^*(2,4)$. The generator matrix for $C_2^\perp$ is

$$G_{C_2^\perp} = \begin{bmatrix} 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{bmatrix} \tag{4.21}$$

and C_1 is generated by the same 4 vectors as $C_2^\perp$ as well as the all ones vector. The dual code $C_2 = C_1^\perp$ is generated by

$$G_{C_2} = \begin{bmatrix} 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&1&1 \\ 0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\ 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{bmatrix}. \tag{4.22}$$

The stabilizer has Z-type operators derived from $C_1^\perp$ and X-type operators derived from $C_2^\perp$.

Every generator has even weight. Therefore we can choose $\bar{X} = X^{\otimes 15}$ and $\bar{Z} = Z^{\otimes 15}$ as generators of $N(S)/\{e^{i\phi}S\}$. These operators anticommute as expected, and since there are 14 generators, we have a complete set of generators of the normalizer. The product of $\bar{Z}$, $I^{\otimes 7}Z^{\otimes 8}$, $I^{\otimes 3}Z^{\otimes 4}I^{\otimes 4}Z^{\otimes 4}$, and $I^{\otimes 11}Z^{\otimes 4}$ is in $N(S) - S$, has weight 3, and there is no operator in $N(S) - S$ of lower weight. Therefore, this code is a $[[15,1,3]]$ code. The code is interesting for reasons we return to in Chapter 6.

A $[[21,3,5]]$ concatenated polynomial code

The construction of this code will be given in Chapter 8. It is a CSS code with generator matrices of $C_2^\perp$ and $C_1^\perp$ as follows

$$G_{C_2^\perp} = \begin{bmatrix} 100000000101011011001 \\ 010000000001110110011 \\ 001000000110100100111 \\ 000100000110001111100 \\ 000010000111011100010 \\ 000001000010111101001 \\ 000000100101101010110 \\ 000000010001001101111 \\ 000000001110110011010 \end{bmatrix}, \; G_{C_1^\perp} \quad = \begin{bmatrix} 100000000101011011001 \\ 010000000001110110011 \\ 001000000110100100111 \\ 000100000110001111100 \\ 000010000111011100010 \\ 000001000010111101001 \\ 000000100101101010110 \\ 000000010001001101111 \\ 000000001110110011010 \end{bmatrix}. \tag{4.23}$$

A generating set of $Z(S)/S$ is $\bar{X}_1 := X_{10}X_{15}X_{16}X_{19}X_{20}$, $\bar{X}_2 := X_{11}X_{14}X_{15}X_{17}X_{19}X_{20}X_{21}$, $\bar{X}_3 := X_{12}X_{13}X_{14}X_{15}X_{18}X_{20}$, $\bar{Z}_1 := Z_{10}Z_{12}Z_{13}Z_{14}Z_{16}Z_{18}Z_{19}$, $\bar{Z}_2 := Z_{12}Z_{13}Z_{14}Z_{15}Z_{18}Z_{20}$, and $\bar{Z}_3 := Z_{10}Z_{11}Z_{14}Z_{16}Z_{17}Z_{21}$.

One reason concatenated polynomial codes are interesting examples is that they can be decoded as codes on **qudits**, i.e. d-dimensional quantum systems, allowing high weight adjacent errors to be corrected. This code can be decoded as a 7 qudit code on 8-dimensional systems, so some weight 3 errors can be corrected. The code also give us an example of a code encoding multiple logical qubits.

4.4 Beyond the stabilizer formalism

The $GF(4)$ framework [CRSS98] provides a natural mapping between stabilizer codes and classical linear (and additive) codes, but a complete, simple, and systematic understanding of quantum codes analogous to nonadditive classical codes has not yet been obtained. Any quantum code that is not a stabilizer code is referred to as a **non-additive quantum code**.

The first nonadditive quantum code was found in [RHSS97], and encodes a six-dimensional space into five qubits with a minimum distance of two. This outperforms the best additive five qubit distance two code, which can have an encoded dimension of at most four. The code was originally found as follows: It was known that the linear programming upper bound was exactly

6 for a blocklength 5 distance 2 code, and in fact it was possible to completely determine what the weight enumerator [Rai99c] of a code meeting this bound must be. The authors of [RHSS97] then performed a numerical search for such a code, and managed to find one. The structure of the resulting code was understood, but generating larger codes in a similar fashion seemed intractable (though [Rai99b] showed how to construct a $((5+2l, 2^{2l+1}3, 2))$ code from this code).

The code is defined by a projection matrix

$$P = \frac{1}{16}\left[3IIIII + IZYYZ_c + IXZZX_c - IYXXY_c + 2ZXYYX_c - 2ZZZZZ\right] \tag{4.24}$$

where A_c is the sum of A and all of its cyclic shifts. The error-correction conditions can be verified numerically but are difficult to verify analytically, despite the cyclic symmetry of the projector.

Once the projector was obtained, the authors of [RHSS97] observed that the $((5,6,2))$ is spanned by stabilizer states. Furthermore, they appreciated that other codes could be constructed in like manner:

> In principal one can construct other codes in a similar manner, e.g. as the spans of translates of self-dual stabilizer codes. Let C_0 be a self-dual additive code of length n with associated stabilizer quantum code Q_0 and let C be the union of K cosets of C_0. If C has minimum distance d, then there exists an $((n,K,d))$ quantum code.

However, an important challenge is to phrase the error-correction conditions so that codes of this form can be systematically found and understood. Some further steps in this direction were taken [RV99, GB97, AKP04], but none provide a simple way to understand the error-correction conditions or a systematic procedure for obtaining new codes. In particular, it is desirable that such codes can be constructed from well-known classical nonlinear codes. In 2006, Aggarwal and Calderbank proposed constructing non-additive codes from Boolean functions and projection matrices [AC08], but it seemed intractable to systematically apply their techniques to obtain new codes or transparently describe known nonadditive codes.

In 2007, Sixia Yu and coauthors [YCLO07] presented an example of a nonadditive code constructed from a graph. Graeme Smith, Bei Zeng, John Smolin, and I interpreted certain sign changes of this construction as classical code words, leading to the formulation of the **codeword stabilized codes (CWS)** framework for understanding and constructing nonadditive quantum codes from stabilizer states and classical codes [CSSZ08].

The innovation of the CWS framework is that all known nonadditive quantum codes with good parameters can be described as a pair of objects: a stabilizer state (i.e. a $GF(4)$-additive classical code) and a (nonlinear) classical code. Furthermore, the error-correction conditions for CWS codes can be stated in a classical language, in terms of a classical code detecting errors induced by a $GF(4)$-additive code. Given these conditions, the process of finding good nonadditive quantum codes can be achieved systematically.

4.4.1 Construction and properties

We begin by reviewing graph states, then proceed to the main theorems of the CWS construction. We conclude the section with examples of CWS codes and an algorithm for systematically obtaining new CWS codes.

Graph states

The $((5, 6, 2))$ is spanned by stabilizer states (which can each be viewed as a classical additive code), and there is a beautiful graphical way to study and think about these states. Any stabilizer state can be represented as a simple graph [Sch02, GKR02, dNDD04]. A **simple graph** is an undirected, unweighted graph that has no loops and at most one edge between a pair of vertices. The edges in this graph represent entanglement between qubits in a way that can be made precise.

The **graph state** associated with the simple graph $G = (V, E)$ is the state stabilized by

$$\langle X_v \otimes \left(\bigotimes_{w \in N(v)} Z_w \right), v \in V \rangle \tag{4.25}$$

where $N(v)$ is the set of neighboring vertices of v. This means that the ith row Λ_i of the adjacency matrix Λ becomes a stabilizer generator with an X on the ith qubit and a Z on every qubit where $\Lambda_{ij} = 1$. The graph is undirected so the generators pairwise commute, since if i is connected to j then j is connected to i. A graph state can be prepared simply by preparing each of the n qubits in the $|+\rangle$ state and applying a controlled-Z gate between the qubits at the end-points of each edge in the graph. These facts are illustrated by example in Figure 4-4.

A stabilizer state can be mapped to a graph state by a unitary gate of the form $U := U_1 \otimes U_2 \otimes \cdots \otimes U_n$ such that each U_i is in the single qubit Clifford group [Sch02, GKR02, dNDD04]. Such a gate is called a **local Clifford gate**. A gate of this form and a graph state can be derived mechanically from the starting stabilizer state. The algorithm to obtain the local gate and graph state from an input stabilizer state can proceed in two steps. First, a Pauli operator is chosen

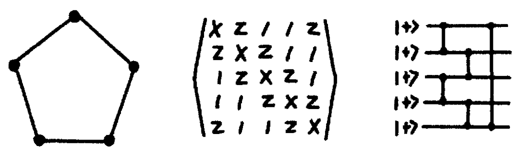


Figure 4-4: Example of a graph state, its stabilizer, and a circuit to prepare the state.

to adjust the signs of each stabilizer generator to be +1, then, second, the process of applying a sequence of local Clifford gates is reduced to that of applying operations to a binary matrix. The product of the Pauli operator and the sequence of operations on the binary matrix corresponds to the desired local Clifford gate. The final binary matrix contains the adjacency matrix of the graph.

The first step of the algorithm is as follows. Any stabilizer state can be mapped by a Pauli operator to a new stabilizer state whose phases are all +1. Let $S = \langle g_i \rangle$ stabilize a state $|S\rangle$. There is a unitary U that maps the state $|00\ldots0\rangle$ to $|S\rangle$. Since these states are both binary stabilizer states, U may be chosen to be in the Clifford group. Therefore, $\bar{g}_i := UX_iU^\dagger$ is a Pauli operator that anticommutes with g_i and commutes with g_j for all $j \neq i$. Hence, we may apply some product P of the $\bar{g}_i$ to $|S\rangle$ so that the resulting state $P|S\rangle$ is stabilized by $S' := \langle g_i' \rangle$ where $g_i' = \pm g_i$ and all g_i' have +1 phase.

The second and final step is now described. Apply the homomorphism binary(g) discussed in Chapter 2 to each generator of S to obtain a binary matrix M representing S. The single qubit Clifford gates are generated by H and K. H, acting on coordinate $i \in [n]$ of a row vector $[\mathbf{a}|\mathbf{b}]$ of M, swaps a_i and b_i. Likewise, K maps b_i to $a_i + b_i$. Any $n \times 2n$ matrix $M = [A|B]$ whose rows pairwise satisfy $(\mathbf{a}|\mathbf{b}) \odot (\mathbf{a}'|\mathbf{b}') = 0$ can be mapped to $M' = [I|\Lambda]$, where Λ is an adjacency matrix, by the following operations. Multiplying generators in S replaces row j by the sum of row j and another row j' in M. Applying $H[i]$ swaps column i and column $i+n$ in M. Finally, applying $S[i]$ adds column i to column $i+n$ in M. Gauss-Jordan elimination puts M into the form

$$\left(\begin{array}{cc|cc} I & A' & B' & 0 \\ 0 & A'' & B'' & I \end{array}\right). \tag{4.26}$$

H gates swap the right blocks of M to get

$$\left(\begin{array}{cc|cc} I & 0 & B' & A' \\ 0 & I & B'' & A'' \end{array}\right), \tag{4.27}$$

and, finally, K gates clear the diagonal of B' and A''. The final form of the right hand block is an adjacency matrix because the rows of $[A|B]$ originally pairwise satisfied $(\mathbf{a}|\mathbf{b}) \odot (\mathbf{a}'|\mathbf{b}') = 0$.

The representation of a stabilizer state as a graph is not unique – many graphs can correspond to a stabilizer state. Ideally this ambiguity, whose origin is a local Clifford freedom, would be completely resolved. Unfortunately, it must be resolved by a painstaking process of classifying inequivalent graph states at present. This has been accomplished for graphs of up to 13 vertices [Dan05].

CWS construction

An $((n, K))$ code will be described by two objects—S, a 2^n element abelian subgroup of the Pauli group not containing minus the identity, which we call the **word stabilizer**, together with a family of K n-qubit Pauli elements, $W = \{w_l\}_{l=1}^{K}$, which we call the **word operators**. There is a unique state $|S\rangle$ stabilized by S, *i.e.* $|S\rangle$ satisfies $s|S\rangle = |S\rangle$ for all $s \in S$. Our code will be spanned by basis vectors of the form

$$|w_l\rangle \equiv w_l |S\rangle . \tag{4.28}$$

Since the code vectors should all be different, at most one w_l can be in S. Typically we will choose $w_1 = I$ and later we will prove this can be done without loss of generality. Note that $|w_l\rangle$ is an eigenvector of all $s \in S$ with eigenvalue $\lambda_s = \pm 1$, but $|w_l\rangle$ is not stabilized by S unless $w_l \in S$. Each $|w_l\rangle$ is stabilized by a different stabilizer $w_l S w_l^\dagger$.

We would now like to understand the error correction capabilities of such a **codeword stabilized (CWS) code**. An $((n, K, d))$ code is an $((n, K))$ code capable of detecting Pauli errors of weight up to $d - 1$, but not d, and is said to have minimum distance d. From Section 4.2, we know that the error correction conditions for a general code with basis vectors $|w_l\rangle$ are that, in order to detect errors from a set $\mathcal{E}$, it is necessary and sufficient to have

$$\langle c_i| E |c_j\rangle = c_E \delta_{ij} \tag{4.29}$$

for all $E \in \mathcal{E}$. For a code of the form described above, this becomes

$$\langle S| w_i^\dagger E w_j |S\rangle = c_E \delta_{ij}. \tag{4.30}$$

To correct errors on a fixed number of qubits, it is sufficient to study errors of the form $Z^{\mathbf{v}} X^{\mathbf{u}}$

with bounded weight since these form a basis [BDSW96]. This leads to the *necessary and sufficient conditions* for detecting errors in $\mathcal{E}$ that for all $E \in \mathcal{E}$

$$\forall i \neq j \; w_i^\dagger E w_j \notin \pm S \tag{4.31}$$

and

$$\left(\forall i \; w_i^\dagger E w_i \notin \pm S\right) \text{ or} \tag{4.32}$$
$$\left(\forall i \; w_i^\dagger E w_i \in S\right) \text{ or} \tag{4.33}$$
$$\left(\forall i \; w_i^\dagger E w_i \in -S\right) \tag{4.34}$$

Eq. (4.31) is the condition that two codewords should not be confused after an error, while the final three conditions express that each error must either be detected (Eq. (4.32)), or the code must be "immune" to it–*i.e.* the code is *degenerate.*

Theorem 7 *An $((n, K))$ codeword stabilized code with word operators $W = \{w_l\}_{l=1}^K$ and codeword stabilizer S is locally Clifford-equivalent to a codeword stabilized code with word operators $w_l' = Z^{\mathbf{c}_l}$ and codeword stabilizer S' generated by*

$$S_l' = X_l Z^{\mathbf{r}_l}. \tag{4.35}$$

In other words, any CWS code is locally equivalent to a CWS code with a graph-state stabilizer and word operators consisting only of Zs. The set of $\mathbf{r}_l$s form the adjacency matrix of the graph. Moreover, the word operators can always be chosen to include the identity. We call this **standard form for a CWS code.**

Proof First note that S is local-Clifford equivalent to a graph state due to [Sch02, GKR02, dNDD04] so there is some local-Clifford unitary $C = \bigotimes_{l=1}^n C_l$ that maps S to S' of the form (4.35). In the new basis the word operators are $C w_l C^\dagger = \pm Z^{\mathbf{a}_l} X^{\mathbf{b}_l}$, and we have

$$C w_l C^\dagger \prod_i \left(S_i'\right)^{(\mathbf{b}_l)_i} = \pm Z^{\mathbf{c}_l}, \tag{4.36}$$

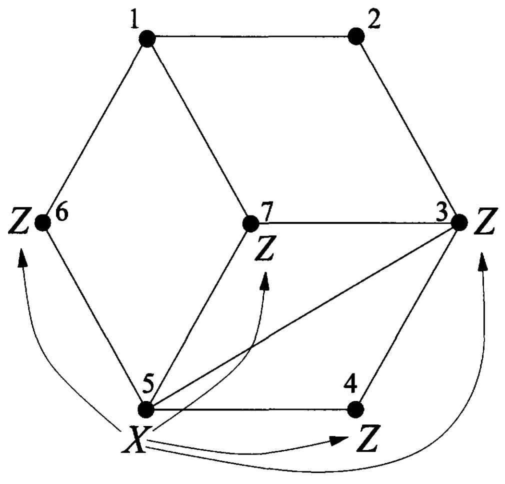


Figure 4-5: Example of the induced error on a graph state: The state has stabilizer generators $XZIIIZZ$, $ZXZIIII$, $IZXZIIZ$, $IIZXZII$, $IIZZXZZ$, $ZIIIZXI$, and $ZIZIZIX$. An X error applied to node 5 in the lower-left is translated by multiplying with the stabilizer element $IIZZXZZ$ and turns into Z errors on the nodes indicated.

so that, letting $w'_l = Z^{\mathbf{c}_l}$, we have

$$Z^{\mathbf{c}_l}\left|S'\right\rangle = \pm C w_l C^\dagger s' \left|S'\right\rangle = \pm C w_l C^\dagger \left|S'\right\rangle = \pm C w_l \left|S\right\rangle .$$

Since C consists of local Clifford elements, we see that the CWS code defined by S' and w' is locally Clifford equivalent to the original code.

Finally, to ensure the codeword operators include the identity we can choose $\tilde{W} = \{\tilde{w}_l{=}w'_l w'_1\}$ which always has $\tilde{w}_1 =$ Identity. This can be seen by commuting the w'_1 through the E in the error-correction conditions which can at worst pick up a sign depending only on E. The two conditions with $\pm S$ on the right are insensitive to this and the other two conditions at most change places. □

This structure theorem gives rise to the following lemma, which is at the heart of our construction:

Lemma 8 *A single qubit Pauli error Z, X or $Y = ZX$ acting on a codeword $w\left|S\right\rangle$ of a CWS code in standard form is equivalent up to a sign to another (possibly multi-qubit) error consisting only of Zs.*

Proof Let the error E_i act only on the ith qubit. If it is a Z error the result is immediate. Otherwise use the fact that $E_i w \left|S\right\rangle = \pm E_i S_i w \left|S\right\rangle$, and take S_i to be the generator having X on bit i. Then since $E_i = Z_i^{\{0,1\}} X_i$ the X in E_i cancels with the X from S_i and we are left with the

Zs from S_i as well as a Z_i if E_i was Z_iX_i. □

Lemma 8 allows us to construct CWS codes with a satisfying interpretation: X errors on any qubit are "pushed" outwards along the edges of the graph and transformed into Zs. This is illustrated in Figure 4-5. Similarly Y errors are pushed along the edges, but also leave a Z behind at their original locations. Since all errors become Zs, we can think of the error model as classical, albeit consisting of strange multi-bit errors. We define this translation to classical errors by the function $\text{classical}_S(E \in \mathcal{E}) \to \{0,1\}^n$:

$$\text{classical}_S(E = \pm Z^{\mathbf{v}} X^{\mathbf{u}}) = \mathbf{v} \oplus \bigoplus_{l=1}^{n} (\mathbf{u})_l \mathbf{r}_l \tag{4.37}$$

where $\mathbf{r}_l$ is the lth row of the stabilizer's adjacency matrix (recall from Eq. (4.35) $S_l = X_l Z^{\mathbf{r}_l}$ defines $\mathbf{r}_l$). The codeword operators $w_l = Z^{\mathbf{c}_l}$ will be chosen to so that the $\mathbf{c}_l$s are a classical code for this error model.

Theorem 9 *A CWS code in standard form with stabilizer S and codeword operators $\{Z^{\mathbf{c}}\}_{\mathbf{c}\in\mathcal{C}}$ detects errors from $\mathcal{E}$ if and only if $\mathcal{C}$ detects errors from $\text{classical}_S(\mathcal{E})$ and in addition we have for each E,*

$$\begin{aligned} \textit{classical}_S(E) &\neq 0 & (4.38)\\ \text{or } \forall i\ Z^{\mathbf{c}_i}E &= EZ^{\mathbf{c}_i}\ . & (4.39)\end{aligned}$$

The condition $\forall i, Z^{\mathbf{c}_i}E = EZ^{\mathbf{c}_i}$ can be written as $\forall i, \mathbf{c}_i \cdot \mathbf{u} = 0$. Thus, any CWS code is completely specified by a graph state stabilizer S and a classical code $\mathcal{C}$.

Proof When $i \neq j$, $w_i^\dagger E w_j \notin \pm S$ is satisfied exactly when $Z^{\mathbf{c}_i} E Z^{\mathbf{c}_j} \notin \pm S$, which is in turn equivalent to $Z^{\mathbf{c}_i} Z^{\text{classical}_S(E)} Z^{\mathbf{c}_j} \notin \pm S$. In standard form, the only element of S without any X is the identity, so that this is satisfied exactly when $\mathbf{c}_i \oplus \text{classical}_S(E) \neq \mathbf{c}_j$. This is explicitly the classical error-detection condition.

Similarly, when $i = j$, we must satisfy Eqs. (4.32), (4.33) and (4.34), whose three possibilities translate directly to

$$\forall \mathbf{c}\ Z^{\mathbf{c}} E Z^{\mathbf{c}} \notin \pm S \tag{4.40}$$

$$\text{or } \forall \mathbf{c}\ Z^{\mathbf{c}} E Z^{\mathbf{c}} \in S \tag{4.41}$$

$$\text{or } \forall \mathbf{c}\ Z^{\mathbf{c}} E Z^{\mathbf{c}} \in -S. \tag{4.42}$$

Since $Z^{\mathbf{c}} = I$ for the $\mathbf{c} = 0$ codeword, Eq. (4.40) is equivalent to $E \notin \pm S$ and therefore to (4.38). If (4.38) (and therefore (4.40)) is not satisfied, $E \in \pm S$. If any $Z^{\mathbf{c}}$ anticommutes with E we have also $E \in \mp S$. Since no $s \in S$ is also in $-S$ this readily implies the equivalence of (4.39) to (4.41) and (4.42). □

Remark A classical code expressed in quantum terms would traditionally comprise computational basis vectors that are eigenstates of Z, and therefore the operators mapping one codeword to another would be of the form $X^{\mathbf{c}}$ as these are the only errors that have any effect. It then might seem odd that standard form for CWS codes, the intuition of which is to make everything classical, would employ word operators and effective errors consisting only of Zs. This choice is arbitrary (one could exchange Z and X and nothing in the formalism would be affected) and is made since the usual form of a graph state stabilizer is to have one X and some number of Zs rather than the reverse. We hope this historical accident does not cause too much confusion going forward. □

Relation to Stabilizer codes

The CWS framework includes stabilizer codes, and allows them to be understood in a new way. We now show that any stabilizer code is a CWS code, and give a method for determining if a CWS code is also a stabilizer code.

Theorem 10 *An $[n, k]$ stabilizer code with stabilizer generators $S_1, \ldots, S_{n-k}$ and logical operations $\bar{X}_1 \ldots \bar{X}_k$ and $\bar{Z}_1 \ldots \bar{Z}_k$, is equivalent to the CWS code defined by*

$$S = \langle S_1 \ldots S_{n-k}, \bar{Z}_1 \ldots \bar{Z}_k \rangle \tag{4.43}$$

and word operators

$$w_{\mathbf{v}} = \bar{X}_1^{(\mathbf{v})_1} \otimes \ldots \otimes \bar{X}_k^{(\mathbf{v})_k} \tag{4.44}$$

where $\mathbf{v}$ is a k-bit string.

Proof To see that this CWS code describes the original code, note that the stabilizer state associated with S is $|\bar{0}\ldots\bar{0}\rangle$, while the codeword generated by $W_{\mathbf{v}}$ acting on $|\bar{0}\ldots\bar{0}\rangle$ is $|(\bar{\mathbf{v}})_1\ldots(\bar{\mathbf{v}})_k\rangle$. □

Theorem 11 *If the word operators of an $((n, K))$ CWS code are an abelian group W (not containing $-I$), then the code is an $[n, k = \log_2 K]$ stabilizer code.*

Proof The stabilizer S of the CWS code is a maximal abelian subgroup of the Paulis (not containing $-I$) therefore it is isomorphic to the group $S' = \langle X_1 \ldots X_n\rangle$ and the mapping from S to S' is a Clifford operation C (not necessarily local). This follows from the definition of the Clifford group as the automorphisms of the Pauli group. Because this automorphism group allows one to achieve any bijective mapping that preserves commutation relations (see Chapter 4 of [Got97]), the map can further be chosen to map W to $W' = \langle Z_1 \ldots Z_k\rangle$. Here we have made use of the facts that all $w \in W$ anticommute with at least one $s \in S$ (which implies $S \cap W = \{I\}$) and that S' is maximal, which allows us to choose for W' any order K group made only of Zs we like (since *all* products of X's are in S'). Note this nonlocal Clifford mapping is not the same as the conversion to Zs used in Theorem 7.

We can now choose T', $\bar{X}'$ and $\bar{Z}'$ as follows:

$$\bar{X}' = W' = \langle Z_1 \ldots Z_k\rangle \tag{4.45}$$

$$\bar{Z}' = \langle X_1 \ldots X_k\rangle \tag{4.46}$$

$$T' = \langle X_{k+1} \ldots X_n\rangle \tag{4.47}$$

The inverse Clifford operation $C^\dagger$ maps these to our stabilizer code with stabilizer T, and logical operations $\bar{X} = W$ and $\bar{Z}$.

It remains to show this is the same as the CWS code we started with. T is by construction a subgroup of S (T' is explicitly generated by a subset of the generators of S') and therefore stabilizes $|S\rangle$. T also stabilizes all $\bar{x}\,|S\rangle$, $\bar{x} \in \bar{X}$, since T and $\bar{X}$ commute. Using $\bar{X} = W$ we see these states are exactly the basis states of the CWS code. □

4.4.2 Encoding circuits

Thus far, we have focused on the existence and structure of CWS codes. We now address a question of fundamental importance: What is the complexity of encoding a CWS code? The answer we find is perhaps the strongest one could hope for: a CWS code will have an efficient encoding circuit as long as there is an efficient encoding circuit for the *classical* code $\mathcal{C}$.

We will use the fact [RBB03] that a graph state $|S\rangle$ whose graph has edges E is equal to $\prod_{(j,k)\in E} P_{(j,k)} H^{\otimes n} |0\rangle^{\otimes n}$, where $P_{(j,k)}$ is the two qubit controlled phase gate, acting on qubits j and k: $P|x\rangle|y\rangle = (-1)^{xy}|x\rangle|y\rangle$.

Theorem 12 *Let S and $\mathcal{C}$ define CWS code $\mathcal{Q}$, C be a unitary encoding circuit for the classical code $\mathcal{C}$, and Q be the unitary mapping $|0\rangle^{\otimes n}$ to $|S\rangle$. Then,*

$$U_{(\mathcal{Q},\mathcal{C})} = QC \tag{4.48}$$

is an encoder for $\mathcal{Q}$. See Figure 4-6. In particular, since Q has complexity no more than n^2, if C has complexity $f(n)$, the complexity of our encoder is $\max(n^2, f(n))$.

Proof The ith quantum codeword $|\mathbf{c}_i\rangle$ is given by $C|i\rangle$ where $\mathbf{c}_i$ is the ith codeword of $\mathcal{C}$. So,

$$\begin{aligned} QC|i\rangle &= \prod_{(j,k)\in E} P_{(j,k)} H^{\otimes n} X^{\mathbf{c}_i} |0\rangle^{\otimes n} && (4.49)\\ &= Z^{\mathbf{c}_i} \prod_{(j,k)\in E} P_{(j,k)} H^{\otimes n} |0\rangle^{\otimes n} && (4.50)\\ &= Z^{\mathbf{c}_i} |S\rangle && (4.51) \end{aligned}$$

□

The standard encoding circuit for a CWS code is shown in Figure 4-6. The circuit C is a classical reversible circuit that can be constructed from X, $\Lambda(X)$, and/or $\Lambda^2(X)$. G is a circuit corresponding to the graph state and can be constructed with controlled-Z gates.

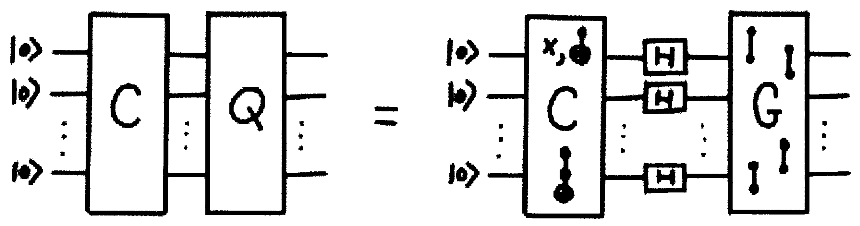


Figure 4-6: Standard encoding circuit for a CWS code.

4.4.3 Examples of important codeword stabilized codes

We now give some examples of our construction and include all known nonadditive codes with good parameters.

The $[[5,1,3]]$ code

The celebrated $[[5,1,3]]$ quantum code [BDSW96, KL00] can be written as a CWS code using Eqs. (4.43) and (4.44) but another way of writing it demonstrates the power of the CWS framework. Take generators corresponding to a ring graph (see Figure 4-4):

$$S_i = ZXZII \text{ and cyclic shifts.} \tag{4.52}$$

This induces effective errors as follows. Letting $|\text{R5}\rangle$ be the graph state corresponding to the unique simultaneous $+1$ eigenvector of these generators, we have

$$\begin{aligned} Z_i\,|\text{R5}\rangle &= Z_i\,|\text{R5}\rangle \\ X_i\,|\text{R5}\rangle &= Z_{i-1}Z_{i+1}\,|\text{R5}\rangle \\ Y_i\,|\text{R5}\rangle &= Z_{i-1}Z_iZ_{i+1}\,|\text{R5}\rangle\,, \end{aligned} \tag{4.53}$$

where all additions and subtractions are taken modulo 5. The corresponding 15 classical errors are:

$$\begin{array}{rlllll} Z: & 10000 & 01000 & 00100 & 00010 & 00001 \\ X: & 01001 & 10100 & 01010 & 00101 & 10010 \\ Y: & 11001 & 11100 & 01110 & 00111 & 10011 \end{array} \tag{4.54}$$

We then must choose $w_l = Z^{\mathbf{c}_l}$ where the $\mathbf{c}_l$s form a classical code capable of detecting pairs of these errors. Since no pair of these errors produces 11111 the codewords $\mathbf{c}_0 = 00000$ and $\mathbf{c}_1 = 11111$ will serve, and together with the stabilizer (4.52) completely define the code. Since the $((5,2,3))$ code is known to be unique we need not otherwise check that our construction is equivalent to the traditional presentation of this code. We note also that for $n \geq 7$ a ring code with codeword operators I and $\otimes_{l=1}^{n} Z_l$ gives a $((n,2,3))$ code.

The $((5,6,2))$ code

As a CWS code the $((5,6,2))$ code of [RHSS97] becomes simple. We again use the ring stabilizer (4.52) and will have to detect the induced errors (4.54), but since we are seeking a distance-2 code we need only consider single errors rather than pairs. The classical codewords $\mathbf{c}_l$, $l = 0 \ldots 5$, are

$$00000 \quad 11010 \quad 01101 \quad 10110 \quad 01011 \quad 10101 \tag{4.55}$$

and the code generated by $\left|c^{\mathrm{R5}}\right\rangle$ and $W_l = Z^{\mathbf{c}_l}$ is locally Clifford equivalent to the $((5,6,2))$ code. The $((5+2l, 2^{2l+1}3, 2))$ codes of [Rai99b] are also CWS codes whose graph state is the union of the ring graph and l Bell pair graphs, and whose classical codewords can be derived straightforwardly from the $((5,6,2))$ classical codewords.

The SSW codes

A family of distance two codes was found in [SSW07, FX08], which outperforms the family of [Rai99b] for odd blocklengths of eleven or larger. The codes were originally described in terms of their codewords as follows. If $n = 1 \mod 4$, a basis of our code consists of vectors of the form

$$|\mathbf{x}\rangle + |\bar{\mathbf{x}}\rangle, \tag{4.56}$$

where $\mathbf{x}$ ranges over all n-bit vectors of odd weight less than $(n-1)/2$ and $\bar{\mathbf{x}}$ is the complement of $\mathbf{x}$, while if $n = 3 \mod 4$, we let $\mathbf{x}$ range over even weight vectors of weight less than $(n-1)/2$, leading to an encoded dimension of $2^{n-2}\left(1 - \frac{\binom{n-1}{(n-1)/2}}{2^{n-1}}\right)$.

We now show that these are actually CWS codes. Indeed, the codeword stabilizer of this code will be generated by

$$\langle X_1 Z_2 \ldots Z_n, Z_1 X_2, Z_1 X_3, \ldots, Z_1 X_n \rangle, \tag{4.57}$$

with the corresponding stabilizer state being equivalent to a GHZ state, $(|0\rangle\,|+\rangle^{\otimes n-1} + |1\rangle\,|-\rangle^{\otimes n-1})/\sqrt{2}$. The codeword operators are simply $W_{\mathbf{x}} = X^{(\mathbf{x})_1} Z^{((\mathbf{x})_2, \ldots, (\mathbf{x})_n)}$ for each allowed $\mathbf{x}$, which can immediately be seen to generate, up to local unitaries, the same codewords as Eq. (4.56). Putting the stabilizer into standard form, we find that the graph state it describes corresponds to a star graph.

A $((9,12,3))$ **code**

Like the $((5,6,2))$ code, the codeword stabilizer is of the form

$$S_i = ZXZIIIIII \text{ and cyclic shifts.} \tag{4.58}$$

The associated classical code correcting the induced errors is:

$$\begin{array}{cccc}
000000000 & 100100100 & 010001100 & 110101000 \\
000110001 & 100010101 & 011001010 & 111101110 \\
001010011 & 101110111 & 011111111 & 111011011
\end{array} \tag{4.59}$$

This code was found by Yu [YCLO07] and motivated our work.

A $((10,24,3))$ **code**

Yu et al found an optimal 10 qubit code [YCO07] after we presented the CWS formalism. The codeword stabilizer is given by the graph in Figure 4-7. The associated classical code correcting the induced errors, with bits numbered from left to right, is:

$$\begin{array}{cccc}
0000000000 & 0010100010 & 0100000101 & 0110100111 \\
1010100100 & 0100001010 & 1010000001 & 0010001011 \\
1011010000 & 0011011010 & 0001001001 & 1010110011 \\
1110010100 & 0111001110 & 0101010001 & 0011110111 \\
1001110100 & 0101111010 & 1101111001 & 1001111111 \\
1100111100 & 1101101110 & 1101001101 & 1110111111
\end{array} \tag{4.60}$$

4.4.4 Search algorithm for codeword stabilized codes

One approach to finding new CWS codes might be to use existing classical codes directly. However, that approach gives sub-optimal code parameters, due to the fact that $\mathcal{C}$ must detect errors of the highest weight in the induced error patterns $\text{classical}_{\mathcal{S}}(\mathcal{E})$. So, the classical code $\mathcal{C}$ must have distance significantly greater than that of the corresponding quantum code, as shown in the following example:

Example Let $\mathcal{G}$ be an n qubit ring graph. If $\mathcal{E}$ is the set of single qubit Pauli X, Y, and

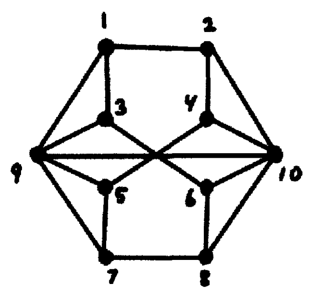


Figure 4-7: Graph state for deriving a $((10,24,3))$.

Z errors, then the induced classical errors $\text{classical}_{\mathcal{G}}(\mathcal{E})$ are single, triple, and double bit flips respectively. Choosing the classical code $\mathcal{C}$ to be a binary $((n,K,7))$ code results in a CWS code with parameters $((n,K,3))$. However, $\mathcal{C}$ also detects many additional errors which are unnecessary for this construction, such as all the one to six bit flip errors; $\text{classical}_{\mathcal{G}}(\mathcal{E})$ only includes a subset of those errors. □

This example motivates a search for specific classical codes which correct just the relevent errors for the CWS construction. However, classical coding theory provides no efficient, systematic constructions for codes that correct the potentially exotic error patterns involved in the CWS construction. On the other hand, finding a code with the best K for given graph and minimum distance is a problem which can be naturally encoded into an NP-complete problem such as MAXCLIQUE. This classic approach has been employed, for example, to show that the $(10,K,3)$ classical code with $K=72$ has optimal parameters [OBK99].

Algorithm

There are two sets of classical bitstrings that are important for describing the search algorithm. The first set, $\text{classical}_{\mathcal{S}}(\mathcal{E})$, was described earlier in this section. The second set derives from Equations 4.38 and 4.39, which are needed when the code is degenerate. For degenerate CWS codes, it will be useful to introduce the set of classical bitstrings

$$\text{degenerate}_{\mathcal{S}}(\mathcal{E}) = \{\mathbf{c} \in \{0,1\}^n \mid \text{classical}_{\mathcal{G}}(E) = 0 \text{ and} \tag{4.61}$$

$$\mathbf{c}\cdot\mathbf{u} \neq 0 \text{ for some } E = \pm Z^{\mathbf{v}} X^{\mathbf{u}} \in \mathcal{E}\}\,. \tag{4.62}$$

These bitstrings indicate codewords which are inadmissible, because they violate the condition given by equations (4.38) and (4.39) of Theorem 9. Specifically, fix a codeword $\mathbf{c}$, then for all

$E \in \mathcal{E}$ we must have $Z^{\mathbf{c}}E = EZ^{\mathbf{c}}$ if $\text{classical}_S(E) = 0$. Writing $E = \pm Z^{\mathbf{v}}X^{\mathbf{u}}$, $\mathbf{c}$ is not an admissible codeword if $\text{classical}_S(E) = 0$ and $\mathbf{c} \cdot \mathbf{u} \neq 0$. In other words, if a CWS code is degenerate, some low weight errors act trivially on the code space (i.e. $\text{classical}_S(E) = 0$), and these errors must act trivially on each basis state generated from the graph state (i.e. $[Z^{\mathbf{c}}, E] = 0$). $\text{degenerate}_S(\mathcal{E})$ describes basis states for which this is not the case.

CWS-MAXCLIQUE is a mapping onto MAXCLIQUE, of the problem of finding the CWS code with the largest possible dimension K, for a given minimum distance d and graph $\mathcal{G}$. The CWS-MAXCLIQUE algorithm gives steps to solve this problem, and is given in detail in the Algorithm 3 box. It proceeds in several simple steps. The first step, **Setup**$(\mathcal{E}, \Lambda)$ (Algorithm 1), finds the elements of $\text{classical}_{\mathcal{G}}(\mathcal{E})$ and $\text{degenerate}_{\mathcal{G}}(\mathcal{E})$. The second step, **MakeCWSCliqueGraph**(CL, D) (Algorithm 2), constructs a graph, denoted as the CWS "clique graph," whose vertices are classical codewords and whose edges indicate codewords that can be in the same classical code together. When searching for ordinary classical codes using an analogous procedure, the usual condition for joining two vertices by an edge is that the vertices are Hamming distance d apart. In our situation, vertices are joined by an edge if there is no error induced by the graph state that maps one codeword to the other. Finally, an external subroutine **findMaxClique**(V, E) is called; this routine is to employ known techniques to find the maximum clique in the CWS clique graph. The clique-finding subroutine is not specified here because there are many exact and heuristic techniques known in the community, for solving this classic NP-complete problem. Note that in the detailed description of the algorithms, two functions are used: String(i) : integer $i \rightarrow$ binary string of i with length n, and its inverse, Integer(i) : binary string with length n $i \rightarrow$ integer of i. Also, an error configuration is a list of ordered pairs (LOC, TYPE) where LOC is the coordinate of the affected qubit and TYPE is one of X, Y, or Z.

Bounds on code parameters

Linear programming upper bounds and known lower bounds are collected in Table 4-8. We hope this table will motivate further search for CWS codes with optimal parameters.

4.5 Conclusion

We have reviewed the stabilizer formalism and given examples of important stabilizer codes that also appear in Chapters 6 and 8. We have also introduced a new family of quantum codes called

Algorithm 1 **Setup**($\mathcal{E}, \Lambda$): Compute classical$_{\mathcal{G}}(\mathcal{E})$ and degenerate$_{\mathcal{G}}(\mathcal{E})$, where $\mathcal{E}$ is a set of Pauli errors and Λ is the adjacency matrix associated with graph $\mathcal{G}$.

```
Require: $\Lambda^T = \Lambda$, $\Lambda_{ij} = \{0,1\}$ and $\Lambda_{ii} = 0$
Ensure: $\text{CL}[i] = \delta(\text{String}(i) \in \text{classical}_{\mathcal{G}}(\mathcal{E}))$ and $\text{D}[i] = \delta(\text{String}(i) \in \text{degenerate}_{\mathcal{G}}(\mathcal{E}))$
 1: for $i \in \{0,1\}^n$ do
 2:     $\text{CL}[\text{Integer}(i)] \leftarrow 0$
 3:     $\text{D}[\text{Integer}(i)] \leftarrow 0$
 4: end for
 5: for error configuration $E \in \mathcal{E}$ do
 6:     ERR← String(0)
 7:     ERRX← String(0)
 8:     for (LOC, TYPE) in $E$ do
 9:         if TYPE is X or Y then
10:             ERR ← ERR ⊕ (row LOC of $\Lambda$)
11:             ERRX ← ERR ⊕ String($2^{\text{LOC}}$)
12:         end if
13:         if TYPE is Z or Y then
14:             ERR ← ERR ⊕ String($2^{\text{LOC}}$)
15:         end if
16:     end for
17:     CL[Integer(ERR)] ← 1
18:     if Integer(ERR) is 0 then
19:         for $i \in \{0,1\}^n$ do
20:             if ERRX $\cdot\, i \neq 0$ then
21:                 D[i] ← 1
22:             end if
23:         end for
24:     end if
25: end for
26: return (CL, D)
```

Algorithm 2 MakeCWSCliqueGraph(CL, D): Construct a graph whose vertices V are classical codewords and whose edges E connect codewords that can belong to the same classical code, according to the error model indicated by $\text{classical}_{\mathcal{G}}(\mathcal{E})$ and $\text{degenerate}_{\mathcal{G}}(\mathcal{E})$.

```
Require: CL and D are binary arrays of length $2^n$
Ensure: $0^n \in V$, $0^n \neq v \in V \Rightarrow \mathrm{D}[v] = 0$ and $\mathrm{CL}[v] = 0$, $(v, w) \in E \Rightarrow \mathrm{CL}[v \oplus w] = 0$
1: $V \leftarrow \{0^n\}$
2: $E \leftarrow \emptyset$
3: for $s \in \{0,1\}^n$ do
4:     if $\mathrm{D}[s] = 0$ and $\mathrm{CL}[s] = 0$ then
5:         $V \leftarrow V \cup \{s\}$
6:         for $v \in V \setminus \{s\}$ do
7:             if $\mathrm{CL}[v \oplus s] = 0$ then
8:                 $E \leftarrow E \cup \{(v, s)\}$
9:             end if
10:        end for
11:    end if
12: end for
13: return $(V, E)$
```

Algorithm 3 CWS-MAXCLIQUE($\mathcal{E}, \Lambda$): Find a quantum code $\mathcal{Q}$ detecting errors in $\mathcal{E}$, and providing the largest possible dimension K for the given input. The input Λ specifies the adjacency matrix of the graph $\mathcal{G}$. The output $\mathcal{C}$ is a classical code such that $\mathcal{Q} = (\mathcal{G}, \mathcal{C})$ is a CWS code detecting errors in $\mathcal{E}$.

```
Require: $\Lambda^T = \Lambda$, $\Lambda_{ij} = \{0,1\}$ and $\Lambda_{ii} = 0 \ \forall i$
Ensure: $K = |\mathcal{C}|$ is as large as possible for the given input, $0^n \in \mathcal{C}$, and $\mathcal{C}$ satisfies the conditions
    of Theorem 9
1: $(\mathrm{CL}, \mathrm{D}) \leftarrow$ Setup($\mathcal{E}, \Lambda$)
2: $(V, E) \leftarrow$ MakeCWSCliqueGraph(CL, D)
3: $\mathcal{C} \leftarrow$ findMaxClique($V, E$)
4: return $\mathcal{C}$
```

n\d	2	3	4	5
4	4	-	-	-
5	6	2	-	-
6	16	2	1	-
7	24-26	2-3	0-1	-
8	32	8-9	1	-
9	96-112	12-13	1	-
10	256	24	4-5	1
11	386-460	32-53	4-7	2
12	2^{10}	64-89	16-20	2

Figure 4-8: Upper and lower bounds on the optimal K for a nonadditive $((n, K, d))$ code. The lower bounds are drawn from [CRSS98, RHSS97, SSW07, YCLO07, YCO07, Rai99b]. The upper bounds are obtained from the linear program of [Rai99c], and for distance 2 its improvement in [Rai99b].

CWS codes that contains all of the stabilizer codes and presented methods for systematically understanding, constructing, and finding these codes.

In work going beyond the results presented in this chapter, CWS code structure has been further investigated and connections with the framework of Aggarwal and Calderbank have been clarified [CCS+08].

The current work on CWS codes has only discussed codes correcting general low weight errors. It is very interesting to consider codes that correct for specific types of error. Biased noise can be treated using asymmetric CSS codes, for example, and amplitude dampling has been treated with stabilizer codes [LNCY97]. Nonadditive codes with better parameters can be constructed for these models [LS07]; so how can the systematic CWS framework can be suitably expanded to include codes for different noise models?

It is also natural to wonder if there are nonadditive CWS code families with high degrees of symmetry that can be easily encoded, decoded, and error-corrected, and the relationship of these circuits to corresponding circuits for stabilizer codes. Indeed, one family of nonadditive CWS codes has been discovered, Goethals-Preparata CWS codes, using an idea of *codespace* stabilization [GR08]. It is also interesting to see if the usual methods of obtaining new codes from existing codes, such as lengthening, shortening, puncturing, and concatenating, lend themselves to a graphical interpretation in the CWS framework.

Chapter 5

Introduction to fault-tolerant computing

5.1 Introduction

Von Neumann introduced the concept of fault-tolerance in a paper published in 1956 [vN56]. His paper discusses a theory for how organisms such as humans maintain proper brain function despite unreliable individual neurons. The concepts he introduced were thought to be necessary for electronic computing systems, whose vacuum tubes frequenty failed and had to be replaced. The main result of his theory is that arbitrarily reliable classical computation is possible if the error rate is below a constant threshold, and the overhead involved in making the computation reliable scales efficiently with the desired final error rate.

Forty years later, Shor showed that fault-tolerant quantum error-correction circuits and gates can be constructed [Sho96]. The result quickly grew into a rich theory of fault-tolerance for quantum computation, and the seminal proofs of a constant accuracy threshold appeared in [ABO97, ABO99, Pre98, KLZ98, Got98b]. It is amazing that Von Neumann's ideas can be applied in a quantum setting with a small number of essential differences, although the analysis is generally more complicated, and it is challenging to prove results for strong noise models.

This chapter gives a conceptual introduction to fault-tolerant computing and highlights the differences between classical and quantum fault-tolerance, without getting involved in many of the technical details. The classical concepts that are introduced in Section 5.2 define what we refer to as **the standard approach** to fault-tolerant computing. This approach is a set of concepts that is sufficient for classical fault-tolerant computing. In Section 5.3, we see that a major difference in quantum fault-tolerance is that the standard approach is augmented by "quantum software" methods to obtain universal gates. Deviations from and developments beyond the standard approach in both the classical and quantum setting are discussed in Section 5.4.

The reason to linger on the idea of a standard approach is that Chapter 6 explores a limitation of the standard approach in the setting of quantum fault-tolerance, indicating that the standard approach, although sufficient for classical fault-tolerance, is insufficient for quantum fault-tolerance. Therefore, it turns out, it is likely that "quantum software" is necessary to achieve universality. Furthermore, Chapters 7 and 8 apply the ideas in this chapter to study accuracy thresholds for quantum fault-tolerance based on a hierarchy of codes.

5.2 Concepts of the standard approach to fault-tolerance

The two-input, one-output NAND gate, $\text{NAND}(a,b) = \overline{a \wedge b}$, is a universal gate for classical computation [MK97]. If our goal is to compute any binary function from n bits to m bits, we can do so with a classical circuit made out of wires, fanout gates that copy the value of a wire, and NAND gates. Suppose however that each NAND gate in our circuit fails, independent of the others, with probability p, and, when it fails, the output of the NAND gate is inverted. Such a failure is called a fault and it introduces an error. This model of noise is a stochastic bit-flip noise model. It is clear that we can expect to be able to compute the output of a circuit that contains about $T = p^{-1}$ NAND gates, since, to first order, this is when the probability of failure approaches unity. If we attempt many more than T gates, it is very likely that the circuit computes the incorrect function.

Amazingly, the original circuit can be modified slightly, introducing a reasonable amount of overhead, so that the effects of noise can be largely obviated. The following is a version of the classical threshold theorem [Imp04, TIC$^+$05]:

Theorem 13 *A classical circuit computing a function $f : \{0,1\}^n \to \{0,1\}^m$ subject to a stochastic bit-flip noise model with probability p can be simulated by another circuit that computes f with arbitrarily small failure probability $\delta > 0$ using at most a polylog(δ^{-1}) increase in circuit size and depth provided $p < p_{th}$ where p_{th} is a constant failure probability that does not depend on the original circuit size.*

This is an amazing theorem – suppose someone needs to know the value of a complicated but important function, but the function will take decades or centuries to compute using the best known techniques. If the noise strength in the system is less than some constant, which, as we will see, is classically quite large, an overhead roughly like the logarithm of the duration is adequate to ensure that the computation completes and obtains the correct answer with high probability. Even if the error rate is only slightly below threshold, we can apply the threshold theorem to realize a useful circuit whose mean time to failure is very large.

A key point is that the noise is not too correlated, although some correlation can be tolerated – for example, exponentially decaying correlations are bounded by the envelope for independent stochastic errors. Rare but highly correlated errors are almost impossible to correct depending on their strength, and cannot possibly be covered in full generality by a threshold theorem.

The ability to construct a fault-tolerant simulating circuit used in the threshold theorem relies on several crucial concepts. First, important information about the system's state should be stored in such a way that individual failures can be corrected. State information should never be stored in a single exposed bit at any time. Second, components acting on the information must be constructed so that the information is preserved even if a small number of devices within the component fail. This means that components are constructed to limit both the initial damage due to device failures and the subsequent damage caused as the initial errors propagate through the component. Third, the codes storing important state information can be concatenated and the components can be recursively simulated. Finally, these techniques lead to exponential reduction in error rates for at most polynomial increases in circuit size and depth, provided the error rate is below a fixed constant that does not depend on the size of the computation. Error rates per bit must remain constant as the simulation procedure increases the size of the circuit.

This section introduces each of these concepts by way of a running example. The example is called triple modular redundancy (TMR), and it is closely related to von Neumann's original ideas.

5.2.1 Information is encoded and never decoded

The first important concept is that information is encoded into an error-correcting code and is never decoded. A codeword can be restored if it is subject to a small number of errors. However, if the code is ever decoded down to a single bit, and an error occurs on that bit, the state is lost.

Chapter 4 introduced the classical $[n, 1, n]$ repetition code. Encoding into this code, a bundle of n wires carries the value of a single bit as the majority value over all n wires. To be concrete, let us take $n = 3$. The wires carry 000 if the encoded state is 0 and 111 if the encoded state is 1. There

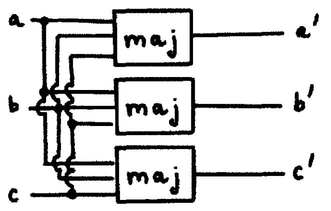


Figure 5-1: An error-correction component for classical information encoded into the 3-bit repetition code. Fan-outs are denoted by filled circles, and wire crossings are denoted by U-shaped bends in wires.

is an **error-correction (EC) component** to reset all of the bits in the bundle to the majority value of all the bits. The EC component can be constructed from three $\mathrm{MAJ}(a, b, c) = ab + bc + ca$ gates, each one taking the same set of inputs, copied using fan-out. This EC component is shown in Figure 5-1. For simplicity, we take the MAJ gate as part of our finite basis of gates for this example, and we allow any majority gate to fail with probability p.

The EC component may be used to restore the state at any time during the computation. It is clear that a bundle of wires that carries a single error is restored by the EC. Furthermore, if one of the MAJ gates fails, the output bundle could be restored in principle by an ideal EC, so the information is not yet lost. In practice, it is likely that a subsequent EC will restore the state. Since the EC must suffer two faults to produce the incorrect output, failure becomes a second-order event occuring with probability $O(p^2)$. If the information is decoded at any point during the computation, it is carried on a single wire for some period of time, and it can be flipped by a failure with probability $O(p)$, so this should be avoided.

5.2.2 Components control the introduction and spread of errors

The EC component introduced in Section 5.2.1 restores the state if there are no faults and introduces errors with probability p on single wires of the bundle when there are single faults. These properties can be considered general design rules or required conditions for fault-tolerant components. In addition, it is important to ensure that gates, such as the classical $\mathrm{CNOT}(a, b) = (a, a \oplus b)$ that can spread errors on their inputs to both outputs, are carefully incorporated into components to control the spread of errors. Such gates do not appear in the TMR example but are important to mention.

A component that acts on encoded inputs and has encoded outputs is called an **encoded logic gate** or a **logical gate**, to distinguish it from a physical gate acting on bare wires. Logical gates

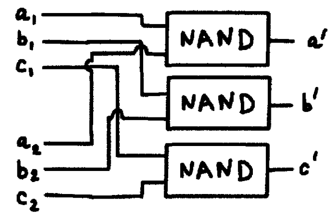


Figure 5-2: A logical NAND gate constructed from three physical NAND gates.

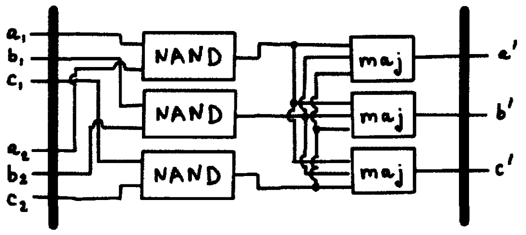


Figure 5-3: A NAND rectangle is constructed from a logical NAND gate followed by an error-correction component.

are written with bars over their labels, which should not be confused with negation in Boolean logic. We may casually drop the overbars at times and refer to $\overline{NAND}$ as a logical NAND, for example. When it is clear, the encoded logic gate will just be called a logic gate.

A NAND gate component performs a $\overline{NAND}$ gate on two encoded inputs and produces an encoded output. Figure 5-2 shows how a logical NAND gate can be constructed from three physical NAND gates. A logic gate is said to be **transversal** if it is implemented by physical gates on each bit independently. The $\overline{NAND}$ gate construction is tranversal. In this case, if a single NAND gate fails, only a single wire in the output bundle is corrupted.

The logical NAND gate becomes fault-tolerant when it is composed in series with an EC component. Any single fault in the circuit cannot produce more than one error in the output bundle, and one error in the input bundle is corrected if there are no faults in the circuit. This template of a logic gate followed by an error-correction is called a **rectangle**. Rectangles are a standard way to construct fault-tolerant logic gates. Figure 5-3 shows the NAND rectangle. Since the NAND is universal, we know how to construct a fault-tolerant version of a circuit: express it as NAND gates then replace each wire by a triple of wires and each NAND gate by a NAND rectangle.

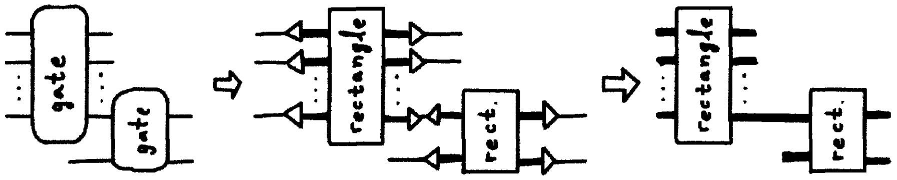


Figure 5-4: The recursive simulation procedure consists of two steps. The first step replaces each gate by a rectangle that is preceeded by encoders and followed by decoders. In this figure, the encoders and decoders are represented by triangles. Bundles of wires are drawn with thick lines. The second step replaces each decoder/encoder pair by a bundle of wires, since an ideal decoder/encoder pair acts like an identity gate.

5.2.3 Recursive circuit simulation can suppress logical fault rates

If each gate in a classical circuit is replaced by a rectangle, the resulting circuit is called a **level-1 simulation** of the original circuit. The original classical circuit can also be considered to simulate itself – this is a **level-0 simulation**. The following procedure can be used to make the replacement procedure more precise. First, take the associated rectangle and place encoders at each input and decoders at each output. Now the rectangle's inputs and outputs correspond directly to those of the gate. The level-1 simulation is obtained by replacing each gate with this circuit, then making a pass over the entire circuit, replacing decoder/encoder pairs by identity. This is illustrated in Figure 5-4. In our example, any classical circuit can be expressed as a circuit containing only NAND gates, and each NAND gate is replaced by the rectangle in Figure 5-2 to obtain a level-1 simulation of that circuit.

In our example, the level-1 simulation computes the incorrect function with probability $O(p^2)$ rather than $O(p)$. The simulation can continue recursively. A level-2 simulation is obtained from a level-1 simulation by applying the replacement procedure again. In general, a **level-u simulation** is obtained from a level-v simulation, $v < u$, by applying the simulation procedure $u - v$ times. Each time, the probability p that each gate fails is mapped to the $O(p^2)$ failure probability of the rectangle, so the probability that the level-ℓ simulation is incorrect is $O(p^{2^\ell})$. If the coefficients involved are not too large, the failure probability decreases very quickly at sufficiently high levels of recursive simulation.

5.2.4 A constant threshold failure probability exists

Consider the rectangle in Figure 5-3. The rectangle is designed so that a) if the input has no errors then a single fault causes the output to have no more than one error and b) if the input has one error and there are no faults then the output has no errors. The rectangle is considered **incorrect** if the output of the rectangle decodes (by an ideal device) to a different value than what we would get if we first decoded (by an ideal device) the inputs and fed them into an ideal NAND gate. In this case, the rectangle is incorrect if the output has two or more errors.

We would like to derive an upper bound on the probability that the rectangle in Figure 5-3 is incorrect. There are six gates in the rectangle that can fail with probability p. Some errors may originate in the logical NAND gate preceeding the rectangle, or earlier, so we allow single faults to occur on each input with probability p. This captures the chance that the majority voters of the preceeding NANDs fail. Failures at earlier points in the simulated circuit are considered corrected by the majority voters, so only the voters need to be considered. If we assume pessimistically that $w < w_0$ faults lead to a correct rectangle and $w \geq w_0$ faults lead to an incorrect rectangle, the failure probability is

$$\mathbb{P}(\text{incorrect}) = \sum_{w=w_0}^{C} \binom{C}{w} p^w (1-p)^{C-w} \leq \binom{C}{w_0} p^{w_0}, \tag{5.1}$$

where C is the number of possible fault locations in the rectangle plus the number of fault locations in a subcircuit of the preceeding rectangle. Generally, it suffices to take an EC as that subcircuit, as we do in this example. In this case $C = 12$, since there are 6 gates and 6 input wires, and $w_0 = 2$ since there are pairs of errors we cannot correct.

The bound in equation 5.1 is proven by

$$\begin{aligned}
\binom{C}{w_0} p^{w_0} &= \binom{C}{w_0} p^{w_0} \sum_{w=w_0}^{C} \binom{C-w_0}{w-w_0} p^{w-w_0} (1-p)^{C-w} && (5.2)\\
&= \sum_{w=w_0}^{C} \binom{C}{w}\binom{w}{w_0} p^w (1-p)^{C-w} && (5.3)\\
&\geq \sum_{w=w_0}^{C} \binom{C}{w} p^w (1-p)^{C-w}. && (5.4)
\end{aligned}$$

The **accuracy threshold for stochastic noise** is the critical probability below which any arbitrarily small logical failure probability can be achieved. Level-1 simulation certainly reduces

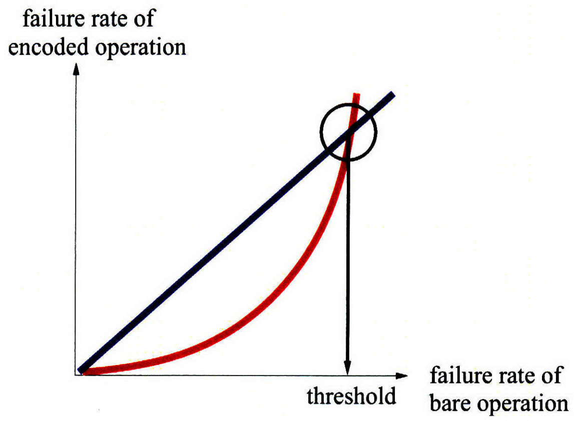


Figure 5-5: Rough schematic illustrating the concept of the accuracy threshold for stochastic noise. The figure is not strictly correct, but nevertheless presents a meaningful intuition – see [SCCA06].

$\mathbb{P}$(incorrect) if $\binom{C}{w_0}p^{w_0} < p$, and this is true if $p < \binom{C}{w_0}^{-1/(w_0-1)}$. Let us define a lower bound on the threshold

$$p_{th} := \binom{C}{w_0}^{-1/(w_0-1)}. \tag{5.5}$$

If $p < p_{th}$, recursive simulation continues to reduce the probability that the gate is incorrect; see Figure 5-5. At level-ℓ, we have

$$\mathbb{P}(\text{incorrect at level-}\ell) \leq p_{th}\left(\frac{p}{p_{th}}\right)^{w_0^\ell}, \tag{5.6}$$

so the probability that a rectangle is incorrect is bounded above by a quantity that decreases like a double exponential in the level of the recursive simulation. If there are T NAND gates in the circuit we plan to simulate, the final probability that the circuit computes the incorrect function is bounded above by $T\mathbb{P}$(incorrect at level-ℓ).

If we want $\mathbb{P}(\text{incorrect}) \leq \epsilon$, then Equation 5.6 implies

$$w_0^\ell \leq \frac{\ln(\epsilon^{-1})}{\ln(p_{th}/p)}. \tag{5.7}$$

Suppose no more than $N = \alpha^\ell$ gates are needed for a level-ℓ simulation of a NAND gate, for some $\alpha > 1$. Then

$$N \leq \exp\left(-\frac{\ln(w_0)}{\ln(\alpha)}\right)\frac{\ln(\epsilon^{-1})}{\ln(p_{th}/p)}, \tag{5.8}$$

so $N = O(\log(\epsilon^{-1}))$.

The threshold lower bound depends on a quantity C related to the size of the rectangle and also on the fact that the repetition code can correct some number of errors $t = w_0 - 1$. The lower bound does not depend on the number of gates in the circuit we are simulating. We can think of a particular threshold bound applying to a particular error-correcting code, since the number of locations C is a function of the parameters of the code.

However, the accuracy threshold for a particular model of computation is a more general concept that does not depend on a particular code – it is the highest possible accuracy threshold over all ways we can devise to perform fault-tolerant computation in that model. It is, in this sense, analogous to the concept of channel capacity in information theory [Sha48]. Provided that the threshold exists, the accuracy threshold for a particular model of computation depends on the noise model, the structure of space and time in the model, and how space and time may be used in the form of locality and parallelism.

We can consider specific points in this "space" where we fix, say, the noise model, the layout of qubits in space, aspects of the fault-tolerant circuit, and overhead constraints, but vary over, say, a subset of all error-correcting codes. For each choice of parameters, we obtain an accuracy threshold. Classically, varying over the code may not make sense, since the repetition code is excellent at correcting errors, so we may choose instead to vary the noise model or geometry or something else.

5.3 Concepts specific to quantum fault-tolerance

The TMR example introduced the essential concepts of fault-tolerance. All of these concepts carry over to the quantum domain. Classical circuits are replaced by quantum circuits. NAND gates are replaced by a universal set of quantum gates. The repetition code is replaced by a suitable quantum code, possibly from the pallette of codes introduced in Chapter 4. Even the concept of a rectangle constructed from a logic gate circuit followed by an error-correction circuit carries over. Most importantly, an accuracy threshold exists for fault-tolerant quantum computation and it has the same qualitative properties as the classical accuracy threshold – it is a constant and below threshold, accurate simulation incurs only a polylogarithmic overhead.

This section reviews some important differences between classical and quantum fault-tolerance. Some of the differences, summarized here, will be discussed in more detail in this section and in later chapters:

1. Noise models for quantum gates can be significantly more complicated as suggested in Chap-

ter 2. Stochastic gate faults may introduce an arbitrary quantum operation $\mathcal{E}(\rho)$ on the outputs of all failed gates and inputs of all failed measurements. This model is called **adversarial stochastic noise**. A slightly relaxed version of this model, depolarizing noise, is what is used in Chapter 8. Even more generally, the fault may couple the qubits to a shared environment that persists for durations of the computation. Such environments can create correlated coherent errors between fault locations. Amazingly, a threshold exists even under such adverse conditions [TB05, AGP06].

2. Unitary gates are reversible, so gates on multiple qubits, such as the CNOT, have multiple outputs and can spread errors more aggressively than the TMR example.

3. Classically, a NAND gate is universal, and we saw that a logical NAND gate has a transversal implementation on bits encoded in the repetition code. In the quantum setting, no universal set of transversal gates has been found for any quantum code, so other methods are needed to achieve universality. This issue of transversality in quantum fault-tolerance is considered in great detail in Chapter 6.

4. Quantum error-correction circuits may have quantum and classical subcircuits. The quantum subcircuits are responsible, at least, for gathering the error syndromes of the error-correcting code. The error syndromes can be measured immediately and the outcomes processed by reliable classical subcircuits. These classical subcircuits then control the recovery operation. An example is given in 5.3.1. Circuits for quantum error-correction are reviewed in Chapter 7 and many explicit constructions are given in Chapter 8.

5. Quantum states can be prepared, tested for accuracy, and stored for later use. They can be used in error-correction circuits and as "quantum software" resources for universality constructions [Pre99]. This is reviewed in 5.3.1 and Chapter 7.

6. Quantum storage is not considered reliable and faults can occur to qubits that do not participate in gates. Therefore, massive parallelization of the control hardware is necessary [ABO97].

7. "Cold", i.e., high fidelity, reference quantum states are necessary in the error-correction circuits; otherwise entropy cannot be removed from the system [ABO97].

For these reasons and others, lower bounds on the threshold tend to be smaller in the quantum setting than the classical setting.

5.3.1 Use of ancilla states in quantum components

This brief section reviews ways that quantum software is used in quantum fault-tolerant circuits. Standard ancilla states are used in error correction circuits. Gate-specific logical ancilla states are used in some logical quantum gates.

Use in error-correction

Steane observed that the process of quantum error-correction for CSS codes can be reduced to that of synthesizing certain encoded quantum states [Ste97]. We call this type of error-correction circuit **Steane-EC**.

Recall that CSS codes are generated by a set of X-type generators and a set of Z-type generators. Furthermore, the generators $\langle \bar{Z}_i, \bar{X}_i, i \in [k]\rangle$ of the centralizer $Z(S)/S$ can be written so that each $\bar{Z}_i$ is Z-type and $\bar{X}_i$ is X-type. Therefore, the logical CNOT gate, $\overline{CNOT}$, between two blocks of n encoded qubits is implemented transversally by n physical CNOT gates on corresponding qubits of each block; see Figure 5-6. This is because CNOT propagates X from control to target and Z from target to control, so for CSS codes $CNOT^{\otimes n}(S \otimes S)CNOT^{\otimes n} = S \otimes S$.

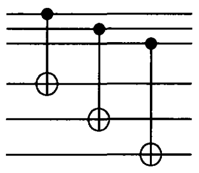


Figure 5-6: Transversal CNOT gate acting on two code blocks with $n = 3$ qubits each.

Figure 5-7 shows Steane-EC for an encoded qubit. The circuit uses ancilla prepared in the encoded quantum states $|\bar{+}\rangle$ and $|\bar{0}\rangle$ to measure the Z-type and X-type stabilizer generators, respectively. If the encoding circuits are not fault-tolerant, then the ancilla can be tested by a sequence of fault-tolerant measurements [Ste02]; see Chapter 8 for examples. The syndrome measurement outcomes are stored and processed by a separate classical circuit.

As we have seen, Steane's [[7, 1, 3]] code is a CSS code whose stabilizer has X-type and Z-type generators with the same supports. Therefore, $\bar{H}$ is transversal and is implemented by $H^{\otimes 7}$. This is a direct consequence of $C_1 = C_2$ in the CSS construction being dual-containing. Furthermore, $\bar{K}$ is transversal and is implemented by $(K^{\dagger})^{\otimes 7}$ since $(-i)^7 = i$ whereas the weight of every codeword in $C_2^{\perp}$ is divisible by 4 (i.e. $C_2^{\perp}$ is doubly even).

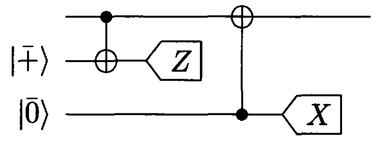


Figure 5-7: The Steane-EC circuit measures the stabilizer generators using only two transversal interactions with the encoded qubit. For an $[[n, 1, d]]$ code, each interaction uses an ancilla qubit prepared in either $|\bar{+}\rangle$ or $|\bar{0}\rangle$. The interaction with $|\bar{+}\rangle$ measures the Z-type stabilizer generators by taking parities of the appropriate Z-basis measurement outcomes. Similarly, the interaction with $|\bar{0}\rangle$ measures the X-type generators by taking parities of X-basis measurement outcomes. This circuit is correct because the generators are in the stabilizer of the ancilla and the circuit is logically an identity gate when there are no faults.

Use in universality constructions

The Steane code has transversal $\overline{CNOT}$, $\overline{H}$, and $\overline{K}$ gates, but this is not a universal set by the Gottesman-Knill theorem. Quantum software methods can be used to implement a logical T gate for this code [Pre98]. Broadly speaking, quantum software methods create an encoded ancilla state through online preparation and verification, interact this ancilla with codeblock(s), and measure some of the codeblocks. The methods have been generalized using quantum teleportation techniques to enable universal sets of gates to be constructed for any stabilizer code [Got98b, GC99].

We follow [AGP06] to construct the gate $\bar{T}$ for the $[[7, 1, 3]]$ code. Consider the circuit in Figure 5-8 for implementing a diagonal gate $\mathrm{diag}(1, e^{i\theta})$ up to an irrelevant global phase. When $\theta = \pi/4$, the gate $U_z(\theta) = T$, so we would like to implement this circuit on qubits encoded in the $[[7, 1, 3]]$ code. $U_z(2\theta)$ is K, which is transversal for the 7 qubit code. The Z-basis measurement can be implemented transversally and fault-tolerantly as well. The parity of the outcomes is the eigenvalue of $\bar{Z}$, and the outcomes can be corrected classically before the eigenvalue is computed, because they are encoded in C_1. Finally, we have already seen that $\overline{CNOT}$ is transversal. Therefore, the only part of this circuit that may not be fault-tolerant is the circuit that prepares $|\overline{A_{\pi/4}}\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\pi/4}|\bar{1}\rangle)$.

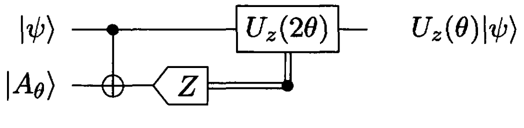


Figure 5-8: This circuit implements the gate $U_z(\theta) = \exp(-i\theta Z/2)$ using an ancilla prepared in the state $|A_\theta\rangle = U_z(\theta)|+\rangle$. The gate $U_z(2\theta)$ is conditionally applied if the outcome of the Z-basis measurement is -1.

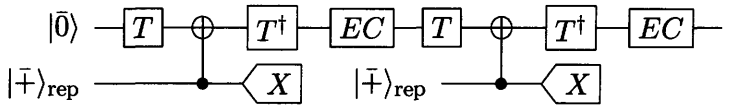
The state $|A_{\pi/4}\rangle$ is an eigenstate of $TXT^\dagger = KX$. Since this operator is transversal for the

$[[7,1,3]]$, it can be measured fault-tolerantly using Shor's cat state method [Sho96]. Shor's cat state method is a quantum software method, which can also be used for error-correction (**Shor-EC**), that allows a transversal operator M to be measured fault-tolerantly. An ancilla is prepared in the cat state $|+\rangle$ and encoded in the repetition code

$$|\bar{+}\rangle_{\text{rep}} := |00\dots0\rangle + |11\dots1\rangle. \tag{5.9}$$

The encoding circuit for the repetition code is not fault-tolerant, so the state must be verified by parity measurements so that any single fault producing a high weight error is detected, like with the Steane-EC ancilla states. The verified ancilla is then coupled via a transversal controlled-M to the encoded qubit being measured. Finally, the ancilla is measured and the eigenvalue of M is given by the parity of the outcomes. Faults can produce an incorrect outcome, so the process generally must be repeated.

The full circuit for preparing $|\overline{A_{\pi/4}}\rangle$ is shown in Figure 5-9. The $|\overline{A_{\pi/4}}\rangle$ state is accepted if both $\overline{TXT^\dagger}$ measurements agree and both syndrome measurements have trivial syndrome outcomes, otherwise it is rejected and the process restarted. The syndrome measurements ensure that single faults cannot propagate errors between the measurement subcircuits, causing them to agree without us knowing. The input $|\bar{0}\rangle$ state is prepared and verified in the same way it is prepared for Steane-EC.



$|\bar{0}\rangle$
T
$T^\dagger$
EC
T
$T^\dagger$
EC
$|\bar{+}\rangle_{\text{rep}}$
X
$|\bar{+}\rangle_{\text{rep}}$
X

Figure 5-9: This circuit prepares $|\overline{A_{\pi/4}}\rangle$ fault-tolerantly using Shor's cat state method. The boxes marked EC perform a syndrome measurement like Figure 5-7 but do not correct the identified errors. The ancilla is accepted if both $\overline{TXT^\dagger}$ measurements agree and both EC measure a trivial error syndrome. The order of T and $T^\dagger$ are the reverse of what is expected because $\bar{K} = (K^\dagger)^{\otimes 7}$.

5.3.2 Quantum accuracy threshold

As in the classical case, a rigorous proof of the threshold theorem for quantum computation makes use of the concept of a rectangle consisting of a logical gate and error-correction; see Figure 5-10. For codes of distance less than 5, the concept of an **extended rectangle**, or ex-Rec, provides a clean way to account for failures caused by pairs of faults in adjacent rectangles [AGP06]. An ex-Rec consists of a rectangled along with its *preceeding* 1-EC(s) on the input code block(s). We made use of a conceptually similar idea when we counted faults in the majority voting gates preceeding the classical NAND rectangle. We discuss this concept in more detail in Chapter 7.

Locations in a quantum circuit are defined to be gates, single-qubit state preparations, measurement steps, or memory locations where a qubit is not involved in any gate. After one level of recursive simulation, every location (denoted as 0-Ga) is mapped onto a rectangle or 1-rectangle (1-Rec), a space-time region in the simulated circuit, which consists of the encoded gate (1-Ga) followed by error correction (1-EC), as shown in Figure 5-10. For transversal gates, the 1-Ga consists of performing the 0-Ga's on each qubit in the block(s).

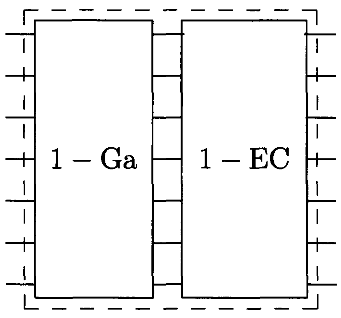


Figure 5-10: A 1-rectangle (1-Rec), indicated by a dashed box, which replaces a single-qubit 0-Ga location. The 1-Rec consists of the encoded fault-tolerant implementation of the 0-Ga (1-Ga) followed by an error correction procedure (1-EC).

The quantum threshold theorem can be proven in a beautiful non-inductive, syntactic way [Ali07]. The excellent references [AGP06, Ali07, AGP08] describe this proof technique and its implications for several noise models, and we do not review the proof here. An element that we use directly in Chapter 8 is the concept of **correctness** defined in Figure 5-11. An extended rectangle is said to be correct if an ideal decoder can be "pushed back through the rectangle" to give an ideal quantum gate. A level-1 exRec, or 1-exRec, is **bad** if it contains more than t faults and if it is not bad it is **good**. A 1-exRec satisfies the **exRec-Cor property** if the 1-Rec contained in a good 1-exRec is correct. For a more refined count, a set of locations in a 1-exRec is **benign** if the 1-Rec contained in the 1-exRec is correct when faults are placed at locations in the set. If a set of locations is not benign then it is **malignant**.

Theorem 14 (Quantum accuracy threshold for independent stochastic noise [AGP06]) *Suppose that fault-tolerant components can be constructed such that all* 1*-exRecs obey the exRec-Cor property, and such that ℓ is the maximal number of locations in a* 1*-Rec, d is the maximal depth of a* 1*-Rec, and ϵ_0^{-1} is the maximal number of pairs of locations in a* 1*-exRec. Suppose that independent*

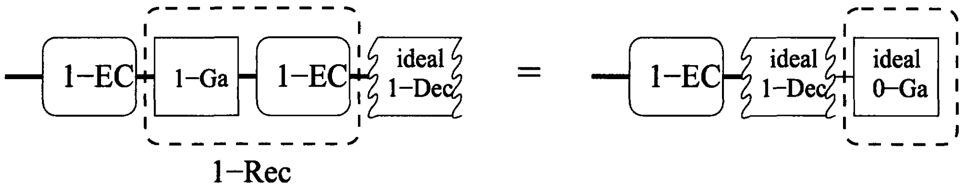


Figure 5-11: Syntactic definition of correctness for an extended rectangle [AGP06].

stochastic faults occur with probability $\epsilon < \epsilon_0$ at each location in a noisy quantum circuit. Then for any fixed δ, any ideal circuit with L locations and depth D can be simulated with error δ or better by a noisy circuit with L^* locations and depth D^*, where

$$L^* = O(L(\log L)^{\log_2 \ell}),\ D^* = O(D(\log L)^{\log_2 d}).$$

For the standard approach, using a concatenated quantum error-correcting code, and for adversarial stochastic noise, there are several lower bounds we can quote. A bound for the well-known [[7, 1, 3]] code is 2.7×10^{-5} [AGP06]. When constrained to a two-dimensional lattice, and using some flow map techniques [SCCA06], a lower bound for the [[7, 1, 3]] is 3.6×10^{-5} compared to 1.8×10^{-5} for the same circuits unconstrained [SDT07]. The bound for the non-local setting was significantly improved to 1.9×10^{-4} in [AC07] using the [[9, 1, 3]] Bacon-Shor subsystem code. See Chapters 8 and 9 for more thorough summaries of threshold values.

The threshold theorem is a very strong theorem that can be stated and proven for noise models that are even more challenging than adversarial stochastic noise. It has been proven for correlated models [AKP06] and non-Markovian models [TB05]. There is reason to believe that the thresholds for these models is much better than the lower bound that can be proven, and it is an open problem to prove a rigorous statement to this effect [AGP06].

5.4 Important developments beyond the standard approach

What we have called the standard approach is merely one approach to fault-tolerant computation. Some important alternative approaches exist and other developments have occurred beyond the introduction given in this chapter. This section briefly surveys some of these alternatives without going into much detail.

Information theoretic techniques have been used in classical settings to obtain several new results for classical fault-tolerant computation [Eva94]. The techniques bound the fraction of information that can cross a noisy channel and so relate fault-tolerant computation to the idea of a capacity for noisy computation [Eli58]. The main results, which have built on the work of many authors since von Neumann [WC63, DO77, GG94, Fed89, RS89, RS91, HW91, Pip85, Pip89, PST91], are as follows. For binary symmetric channels with probability $\epsilon = (1-\eta)/2$ and a basis consisting of gates with k inputs, a lower bound on the depth of a reliable circuit is $\log_k^{-1}(k\eta^2)$. The size of a reliable circuit is $O(c\log c)$ where c is the size of the noiseless circuit. The sensitivity of a function is the maximum over all inputs of the number of bits that change the function value when flipped individually. Functions with sensitivity s require reliable circuits of size $\Omega(s\log s)$. Reliable computation is impossible for $\epsilon \geq 1/2 - 1/2\sqrt{k}$, giving an upper bound on the threshold. For k odd, the threshold is given by

$$\beta_k = \frac{1}{2} - \frac{2^{k-2}}{k\binom{k-1}{\frac{k-1}{2}}}, \tag{5.10}$$

which is tight; for large k, $\beta_k \approx 1/2 - \sqrt{\pi}/2\sqrt{2k}$ [ES03]. For example, for 3-input gates the threshold is $1/6 \approx 0.166$ [HW91], and for noisy 2-input NAND gates the threshold is $(3-\sqrt{7})/4 \approx 0.088$ [EP98]. In both cases, reliable computation at higher error is impossible. Some thresholds for reversible circuits in one and two dimensions are given in [BR05].

These results are obtained for a different circuit construction than we have considered in the standard approach. Concatenation is not necessary if we can find a code family with good distance that allows simple-enough error-correction rectangles. The repetition code has the largest possible distance. Furthermore, it has constant depth error-correction circuits that can be constructed explicitly using expander graphs [Pip85].

What are the corresponding results for quantum circuits? The picture is less complete. The tightest upper bound on the quantum accuracy threshold is $p_{th} \leq 1 - \Theta(1/\sqrt{k})$ for depolarized k-qubit unitary gates and noiseless single qubit gates [KRUdW08]. This matches the scaling behavior of the classical bound. For $k = 2$ and CNOT gates, $p_{th} \leq 0.293$. Additional upper bounds have been given in [Raz04, BCL$^+$06]. The best known lower bound, in constrast, is 1.04×10^{-3} [AGP08]. Therefore, the separation between upper and lower bounds is presently about two orders of magnitude.

In the classical setting, the best thresholds are achieved by using a family of codes whose minimum distance scales linearly with the block size and is as large as possible. The quantum Singleton (or MDS) bound for an $[[n,k]]$ binary quantum code is $d \leq (n-k+2)/2$ [Rai99a]. Therefore, the best quantum codes encoding one qubit have distance $(n-1)/2$ as opposed to the classical case where the distance can be as much as n. Most families of codes that have been explicitly constructed have a distance that does not even scale linearly with the block size, although there are some [ATL01]. The surface codes are an example of a family of $[[n, 1, O(\sqrt{n})]]$ codes that, despite their distance, correct many high weight errors. This is one of the only families of codes with an effectively good distance that scales linearly with the block size. Since there are also very simple, local error-correction circuits for this code, a threshold exists without concatenation [DKLP02].

Local error-correction circuitry is highly desirable for implementing fault-tolerant circuits that are layed-out in one or two spatial dimensions. Surface codes have simple local error-correction circuits. The Bacon-Shor codes can also be used in a mode where error-correction is entirely local. Furthermore, and somewhat amazingly, it is possible to perform a universal set of gates using only local operations in two dimensions on surface codes [RHG07, BMD07, FSG08]. The threshold for these schemes is roughly 6×10^{-3} with a large classical memory. An improvement in the logical error rate over the physical error rate does not occur until the surface is greater than roughly 7×7 qubits. The effects of a limited classical memory and a practical syndrome decoding time may also affect the threshold of this scheme, but it remains a very promising scheme.

There are several methods for constructing error-correcting circuits for stabilizer codes and for preparing quantum states used in error-correction [Sho96, Ste97, Kni05b]. The complexity of error-correction is important in determining the accuracy threshold for a code or family of codes. There is no example of an error-correction circuit with constant depth when that circuit includes the classical syndrome decoder as a reversible quantum circuit; however, the syndrome extraction alone can be done in constant depth [Ahn04].

Fault-tolerant quantum computation need not use error-correcting codes. Concatenated *error-detecting* codes also lead to a provable threshold [AGP08, Rei06a]. The idea was first given by E. Knill in a series of papers [Kni05a, Kni05b, Kni04a, Kni04b]. This scheme leads to thresholds on the order of 1% but at a significant (constant) cost in overhead. In Chapter 8, we show Knill's simulation results for this scheme together with our results for error-correcting codes to compare the relative overhead.

Finally, schemes for physically motivated noise models have been successful. A scheme by Aliferis and Preskill has given a factor of 5 improvement in the accuracy threshold for biased noise [AP07].

5.5 Conclusion

This introduction to fault-tolerance prepares us for what lies ahead in the later parts of this dissertation. The remarkable result that quantum fault-tolerant computation is possible provides the main reason for continued hope that large scale quantum computation will one day be achieved. The chapter has defined and introduced central concepts of a standard approach to classical and quantum fault-tolerant computing so that we may explore that approach in more detail in later chapters. Finally, the review in Section 5.4 reminds us that there are other approaches to fault-tolerant computing and that there are significant questions still unanswered in the quantum setting.

Chapter 6

Limits to a standard approach to fault-tolerant logic gate construction

6.1 Introduction

Transversal gates are an important class of fault-tolerant gates introduced in Chapter 5, where we reviewed that fault-tolerant components limit the introduction and spread of error within themselves. Transversal gates act bitwise between corresponding qubits in each code block, limiting error-propagation to a small number of interacting qubits. Classically we know that NAND is itself both universal and transversal on the repetition code, so the classical problem can be solved by considering this gate alone.

A long-standing open problem in fault-tolerant quantum computation has been to find a quantum code that admits a universal set of transversal gates. If such a code also admitted a fault-tolerant error-correction procedure, then the structure of fault-tolerant logic gates would be uniform, as in the classical setting, potentially simplifying the code architecture discussed in Chapter 7. Furthermore, it might be possible to give an exceedingly simple lower bound on the accuracy threshold in realistic geometrically local settings, since quantum software approaches could be avoided in universal gate set construction (though simple quantum software is still likely needed in error-correction).

Unfortunately, there are no known examples of a quantum code with a universal set of transversal gates. The most tractable examples of logic gates have been for stabilizer codes, and obtaining non-stabilizer codes systematically has been difficult until the new result discussed in Chapter 4, so little work has appeared about computation on non-stabilizer codes. Given the experience with stabilizer codes, it is generally believed that a quantum code with a universal set of transversal gates does not exist. Analyzing transversal gates for general quantum codes is very difficult, and the question is unresolved even for stabilizer codes (prior to our work), so we restrict the discussion to binary stabilizer codes in this chapter.

A main result of this chapter is that a universal set of transversal gates does not exist for even one of the qubits encoded into a binary stabilizer code. Since binary stabilizer codes are the mainstay of fault-tolerant quantum computation, this strongly supports the idea that other primitives, such as quantum teleportation [GC99], injection by teleportation [Kni05a], or state distillation [BK05], are in fact necessary for universal fault-tolerant quantum computation. In the concluding section, we discuss the assumptions we have made and how they might be lifted in the search for elusive codes with interesting transversal gates.

Throughout the chapter we consider an $[[n,k,d]]_2$ stabilizer code $C(S)$ with stabilizer S and projector $P_{C(S)}$. A block is a set of n qubits encoding k qubits into $C(S)$. Suppose throughout that there are $r \geq 1$ blocks. If $C(S)$ happens to be a subsystem code, it can be viewed as a subspace code by taking the union of the logical Hilbert space and the subsystem Hilbert space. The subsystem code $[[n,k,l,d]]$ becomes a subspace code with the same stabilizer S and new parameters $[[n,k+l,d' \leq d]]$. Regardless of how the subsystem code is viewed, the observations in this section apply equally well to subsystem codes because they only rely on the stabilizer and not on a basis for the codespace. In Section 6.3.3, it is important to distinguish subsystem from subspace codes because we choose a basis for the codespace. At that point, we argue that the results hold for subsystem codes as well as subspace codes.

Section 6.2 discusses what it means to compute on information encoded into a stabilizer code and reviews the important definitions needed to proceed through the chapter. Section 6.3 delves into a so-called subcode method for proving structure theorems about local gates on stabilizer codes. In this section, we prove the first new results of the chapter, which extend the subcode method to the full automorphism group and the group of transversal gates. The section concludes with a proof that transversal gates are not a universal set and full automorphisms are not a universal set. Finally, Section 6.4 applies the structure theorems to explore non-Clifford transversal gates for CSS codes, giving new examples and an interesting conjecture. Section 6.5 concludes with open problems.

6.2 Structure of computation on stabilizer codes

This section discusses in detail what it means to compute on information encoded in a stabilizer code. It begins in 6.2.1 by defining concepts related to transversality and giving examples of types of fault-tolerant gates on stabilizer codes. Next, encoded universality (6.2.2) and the consequences of non-Clifford gates are discussed (6.2.3). Finally, concepts of a subcode approach for grasping the structure of local gates on stabilizer codes are presented in 6.2.4 and 6.2.5.

6.2.1 Transversal logic gates on stabilizer codes

Definition of transversality

A gate $U \in U(2^{nr})$ is an **r-block logical gate on** $C(S)$ if $[P_{C(S)}^{\otimes r}, U] = 0$. The condition says that the code space is invariant under U,

$$UP_{C(S)}^{\otimes r} = P_{C(S)}^{\otimes r}U \iff UP_{C(S)}^{\otimes r}U^\dagger = P_{C(S)}^{\otimes r}, \tag{6.1}$$

so codewords are mapped to codewords in each block by U. Let E_S be an isometry that encodes k bare qubits into $C(S)$. An r-block logical gate U **implements** $\bar{V}$ on the code if $V = E_{C(S)}^\dagger U E_{C(S)}$ where $E_{C(S)}$ is any encoder into $C(S)$. We may also relax and say that $\bar{V}$ is a logical gate.

A **transversal gate acting on r blocks** is an nr qubit unitary U that (a) factors into $U_1 \otimes U_2 \otimes \cdots \otimes U_n$ where each U_i is an r qubit unitary that acts on the ith qubit of each block and (b) U is an r-block logical gate on $C(S)$. See Figure 6-1 for an illustration of a transversal gate applied to r encoded blocks of n qubits each.

Conditions (a) and (b) are not enough to guarantee that U acts nontrivially on $C(S)$; any element of S meets both conditions but an element of S acts like logical identity $\bar{I}$ on the code space. However, we include these trivial gates in the set of transversal gates. Denote the set of transversal gates on r blocks of $C(S)$ by $\mathrm{Trans}_r(S)$ and note that this is a group.

The definition of transversality includes a large class of gates, but it can still be more general. Let us consider how conditions (a) and (b) can each be generalized.

For example, condition (a) can be expanded to include the case where a transversal gate is sandwiched between permutations of the qubit coordinates. The permutations should not exchange qubits between blocks, since this can spread errors, but it is permissible to permute qubits within each block. Such a **permutation transversal gate** has the form $\pi_F\left(\bigotimes_{i=1}^n U_i\right)\pi_I$ where the

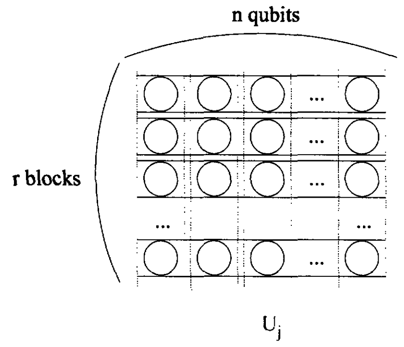


Figure 6-1: Illustration of a transversal gate on r blocks of n qubits each. The blocks are represented by a collection of circles (qubits), grouped into boxes of n. The r blocks undergo a transversal gate whose unitaries U_j act on qubits in the [blue] boxes with rounded edges.

permutations satisfy $\pi_F\pi_I = \prod_{j=1}^{r}\pi_j$ and π_j acts only on the jth block. Unfortunately, these gates are difficult to work with since the permutations act along the rows of Figure 6-1 while the gates act on the columns, and each π_j may be different.

An example of a code with a permutation transversal gate is the [[4, 2, 2]] code. In this code, the gate $\overline{\Lambda(X)[1,2]}$ is achieved by $\pi_{(2,4)}$, where the coordinate permutation is given in cycle notation. Another example is the [[5, 1, 3]] code. The [[5, 1, 3]] has no transversal Hadamard gate. However, it has a permutation transversal Hadamard gate achieved by $H^{\otimes 5}$ followed by $\pi_{(12)(34)}$.

Condition (b) requires the gate to map inputs from $C(S)$ to outputs in $C(S)$. However, a logic gate could map inputs encoded in one input code to outputs encoded in a different target code. Transforming the code may not be a problem if the dimension of the code space does not change and the distance of the target code is not greatly reduced. One way of defining such a **code-transforming transversal gate** U is by the property $U\left(\bigotimes_{j=1}^{r} P_{C(S_j)}\right) = \left(\bigotimes_{j=1}^{r} P_{C(S'_j)}\right)U$.

The polynomial codes are important examples of codes with code-transforming transversal Toffoli gates, but they are defined on higher dimensional quantum systems, not qubits. The polynomial codes are discussed in more detail in Chapter 8.

Finally, it is important to mention that measurement has been purposely excluded from these definitions. Including transversal measurement in the definition certainly makes the scheme universal since we can use quantum software methods. The quantum software methods do require ancilla states, but most state preparations, including initial states, appear to unavoidably require non-fault-tolerant circuits and verification testing as reviewed in Chapter 7. A notable exception is the [[9, 1, 3]] Bacon-Shor code whose $|\bar{0}\rangle$ and $|\bar{+}\rangle$ states can be prepared without verification testing.

Code	Transversal	Transv. with permutation	Not transversal
$[[4,2,2]]$	$\overline{CNOT}$,$\overline{H_1 \cdot H_2 \cdot SWAP[1,2]}$	$\overline{CNOT[1,2]}$	Non-Clif.
$[[5,1,3]]$	$\overline{KH}$, $\overline{M_3}$	$\overline{H}, \overline{K}$	$\overline{CNOT}$, Non-Clif.
$[[7,1,3]]$	$\overline{CNOT}$, $\overline{H}$, $\overline{K}$	–	Non-Clif.
$[[n,1,\sqrt{n}]]$	$\overline{CNOT}$	$\overline{H}$	Non-Clif.
$[[15,1,3]]$	$\overline{T}$, $\overline{K}$, $\overline{CNOT}$	–	$\overline{H}$
$[[2^m-1,1,3]]$	$\{\overline{T_j}, j \in [m]\}$, $\overline{CNOT}$	–	$\overline{H}$

Table 6.1: The table lists stabilizer codes introduced in Chapter 4 and their transversal encoded logic gates. The first column gives the code parameters. The $[[7,1,3]]$ is Steane's code. The $[[n,1,\sqrt{n}]]$ family refers to the Bacon-Shor codes. The $[[2^m-1,1,3]]$ family are constructed from Reed-Muller codes. The second column lists gates that are known to be transversal for that code. The third column lists gates that are permutation transversal. The final column lists gates for which either a) no transversal construction is known or b) the gate is impossible to do transversally by prior known results [Rai99b]. M_3 is a three-qubit Clifford operation [Got97, Got98b] and $T_j := \text{diag}(1, e^{i\pi/2^{j-2}})$, which includes I, Z, K, T, and so on.

Examples of transversal gates

It is well known how to find transversal gates whose terms U_i are in the Clifford group. Property (b) becomes $UgU^\dagger \in S^{\otimes r}$ for all $g \in S^{\otimes r}$, so it becomes enough to look for gates that preserve the stabilizer under conjugation. For single qubit transversal gates, this is exactly the problem of computing a subgroup of the automorphism group of the classical additive code associated with S, a procedure that has been automated in computer algebra packages such as GAP [GAP07] and MAGMA [WCP97]. Finding transversal gates whose U_i are non-Clifford is more difficult. There does not appear to be a known systematic approach, but some gates can be found. Table 6.1 summarizes examples of transversal gates for the stabilizer codes introduced in Chapter 4.

There is a hierarchy of gates that appears in the study of quantum teleportation called the $\mathbf{C_k}$ **hierarchy** [GC99]. The set $C_1^{(n)}$ is the n-qubit Pauli group G_n. The remaining sets are defined recursively by

$$C_k^{(n)} := \{U \in U(2^n) \mid UvU^\dagger \in C_{k-1}^{(n)},\ \forall v \in C_1^{(n)}\}. \tag{6.2}$$

$C_2^{(n)}$ is the Clifford group, but $C_k^{(n)}$ is not a group for $k \geq 3$. It is clear that $C_k^{(n)} \subseteq C_{k+1}^{(n)}$. Both T and Toffoli conjugate Paulis to Cliffords, so they are in C_3. The structure of C_k is well understood only for a small number of qubits [ZCC07]. The main reason for mentioning this hierarchy is to observe that $T_j \in C_{j-1}$ for $j > 1$, which is proven in [ZCC07]. None of the examples in Table 6.1 seem to have transversal gates outside of the C_k hierarchy and the same is true for all examples known to me.

6.2.2 Encoded universality

Transversal gates with $U_i \in C_2$ are easy to study, as we have already said, since there is a poly(n) algorithm for testing whether or not such a gate is a logic gate and finding what action it has on the qubits encoded in the code. However, transversal gates implemented by Cliffords can only give logical gates that act like elements of the Clifford group, since the encoded qubits are described by $Z(S)$, and $Z(S)$ is mapped to itself by such a gate. Therefore, by the Gottesman-Knill theorem, such gates are not a universal finite basis on the qubits encoded into $C(S)$, and it is necessary to add some other gate to such a set.

A set $\mathcal{A}$ of r-block logical gates is **computationally universal on** $C(S)$ if given any r-block logical gate U,

$$\forall \epsilon > 0,\ \exists V_1, \ldots, V_{\eta(\epsilon)} \in \mathcal{A},\ \text{s.t.}\ \left\| U P_{C(S)}^{\otimes r} - \left(\prod_i V_i \right) P_{C(S)}^{\otimes r} \right\| < \epsilon. \tag{6.3}$$

We do not worry about the approximation efficiency, since our results in this chapter show that particular choices of $\mathcal{A}$ are *not* computationally universal on $C(S)$.

6.2.3 Generalized stabilizers and the LU-LC conjecture

Non-Clifford gates can be transversal yet not take the stabilizer to itself, but to a subgroup of the generalized stabilizer. The **generalized stabilizer** $\mathcal{I}(S)$ of $C(S)$ is the group of all unitary gates that fix the code space,

$$\mathcal{I}(S) := \{U \in U(2^n) \mid U|\psi\rangle = |\psi\rangle,\ \forall \psi \in C(S)\}. \tag{6.4}$$

The transversal T gate on the $[[15, 1, 3]]$ is an example of a gate that does not map Paulis to Paulis. It maps $\bar{X} = X^{\otimes 15}$ to $(\frac{1}{\sqrt{2}}(X - Y))^{\otimes 15}$. This is a representative of $\frac{1}{\sqrt{2}}(\bar{X} + \bar{Y})$ but has many more terms. These terms are contributions from another element in $\mathcal{I}(S)$. The $[[9, 1, 3]]$ is another example. The gate $e^{i\phi Z_1} e^{-i\phi Z_2} \in \mathcal{I}(S) \setminus S$ is transversal and acts like the logical identity. However, it does not map $\bar{X} = X^{\otimes 9}$ back to the Pauli group.

It is perhaps not too surprising that stabilizer codes have freedom to be rotated back into themselves by non-Clifford transversal gates, since they are subspaces that encode k qubits. We expect to be able to rotate the encoded qubits, though it is perhaps surprising that some of those rotations can be done with local operations. As we have already seen in Chapter 4, a **local unitary (LU) gate** on n qubits is simply an element of $U(2)^{\otimes n}$ and a **local Clifford (LC) gate** is an element of $(C_2^{(1)})^{\otimes n}$ A **local equivalence of stabilizer codes** $C(S_1)$ **and** $C(S_2)$ is a local unitary U such that $U|\psi_1\rangle \in C(S_2)$ for all $|\psi_1\rangle \in C(S_1)$.

One might expect that local unitary equivalences of stabilizer *states* must be local Clifford gates. This expectation is formally known as the LU-LC conjecture.

Conjecture 15 (LU-LC [dNDM05]) *If two stabilizer states are locally equivalent, then they are locally Clifford equivalent.*

Shockingly, this conjecture turns out to be false – after our work was completed, and after our own attempts to prove the conjecture [ZCCC07], it was shown that there are stabilizer states that can be transformed into one another by local unitary gates but not by local Clifford gates [JCWY07]. These counterexamples show that there are shocking subtleties about the stabilizer formalism, and there is still much to do to understand these subtleties fully.

6.2.4 Automorphisms and semi-Cliffords

The transversal gates on one block are exactly the set of all local unitary equivalences from $C(S)$ to itself. Likewise, the set of all single block permutation transversal gates is exactly the full automorphism group of a stabilizer code. The **full automorphism group Aut(S) of a stabilizer code** $C(S)$ is the collection of all encoded logic gates acting on $C(S)$ that have the form $P_\pi U$ where P_π enacts a coordinate permutation π and $U = U_1 \otimes \cdots \otimes U_n$ is a local unitary. P_π acts on computational basis states like $P_\pi|b_1 b_2 \dots b_n\rangle = |b_{\pi(1)} b_{\pi(2)} \dots b_{\pi(n)}\rangle$. This group is formally a **semidirect product group** [Asc00] defined in the following way, as introduced by Rains [Rai99b]. It is the subgroup of logical operations contained in a group $(S_n, U(2)^{\otimes n}, \nu)$ where $\nu : S_n \to \mathrm{Aut}(U(2)^{\otimes n})$ is

$$\nu(\pi)(U_1 \otimes \cdots \otimes U_n) := U_{\pi(1)} \otimes \cdots \otimes U_{\pi(n)} \tag{6.5}$$

and S_n is the **symmetric group on n items**, i.e., all permutations of n items. The triple $(S_n, U(2)^{\otimes n}, \nu)$ is denoted $S_n \ltimes U(2)^{\otimes n}$ when ν is implied. An element of $S_n \ltimes U(2)^{\otimes n}$ acts on both the qubit coordinates and the codewords in Hilbert space, but it maps the action on the coordinates into an action on the local unitary operator in $U(2)^{\otimes n}$. The definition of the product of two elements in a semidirect product group $S_n \ltimes U(2)^{\otimes n}$ is

$$(\pi_1, U)(\pi_2, V) := (\pi_1\pi_2, (U_{\pi_2(1)}V_1) \otimes \cdots \otimes (U_{\pi_2(n)}V_n)). \tag{6.6}$$

The single block transversal gates are the subgroup $\text{Trans}_1(S) = \{(\pi, U) \in \text{Aut}(S) \mid \pi = ()\}$ without permutation. The single block gates accomplished by permuting the qubits are the subgroup $\text{PAut}(S) := \{(\pi, U) \in \text{Aut}(S) \mid U = I\}$. Generally there are elements of $\text{Aut}(S)$ outside of $\text{PAut}(S) \ltimes \text{Trans}_1(S)$. Consider the $[[5,1,3]]$, where $H^{\otimes 5}$ followed by $(12)(34) \notin \text{PAut}S_{[[5,1,3]]}$ is an automorphism, i.e., the permutation transversal logical Hadamard gate. The two groups can also be equal, as happens with the $[[4,2,2]]$ – $\text{PAut}(S_{[[4,2,2]]}) = S_4$ and the group $S_4 \ltimes \text{Trans}_1(S_{[[4,2,2]]})$ corresponds to the entire group of logical Clifford gates on the $[[4,2,2]]$, which is all of $\text{Aut}(S)$ (see Table 6.1).

There is a classification of gates that will be useful in Section 6.3 when discussing limitations on the form of transversal gates. A unitary gate is Clifford if it is in $C_2^{(n)}$. A unitary gate is **semi-Clifford** if it sends at least one maximal abelian subgroup of G_n to another maximal abelian subgroup of G_n. A unitary gate is **generalized semi-Clifford** if it sends the span of the elements in at least one maximal abelian subgroup of G_n to the span of elements in another maximal abelian subgroup of G_n.

A semi-Clifford gate U can be expressed as $U = LDR$ where $L, R \in C_2^{(n)}$ and D is diagonal [ZCC07]. Indeed, $A := \langle Z_i, i = 1, \dots, n\rangle$ is a maximal abelian subgroup of G_n, and we know that there exist $R^\dagger, L \in C_2^{(n)}$ such that $R^\dagger AR$ is mapped to $LAL^\dagger$ by U. Therefore,

$$UR^\dagger ARU^\dagger = LAL^\dagger \iff (L^\dagger UR^\dagger)A(L^\dagger UR^\dagger) = A \tag{6.7}$$

$$\iff L^\dagger UR^\dagger = D \text{ diagonal} \iff U = LDR. \tag{6.8}$$

Similarly, a generalized semi-Clifford gate can be written as $LPDR$ where P is a permutation matrix, $L, R \in C_2^{(n)}$, and D is diagonal. Let $A := \langle Z_i, i = 1, \dots, n\rangle$ as before and let $\mathbb{C}[A]$ denote the vector space spanned by elements of A over the complex field. By definition, $U\mathbb{C}[A_1]U^\dagger = \mathbb{C}[A_2]$ for some maximal abelian subgroups A_1, A_2. We know that there exist $R^\dagger, L \in C_2^{(n)}$ such that $UR^\dagger\mathbb{C}[A]RU^\dagger = L\mathbb{C}[A]L^\dagger$. Therefore, $(L^\dagger UR^\dagger)\mathbb{C}[A](L^\dagger UR^\dagger)^\dagger = \mathbb{C}[A]$ which implies that $L^\dagger UR^\dagger$ is a monomial matrix PV where P is a permutation matrix and V is diagonal.

6.2.5 Minimal supports, elements, and subcodes

A **support** is a subset of the set of coordinates $[n]$. The support of a local unitary gate $P \in U(2)^{\otimes n}$, denoted supp P, is the set of all $i \in [n]$ such that P_i differs from the identity. The **weight** of P is the size of the support wt $P = |\text{supp } P|$. An element R is said to have **full support** if $\text{supp}(R) = [n]$. For example, the element $XZZXI$ has support $\{1,2,3,4\}$ and weight 4. It does not have full support.

A **minimal support** of S is a nonempty set $\omega \subseteq [n]$ such that there exists an element of S with support ω, but no elements exist with support strictly contained in ω, excluding the identity element. Let $\mathfrak{m}(S) \subseteq [n]$ be the union of the minimal supports of S. An element in S with a minimal support is called a **minimal element**. Let $M(S)$ denote the subgroup generated by all of the minimal elements of S. Suppose at least one element in S has support ω, then the **subcode of** S **associated with** ω is $\rho_\omega := \frac{2^{|\omega|-k}}{|S_\omega|} \operatorname{Tr}_{\bar{\omega}} P_{C(S)}$ where $\bar{\omega} := [n] \setminus \omega$ and $S_\omega := \{g \in S \mid \text{supp}(g) \subseteq \omega\}$ is a subgroup of S.

The subcode is generated by the elements of S with support contained in ω, where for each element the coordinates outside ω are removed. Indeed,

$$\rho_\omega = \frac{2^{|\omega|-k}}{|S_\omega|} \operatorname{Tr}_{\bar{\omega}} P_{C(S)} = \frac{2^{|\omega|}}{2^n |S_\omega|} \sum_{g \in S} \operatorname{Tr}_{\bar{\omega}} g. \tag{6.9}$$

The partial trace of a Pauli element over a set of coordinates is zero unless all of the coordinates are the identity, in which case $\operatorname{Tr}_{\bar{\omega}} g = 2^{\bar{\omega}} g_\omega$ where g_ω is obtained from g by tracing out the coordinates in $\bar{\omega}$. Therefore

$$\rho_\omega = \frac{1}{|S_\omega|} \sum_{g \in S_\omega} g_\omega. \tag{6.10}$$

If there are no elements in S with support ω then ρ_ω is a trivial stabilizer code projecting onto the whole space.

For example, $\{1,2,3,4\}$ is a minimal support of the [[5,1,3]] and $XZZXI$ is a minimal element with that support. $S_{\{1,2,3,4\}} = \{IIIII, XZZXI, YXXYI, ZYYZI\}$. The subcode has projector

$$\rho_{\{1,2,3,4\}} = \frac{1}{4}(IIII + XZZX + YXXY + ZYYZ), \tag{6.11}$$

so it corresponds to a [[4,2,2]] stabilizer code. The minimal supports are every set of 4 contiguous coordinates $\{1,2,3,4\}$, $\{2,3,4,5\}$, $\{3,4,5,1\}$, etc, so $\mathfrak{m}(S) = [5]$. Since $S = \langle S_\omega, \omega \text{ minimal} \rangle =$

$M(S)$, the $[[5,1,3]]$ is the subspace that is the intersection of the codes $C(S_\omega)$. The code $C(S_\omega)$ differs from the minimal code ρ_ω by the tensor product with some unencoded qubits, i.e. in this example, ρ_ω is a $[[4,2,2]]$ and $C(S_\omega)$ is a $[[4,2,2]] \otimes [[1,1,1]]$.

The term subcode is chosen because, if the identity coordinates are not removed, the classical code associated with the stabilizer of the new code is a subcode of the classical code associated with the original stabilizer.

6.3 Limitations on transversal gates for stabilizer codes

In this section, we extend Rains' approach to obtain several new results. The first result is a general form of a stabilizer code automorphism. This result has been independently discovered by David Gross and Martin Van den Nest [GdN08]. The second is a generalization of the approach to transversal gates acting on more than one block, leading to a general form for transversal gates on stabilizer codes.

6.3.1 Gates on a single encoded block

The main result of this section extends earlier work to show that $U \in \text{Aut } S$ if and only if

$$U = P_\pi L_1 \left(\bigotimes_{j=1}^{n} \text{diag}(1, e^{i\theta_j}) \right) L_2 \tag{6.12}$$

and U is a logic gate on S. Here, L_1 and L_2 are local Clifford gates, P_π is a product of swap gates enacting permutation π on the qubit coordinates, and $\{\theta_1, \ldots, \theta_n\}$ are angles. Loosely speaking, automorphisms of stabilizer codes are essentially diagonal up to local Clifford gates. For the result to be true, the stabilizer S must satisfy some additional, simple conditions to rule out trivial cases. In particular, $C(S)$ must not have an unencoded qubit or a Bell pair "tacked on". The goal of this section is to show why this is true and explain the new result.

We begin by reviewing a lemma originally proven by Rains. The lemma shows that transversal logical gates preserves stabilizer subcodes. The result will be meaningful because we can use its contrapositive – if a gate does not preserve subcodes, then it cannot be a transversal logical gate.

Lemma 16 (Subcode lemma [Rai99b]) *Let $\omega \subseteq [n]$ be a nonempty set of coordinates. Given* *a transversal gate $U = \bigotimes_{i=1}^{n} U_i$, let $U_\omega := \bigotimes_{i\in\omega} U_i$. Then*

$$U_\omega \rho_\omega U_\omega^\dagger = \rho_\omega. \tag{6.13}$$

Proof The transversal gate is an encoded logical gate, so $UP_{C(S)}U^\dagger = P_{C(S)}$ by definition. Taking the trace of both sides,

$$\mathrm{Tr}_{\bar\omega}\left[UP_{C(S)}U^\dagger\right] = \mathrm{Tr}_{\bar\omega} P_{C(S)} \iff \tag{6.14}$$

$$U_\omega \left[\mathrm{Tr}_{\bar\omega} P_{C(S)}\right] U_\omega^\dagger = \mathrm{Tr}_{\bar\omega} P_{C(S)}. \tag{6.15}$$

□

The following lemma of Van den Nest, leads to a classification of minimal subcodes of stabilizer codes.

Lemma 17 (Number of elements stabilizing minimal subcodes [dNDM05]) *Let ω be a minimal support of S and let A_ω denote the number of nonidentity elements in S_ω. Then $A_\omega \in \{1,3\}$.*

Proof By definition, there must be some element of S with support ω, so if there are no more, then $A_\omega = 1$. If there are two elements M, N with support ω, then their product MN must have support ω, otherwise ω is not minimal. So A_ω cannot be 2, but it can be 3. Suppose there is a fourth element M' with support ω. There are only three nonidentity Pauli operators, so one of them must appear twice at some coordinate in ω. By then we can form another product whose support is strictly contained in ω, so ω is not minimal. Therefore, A_ω cannot be greater than 3. □

Notice that when $A_\omega = 3$, $|\omega|$ must be even, otherwise the operators in S_ω do not commute. This result allows one to characterize the minimal subcodes.

Corollary 18 (Characterization of minimal subcodes [dNDM05]) *Let ω be a minimal support of S and let $I^{|\omega|} = \underbrace{I \otimes \cdots \otimes I}_{|\omega|\ times}$. Then the minimal subcode of S associated with ω is either*

$$\rho_\omega = \frac{1}{2}(I^{|\omega|} + g_\omega) \tag{6.16}$$

or

$$\rho_\omega = \frac{1}{4}(I^{|\omega|} + g_\omega + h_\omega + (gh)_\omega), \tag{6.17}$$

where g_ω and h_ω are commuting Pauli operators in S, restricted to ω, whose product also has support on ω. In the first case, ρ_ω projects onto a $[[|\omega|, |\omega|-1, 1]]$ code, and, in the second case, ρ_ω projects onto a $[[|\omega|, |\omega|-2, 2]]$ code.

There is a local Clifford gate that maps the first code into a code stabilized by $\langle Z^{|\omega|}\rangle$. This code is a classical code with one parity check. Similarly, the second code maps into $\langle Z^{|\omega|}, X^{|\omega|}\rangle$, which is a quantum code detecting one error and saturating the quantum Singleton bound.

The extent to which a stabilizer code can be described by its minimal subcodes depends on the particular stabilizer code. For example, the $GF(4)$-linear codes are one family of stabilizer codes that can be described completely by their minimal subcodes [Rai99b, dNDM05], i.e. $M(S) = S$ and $\mathfrak{m}(S) = [n]$. A **$GF(4)$-linear code** is a code whose stabilizer group is preserved by the transversal gates $C_\omega : X \mapsto Z \mapsto Y \mapsto X$ and $C_{\omega^2} : X \mapsto Y \mapsto Z \mapsto X$. The reason for the terminology "$GF(4)$-linear" is that these transversal gates correspond to multiplying by field elements ω and ω^2, so that the stabilizer group corresponds to a linear code over the alphabet $GF(4)$. A linear code is generated by its minimal elements. Indeed, if a vector v in a linear code is not minimal, then we can find another vector w whose support is contained in the support of v. There is some constant c such that $v - cw$ has lower weight and is contained in the code. It is clear that we can continue this process until we arrive at a minimal vector. The process gives us v expressed as a linear combination of minimal vectors.

The doubly even dual-containing CSS codes, such as the $[[7, 1, 3]]$ Steane code [Ste96] and the $[[23, 1, 7]]$ Golay code [Rei06a], are familiar examples of $GF(4)$-linear codes. Codes such as these have transversal Phase $\bar{K}$ and Hadamard $\bar{H}$ gates implemented bitwise (i.e., by applying the gate or its conjugate to each bit of the code). Therefore, all of their minimal subcodes have $A_\omega = 3$, and all of their transversal gates are Clifford. The codes are designed this way – they have transversal $\overline{CNOT}$, $\overline{H}$, and $\overline{K}$, so any logical Clifford is transversal.

Now we will focus on error detecting codes with parameters $[[|\omega|, |\omega| - 2, 2]]$.

The case $|\omega| = 2$ is special. A $[[2, 0, 2]]$ is a Bell pair $(|00\rangle + |11\rangle)/\sqrt{2}$. The gate $V \otimes V^*$ maps a Bell pair to itself for any $V \in U(2)$. Indeed, by direct matrix multiplication, one can verify that $U \otimes I|\text{Bell}\rangle = I \otimes U^T|\text{Bell}\rangle$, so $(U^T)^{-1} = U^*$ proves the fact.

Lemma 19 (Clifford-only lemma [Rai99b]) *Fix $2m := |\omega| \geq 4$ and let $C(S)$, and $C(S')$ be $[[|\omega|, |\omega| - 2, 2]]$ stabilizer codes. If U is a local equivalence from $C(S)$ to $C(S')$, then U is a local Clifford gate.*

Proof We must show that every $U \in U(2)^{\otimes 2m}$ satisfying $U\rho_{[[2m,2m-2,2]]}U^\dagger = \rho_{[[2m,2m-2,2]]}$ is a local Clifford operator. We use the notation $\sigma_x := X$, $\sigma_y := Y$, and $\sigma_z := Z$. Recall that any $V \in U(2)$ acts on the Pauli matrices as

$$\sigma_a \mapsto V\sigma_a V^\dagger = o_{xa}\sigma_x + o_{ya}\sigma_y + o_{za}\sigma_z,$$

for each $a \in \{x, y, z\}$ and where each coefficient o_{ab} is real. V can therefore be associated with a matrix

$$M_V = \begin{bmatrix} o_{xx} & o_{xy} & o_{xz} \\ o_{yx} & o_{yy} & o_{yz} \\ o_{zx} & o_{zy} & o_{zz} \end{bmatrix} \tag{6.18}$$

with pairwise orthogonal columns and unit determinant; i.e., M_V is in the **special orthogonal group** $SO(3)$. In the standard basis $\{|0\rangle, |1\rangle, |2\rangle\}$ of $\mathbb{R}^3$, the matrix

$$\sigma_x^{\otimes 2m} + (-1)^m\sigma_y^{\otimes 2m} + \sigma_z^{\otimes 2m} \tag{6.19}$$

is associated to the vector

$$v := |00\ldots0\rangle + (-1)^m|11\ldots1\rangle + |22\ldots2\rangle \in (\mathbb{R}^3)^{\otimes 2m} \tag{6.20}$$

acted on by $SO(3)^{\otimes 2m}$. We must show that if $O = O_1 \otimes \cdots \otimes O_{2m} \in SO(3)^{\otimes 2m}$ satisfies $Ov = v$, then each O_i is a monomial matrix (see [HP03]; a **monomial matrix** is the product of a permutation matrix and a diagonal matrix). Mapping O back to a unitary gate will show that $O_i \mapsto U_i \in C_2$.

Consider the operator

$$\langle 0|_1 \operatorname{Tr}_{\{3,4,\ldots,2m\}}(vv^T)|0\rangle_1, \tag{6.21}$$

acting on the second copy of $\mathbb{R}^3$. The matrix vv^T has 9 nonzero elements, and the partial trace over the last $2m-2$ copies of $\mathbb{R}^3$ gives

$$\operatorname{Tr}_{\{3,4,\ldots,2m\}}(vv^T) = |00\rangle\langle 00| + |11\rangle\langle 11| + |22\rangle\langle 22|. \tag{6.22}$$

Hence the matrix in Eq. 6.21 equals the rank one projector $|0\rangle\langle 0|$. Therefore, if $Ov = v$ then the operator

$$\langle 0|_1 \operatorname{Tr}_{\{3,4,\ldots,2m\}}(Ovv^TO^T)|0\rangle_1 \tag{6.23}$$

equals $|0\rangle\langle 0|$ as well. The operator is given by the matrix

$$O_2\langle 0|_1(O_1 \otimes I)\,\mathrm{Tr}_{\{3,4,\ldots,2m\}}(vv^T)(O_1^T \otimes I)|0\rangle_1 O_2^T \tag{6.24}$$

$$= O_2 \begin{pmatrix} (O_1)_{00}^2 & 0 & 0 \\ 0 & (O_1)_{01}^2 & 0 \\ 0 & 0 & (O_1)_{02}^2 \end{pmatrix} O_2^T. \tag{6.25}$$

where we have factored O_2 to the outside. The matrix within Eq. 6.25 equals the rank one projector $O_2^T|0\rangle\langle 0|O_2$ if and only if exactly one of the elements $(O_1)_{00}$, $(O_1)_{01}$, or $(O_1)_{02}$ is nonzero. Repeating the argument for every row of O_1 by considering the operators $\langle i|_1\,\mathrm{Tr}_{\{3,4,\ldots,2m\}}(Ovv^TO^T)|i\rangle_1$, $i \in \{0,1,2\}$, shows that every row of O_1 has exactly one nonzero entry. O_1 is nonsingular therefore O_1 is a monomial matrix. The vector v is symmetric so repeating the analogous argument for each operator O_i, $i \in [2m]$, completes the proof. □

A stabilizer code is **free of Bell pairs** if it cannot be written as a tensor product of a stabilizer code and a Bell pair. A stabilizer code is **free of trivial qubits** if for each $j \in [n]$, there is an element $s \in S$ such that $s_j \neq I$.

Theorem 20 (Local equivalences are semi-Clifford on coordinates in $\mathfrak{m}(S)$ [dNDM05]) *Let $C(S)$ and $C(S')$ be stabilizer codes that are free of Bell pairs and trivial qubits, and let $j \in \mathfrak{m}(S)$. If U is a local equivalence from $C(S)$ to $C(S')$ then U_j is semi-Clifford.*

Proof Consider $C(S)$. There is a least one element $M \in M(S)$ with $j \in \omega := \mathrm{supp}(M)$. Either $A_\omega = 1$ or $A_\omega = 3$ by Lemma 17.

If $A_\omega = 3$, then ρ_ω is LC equivalent to $\rho_{[[|\omega|,|\omega|-2,2]]}$. Moreover, as $C(S)$ is locally equivalent to $C(S')$, ω is also a minimal support of S' with $A_\omega(S') = 3$. Therefore, $\rho_\omega(S')$ is LC equivalent to $\rho_{[[|\omega|,|\omega|-2,2]]}$. By Lemma 16, U_ω maps $\rho_\omega(S)$ to $\rho_\omega(S')$ under conjugation. We must have $|\omega| > 2$; otherwise $C(S)$ is not free of Bell pairs. Since $|\omega|$ is even, $|\omega| \geq 4$, and by Lemma 19, $U_j \in C_2$.

If $A_\omega = 1$ and there are elements $R_1, R_2, R_3 \in M(S)$ such that $(R_1)_j = X$, $(R_2)_j = Y$, and $(R_3)_j = Z$, then there exists another minimal element $N \in M(S)$ such that $j \in \mu := \mathrm{supp}(N)$ and $M_j \neq N_j$. If $A_\mu = 3$ then we can apply the previous argument to conclude that $U_j \in C_2$.

Otherwise, $A_\mu = 1$ and

$$\rho_\omega = \frac{1}{2}(I^{\otimes|\omega|} + M_\omega) \tag{6.26}$$

$$\rho_\mu = \frac{1}{2}(I^{\otimes|\mu|} + N_\mu). \tag{6.27}$$

Since ω and μ are also minimal supports of S' with $A_\omega(S') = 1$ and $A_\mu(S') = 1$, there exist unique $M', N' \in S'$ such that

$$\rho_\omega(S') = \frac{1}{2}(I^{\otimes|\omega|} + M'_\omega) \tag{6.28}$$

$$\rho_\mu(S') = \frac{1}{2}(I^{\otimes|\mu|} + N'_\omega). \tag{6.29}$$

Applying Lemma 16 to U_μ and U_ν, we have

$$U_j M_j U_j^\dagger = \pm M'_j \tag{6.30}$$

$$U_j N_j U_j^\dagger = \pm N'_j \tag{6.31}$$

from Eqs. 6.26-6.29. These identities show that $U_j \in C_2$.

Finally, if $A_\omega = 1$ and $R = (R_1)_j = (R_2)_j$ for any $R_1, R_2 \in M(S)$, then *any* minimal support μ such that $j \in \mu$ satisfies $A_\mu(Q) = 1$. Applying Lemma 16 to U_μ, we have

$$U_j R U_j^\dagger = \pm R' \tag{6.32}$$

for some $R' \in \{X, Y, Z\}$. Therefore, U_j is semi-Clifford. □

Theorem 20 allows to characterize gates on qubits in $\mathfrak{m}(S)$. However, there are codes for which $\mathfrak{m}(S) \neq [n]$, so the theorem does not hold for all of the qubits. Indeed, consider a $[[6, 2, 2]]$ with stabilizer generators $XXXXII$, $ZZIIZZ$, $IIIIXX$, and $IIXXZZ$[1]. For $j = 3, 4$ there is no minimal support containing j.

This restriction on Theorem 20 limits its application to codes whose coordinates are fully covered by minimal supports. Therefore, the theorem is not a complete statement regarding the structure of single qubit transversal logic gates on stabilizer codes. However, the example suggests a new way to extend the theorem to all of the coordinates.

By introducing a new collection of minimal elements, it will be possible to extend Theorem 20

[1] I am indebted to Sergey Bravyi for this counterexample.

to all of the coordinates, proving a new result. Let $S_j := \{g \in S \mid j \in \text{supp}(g)\}$ and define a new collection of minimal elements,

$$M(S_j) := \{g \in S_j \mid \nexists g' \in S_j \text{ s.t. } \text{supp}(g') \subsetneq \text{supp}(g)\}. \tag{6.33}$$

These sets do not define codes because they need not be groups.

Lemma 21 (In $M(S_j)$ with equal supports $\Rightarrow$ same Pauli at coordinate j) *Suppose $j \notin \mathfrak{m}(S)$ and take $g, g' \in M(S_j)$. If $\text{supp}(g) = \text{supp}(g')$ then $g_j = g'_j$.*

Proof If there exist $R, R' \in M(S_j)$ such that $R|_j \neq R'|_j$ and $\text{supp}(R) = \text{supp}(R') = \omega$, then up to a local Clifford operation, we have $R = X^{\otimes|\omega|}$ and $R' = Z^{\otimes|\omega|}$. Without loss of generality, take $j = 1$. Since ω is minimal in S_j but not minimal in S, there exists an element F in $S \setminus S_j$ whose support $\text{supp}(F) = \omega'$ is strictly contained in ω; i.e., $\omega' \subsetneq \omega$. Since F is not in S_j, RF, $R'F$, $R'RF \in M(S_j)$. However, one of RF, $R'F$, $R'RF \in M(S_j)$ will have support that is strictly contained in ω, contradicting the fact that ω is a minimal support of S_j. $\square$

Theorem 22 (Local equivalences are semi-Clifford) *Let $C(S)$ and $C(S')$ be stabilizer codes that are free of Bell pairs and trivial qubits. If U is a local equivalence from $C(S)$ to $C(S')$, then U_j is semi-Clifford for all $j \in [n]$.*

Proof If $j \in \mathfrak{m}(S)$, we are done; so suppose $j \notin \mathfrak{m}(S)$. For any element $R \in M(S_j)$ with a fixed support ω, we have $R|_j = Z$ up to local Clifford operations by Lemma 21. Tracing over qubits in $\bar{\omega}$, we get

$$\rho_\omega = \frac{1}{2^{|\omega|}}(I_j \otimes R_I + Z_j \otimes R_Z), \tag{6.34}$$

where R_I and R_Z are linear operators acting on the other $\omega \setminus \{j\}$ qubits. Since $U_\omega \rho_\omega U_\omega^\dagger = \rho_\omega$, we have $U_j Z_j U_j^\dagger = \pm Z_j$, so U_j is semi-Clifford. $\square$

The following corollary about the elements of the automorphism group of a stabilizer code is immediate from Theorem 22. This corollary means that single block ($r = 1$) permutation transversal gates are essentially diagonal up to local Clifford gates.

Corollary 23 (Stabilizer code automorphisms are semi-Clifford) *Let $C(S)$ be a stabilizer* *code that is free of Bell pairs and trivial qubits. $U \in Aut\ S$ if and only if*

$$U = P_\pi L_1 \left(\bigotimes_{j=1}^{n} diag(1, e^{i\theta_j}) \right) L_2 \tag{6.35}$$

and U is a logic gate on S. Here, L_1 and L_2 are local Clifford gates, P_π is a product of swap gates enacting permutation π on the qubit coordinates, and $\{\theta_1, \ldots, \theta_n\}$ are angles.

After this work was completed, we learned that the statement was independently obtained by D. Gross and M. Van den Nest [GdN08] and that the theorem was first proved by different methods in the diploma thesis of D. Gross [Gro05]. Our result has an advantage that it follows from previously known results [dNDM05] by the application of a simple lemma, Lemma 21.

6.3.2 Gates between multiple encoded blocks

The results proven in the last section for a single block ($r = 1$) can be generalized to an arbitrary number of blocks. Since the generalization is relatively straightforward, the proof is sketched in this section in an even more informal way that the previous section. The new result is stated after the proof, at the end of the section.

The proof begins by noting that the subcode Lemma 16 carries over directly,

$$\mathrm{Tr}_{\bar{\omega}} \left[U P_Q^{\otimes r} U^\dagger \right] = U_\omega \mathrm{Tr}_{\bar{\omega}} \left[P_Q^{\otimes r} \right] U_\omega^\dagger = U_\omega \rho_\omega^{\otimes r} U_\omega^\dagger$$

where $U_\omega = \bigotimes_{i \in \omega} U_i$ is the restriction of U to ω, and ρ_ω is defined as before.

As we have seen in Section 6.3, the challenging behavior to characterize comes from non-Clifford gates. Therefore, we will find it convenient to more or less ignore Clifford gates in what follows; we move to locally Clifford equivalent stabilizer codes freely when studying particular minimal subcodes. Keeping this in mind, we can write the r block projectors when $A_\omega = 1$ and $A_\omega = 3$. If $A_\omega = 1$,

$$\rho_\omega^{\otimes r} \propto (I^\omega + Z^\omega)^{\otimes r} = \sum_{i \in \{0,1\}^r} (Z^\omega)^{i_1} \otimes \cdots \otimes (Z^\omega)^{i_r} = \sum_{i \in \{0,1\}^r} Z(i)^{\otimes |\omega|}$$

where i_j denotes the jth bit of i in the second expression, and $Z(i) = \otimes_{j=1}^r Z^{i_j}$ in the third expression. It may be helpful to consult Figure 6-2 for an illustration of one of the summands as it would look overlayed on Figure 6-1. In the right-most expression, the tensor product "$\otimes|\omega|$" is over the columns of Figure 6-2, because the transversal gate factors into a tensor product over columns.

Similarly, if $A_\omega = 3$, then

$$\begin{aligned}\rho_\omega^{\otimes r} &\propto (I^\omega + X^\omega + Z^\omega + (-1)^{|\omega|/2} Y^\omega)^{\otimes r} \\ &= \sum_{(a|b)\in\{0,1\}^{2r}} \left[(-1)^{|\omega|/2}\right]^{\mathrm{wt}(a\cdot b)} R^\omega(a_1,b_1)\otimes\cdots\otimes R^\omega(a_r,b_r) \\ &= \sum_{(a|b)\in\{0,1\}^{2r}} \left[(-1)^{|\omega|/2}\right]^{\mathrm{wt}(a\cdot b)} R(a,b)^{\otimes|\omega|},\end{aligned}$$

where $R(0,0) = I$, $R(0,1) = Z$, $R(1,0) = X$, and $R(1,1) = Y$, (i.e., $R(a_j,b_j) = i^{a_j\cdot b_j} X^{a_j} Z^{b_j}$) and also $R(a,b) = \otimes_{j=1}^r R(a_j,b_j)$. Again, the tensor product in the bottom-most expression is over columns rather than rows.

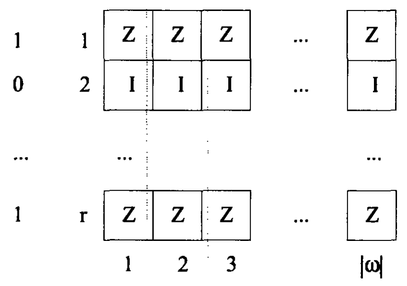


Figure 6-2: Illustration of a single term in the expansion of $\rho_\omega^{\otimes r}$ for the case $A_\omega = 1$. Each box is associated to a qubit, and the value of the bit to the left of each row determines whether that row is $Z^{|\omega|}$ or $I^{|\omega|}$. Therefore, the Pauli Z operator along each column is the same, and it is determined by the bit string. A factor U_j of a transversal gate acts on a column (the [blue] box with rounded edges, for example).

One or both of the projectors are left unchanged by transversal gates when the gates are restricted to a minimal support ω. Since $U_\omega I U_\omega^\dagger = I$, we can subtract the identity from each projector. Like before, the projectors can be viewed as vectors in Euclidean space acted on by rotations; see Lemma 19. This association again shows that rotations fixing these vectors have a special form. The r qubit gate U_j acts by conjugation on a nonidentity r qubit Pauli matrix R_s (s indexes the $4^r - 1$ nonidentity Paulis) as

$$U_j R_s U_j^\dagger = \sum_{R_t\in G_r - \{I\}} \alpha_{ts} R_t.$$

Here G_r denotes the r qubit Pauli group. The identity matrix does not appear on the right hand side because U_j is unitary and R_s is traceless, so the image must be traceless. The coefficients α_{ts} must be real because R_s is Hermitian. Furthermore, $\sum_{R_t\in G_r-\{I\}} \alpha_{ts_1}\alpha_{ts_2} = \delta_{s_1 s_2}$ because R_s

is unitary. So, we can represent U_j by a matrix O_j in $SO(4^r-1)$ whose real entries are α_{ts}, $s,t \in [4^r-1]$, and whose columns are orthonormal. The inverse unitary $U_j^\dagger$ is represented by the transpose O_j^T and its columns are orthonormal, so both the rows and columns are orthonormal. We can represent the nonidentity r qubit Pauli matrices can be represented by the canonical basis vectors $\{|1\rangle, |2\rangle, \dots, |4^r-1\rangle\}$ of $\mathbb{R}^{4^r-1}$. For concreteness, we can associate $|i\rangle$ to the binary representation of its label $(a|b) \in \{0,1\}^{2r}$ and to its representation as a Pauli $i^{\mathrm{wt}(a\cdot b)}X(a)Z(b)$. Continuing, we write the subcode projectors as vectors in $(\mathbb{R}^{4^r-1})^{\otimes|\omega|}$, using "$\mapsto$" to denote this mapping. For $A_\omega = 1$,

$$\rho_\omega^{\otimes r} - I \mapsto \sum_{i=1}^{2^r-1} \underbrace{|ii\dots i\rangle}_{|\omega| \text{ times}} =: w$$

and for $A_\omega = 3$,

$$\rho_\omega^{\otimes r} - I \mapsto \sum_{i=1}^{4^r-1} \alpha_i \underbrace{|ii\dots i\rangle}_{|\omega| \text{ times}} =: v,$$

where $\alpha_j \in \{\pm 1\}$. We can compute

$$ww^T = \sum_{i,j=1}^{2^r-1} |ii\dots i\rangle\langle jj\dots j|, \quad vv^T = \sum_{i,j=1}^{4^r-1} \alpha_i\alpha_j |ii\dots i\rangle\langle jj\dots j|.$$

Consider the following operators when $|\omega| \geq 3$,

$$\langle 1|_1 \operatorname{Tr}_{\{3,\dots,|\omega|\}} ww^T|1\rangle_1 = |1\rangle\langle 1|_2, \quad \langle 1|_1 \operatorname{Tr}_{\{3,\dots,|\omega|\}} vv^T|1\rangle_1 \propto |1\rangle\langle 1|_2,$$

The transversal gate, represented by a rotation O, fixes v or w or both ($Ov = v$ or $Ow = w$), so

$$\begin{aligned}
|1\rangle\langle 1|_2 &= \langle 1|_1 \operatorname{Tr}_{\{3,\dots,|\omega|\}} Oww^TO^T|1\rangle_1 \\
&= O_2\langle 1|_1 \sum_{i=1}^{2^r-1} (O_1 \otimes I)|ii\rangle\langle ii|(O_1^T \otimes I)|1\rangle_1 O_2^T \\
&= O_2 \left[\sum_{i=1}^{2^r-1} (O_1)_{1,i}^2 |i\rangle\langle i|_2\right] O_2^T
\end{aligned}$$

or

$$\begin{aligned}|1\rangle\langle 1|_2 &\propto \langle 1|_1 \operatorname{Tr}_{\{3,\ldots,|\omega|\}} Ovv^T O^T |1\rangle_1 \\ &= O_2 \langle 1|_1 \sum_{i=1}^{4^r-1} |\alpha_i|^2 (O_1 \otimes I)|ii\rangle\langle ii|(O_1^T \otimes I)|1\rangle_1 O_2^T \\ &= O_2 \left[\sum_{i=1}^{4^r-1} (O_1)^2_{1,i} |i\rangle\langle i|_2\right] O_2^T.\end{aligned}$$

In the case where O acts on v, "case v", we can conclude that the entire first row of O_1 has one nonzero entry, and the square of this real entry must be 1. Considering analogous operators, and understanding that O_j is nonsingular, we conclude that O_j is a monomial matrix for "case v", so the corresponding unitary must normalize the Pauli group; i.e., it must be Clifford.

In the case where O acts on w, "case w", the operator only has rank 1 if one of $(O_1)_{1,i}$ is nonzero and the rest are zero for $i \in [2^r - 1]$. However, the equation is only satisfied if the nonzero entry is ± 1 since O_2 is an orthogonal matrix. Therefore, considering analogous operators, O_j has a monomial subblock M for "case w", where $j \in \omega$ and ω is a minimal support, and the off diagonal subblocks are zero, i.e.,

$$O_1 = \begin{pmatrix} M & 0 \\ 0 & M' \end{pmatrix}, \tag{6.36}$$

where M is a monomial matrix whose nonzero entries are ± 1 and M' is in $SO(4^r - 2^r)$. Therefore, the corresponding unitary matrix must normalize the Z-type Pauli operators, i.e., the group $\{\pm R(0, b)\}$ where b runs over r-bit strings.

To summarize, if ω is a minimal support, $|\omega| \geq 4$, and $A_\omega = 3$, then U_j is an r qubit Clifford gate for $j \in \omega$. If $A_\omega = 1$, and $|\omega| \geq 3$, on the other hand, then up to local Clifford gates U_j is an r qubit unitary that normalizes Pauli Z operators but acts arbitrarily on Pauli X operators.

The case $A_\omega = 3$ and $|\omega| = 2$ is a special case. In this case, the minimal subcode is a $[[2, 0, 2]]$, which we know to be a Bell pair preserved by a continuum of local rotations $U \otimes U^*$, so it is an edge case that we must again discard.

The case $A_\omega = 1$ and $|\omega| = 1$ or $|\omega| = 2$ are special cases as well. In the first case, the qubit at the coordinate $j \in \omega$ is in a product state with the rest of the code. We can discard this case by insisting that $Q \neq Q' \otimes [[1, 1, 1]]$ is **free of trivial qubits**. The second case provides new behavior for $r > 1$ since we do not have enough qubits to "lock" the state to the diagonal by projecting onto the first qubit. Instead, U_j maps linear combinations of Pauli Z operators to linear combinations of

Pauli Z operators, $O_1 \left[\sum_{i=1}^{2^r-1} |i\rangle\langle i|\right] O_1^T = O_1 \left[\mathrm{Tr}_2(I \otimes O_2)ww^T(I \otimes O_2^T)\right] O_1^T = \mathrm{Tr}_2\, Ow = \mathrm{Tr}_2\, w = \sum_{i=1}^{2^r-1} |i\rangle\langle i|$. We denote the space of linear combinations of Pauli Z operators by $\mathit{span}\,\{\pm R(0,b)\}$.

Let U be a transversal r block gate and let $M(S)$ be the subgroup of S generated by minimal elements. Let $\mathfrak{m}(S)$ be the union of minimal supports. Fix a coordinate $j \in \mathfrak{m}(S) \subseteq [n]$ and choose a minimal support ω containing j. If $A_\omega = 3$, then $U_j \in C_2^{(r)}$. If $A_\omega = 1$ then there is one element N with support ω and there are three possibilities:

(i) $\exists$ minimal support ω' containing j with $A_{\omega'} = 3$, in which case $U_j \in C_2^{(r)}$.

(ii) $|\omega| \geq 3$ and $\exists N' \in S$ with minimal support μ containing j, $|\mu| \geq 3$, with $A_\mu = 1$ and $N_j \neq N'_j$. In this case, U_j normalizes the Pauli X operators and the Pauli Z operators, so $U_j \in C_2^{(r)}$.

(iii) There are no further minimal supports containing coordinate j or other minimal supports containing coordinate j have the same Pauli at that coordinate.

Case (iii) is the only case in which U_j might not be a Clifford gate. In this final case, U_j is $L_1 V L_2$ where L_1, L_2 are local Clifford gates (which we may have applied to put our minimal code into a standard form) and V is a gate that either:

(a) normalizes $\{\pm R(0,b)\}$

(b) keeps $\mathit{span}\,\{\pm R(0,b)\}$ invariant.

Case (a) occurs if there is a minimal support whose size is greater than 2; otherwise, case (b) occurs. Finally, appealing to Lemma 21, if $j \notin \mathfrak{m}(S)$ then case (iii,b) occurs. Case (a) shows that U_j is semi-Clifford and case (b) shows that U_j is generalized semi-Clifford.

The main result of this section that has now been proven is:

Theorem 24 (r-block transversal gates are generalized semiClifford) *If U is a transversal gate on a stabilizer code $C(S)$ that is free of Bell pairs and trivial qubits, then U_j is an r-qubit generalized semi-Clifford gate for all $j \in [n]$. In addition, if $S = M(S)$, then U_j is an r-qubit Clifford gate for all $j \in [n]$.*

The details of the proof give us even more information about each U_j, allowing us to determine, for a given stabilizer code, whether U_j must be Clifford, semi-Clifford, or generalized semi-Clifford. Since the determination is based entirely on the minimal supports and their associated values of A_ω, there is an obvious way to implement an algorithm to make this determination.

6.3.3 Transversal gates are not a universal set for any encoded qubit

We would like to know if a stabilizer code exists that has transversal gates that are computationally universal on at least one of the encoded qubits. If so, then this means that it is possible to approximate any single qubit logical gate on one of the k encoded qubits (we don't care which one) to any accuracy using only transversal gates. Since $\mathrm{Trans}_r(S)$ is a group, $\mathrm{Trans}_r(S)$ is computationally universal on $C(S)$ if given any r-block logical gate U, $\forall\epsilon > 0$, $\exists V_\epsilon \in \mathrm{Trans}_r(S)$ such that $\|UP_{C(S)}^{\otimes r} - V_\epsilon P_{C(S)}^{\otimes r}\| < \epsilon$. We will assume this statement is true only for single qubit logical gates on a particular encoded qubit and derive a contradiction, which will imply that $\mathrm{Trans}_r(S)$ is not computationally universal on $C(S)$.

Transversal gates may not conjugate Paulis to Paulis, perhaps even if the transversal gate approximates a logical Clifford gate. Such gates have the potential to take us beyond the stabilizer formalism, forcing us to deal with foreign objects such as the subgroup of local gates in the generalized stabilizer. This subgroup is difficult to grasp, and its members are an underlying reason why the LU-LC conjecture is false. Fortunately, it is possible to remain within the powerful, familiar stabilizer formalism, as we now see.

No basis has been introduced for $C(S)$ yet, so the discussion to this point applies to both subsystem and subspace codes. However, as we proceed, we should take care so that our arguments continue to hold for subsystem codes. The Hilbert space of n qubits partitions under a subsystem code as $\mathcal{H} = \oplus_{\mathbf{s}}\mathcal{H}_L \otimes \mathcal{H}_G$, where the direct sum is over error syndromes, $\mathcal{H}_L$ is the protected space, and $\mathcal{H}_G$ is the gauge space. Partition the logical Pauli operations that generate $Z(S)/S$ into two sets, the set of operations on qubits encoded in the protected space and the set of operations on qubits encoded into the gauge space. Let α be a *minimum weight element* of the union of cosets $\bar{X}_p^{(1)}S \cup \bar{Z}_p^{(1)}S \cup \bar{Y}_pS$, where p is a protected logical qubit and "(1)" denotes the first block. Let $\omega := \mathrm{supp}(\alpha)$. Without loss of generality, we can suppose $\alpha \in \bar{X}_p^{(1)}$. The notation $\bar{X}_p^{(1)}S$ means the set of representatives of $\bar{X}_p^{(1)}$ in the Pauli group. Any operator on the gauge qubits in the first block can be applied when choosing a representation α, but in doing so, it is not possible to construct a logical operator on a protected qubit that has weight less than d. Likewise, how we represent identity on blocks other than the first does not matter, since we must transform all representations correctly. We choose to represent it by tensor products of identity operators.

Assume that $\bar{H}_p^{(1)}$ is approximated arbirarily well by elements of $\mathrm{Trans}_r(S)$. $\bar{H}_p^{(1)}$ applies a logical identity gate on the protected logical qubits of blocks other than the first, but again any logical operation can be applied to the gauge qubits in those blocks. Applying $\bar{H}_p^{(1)} \in \text{Trans}_r(S)$ to $\alpha \otimes I$, we get $\beta'' := \bar{H}_p(\alpha \otimes I)\bar{H}_p^\dagger$. The operator β'' must be arbitrarily close to $\bar{Z}_p^{(1)}$ up to elements of the transversal identity and gauge operators. Expand β'' in the basis of Pauli operators,

$$\beta'' = \sum_{R \in G_n^{\otimes r}} \alpha_R R = \sum_{R \in Z(S)^{\otimes r}} \alpha_R R + \sum_{R \in G_n^{\otimes r} - Z(S)^{\otimes r}} \alpha_R R.$$

The operators not in $Z(S)^{\otimes r}$ map the code space to an orthogonal subspace, so there must be terms in the expansion that are in $Z(S)^{\otimes r}$. Let $\beta' := P_{C(S)}\beta'' P_{C(S)}$. Terms in $S^{\otimes r}$ can be neglected because they act trivially. Therefore, there must be an element of $Z(S)^{\otimes r}$ that represents $\bar{Z}_p^{(1)}$ and does an arbitrary logical Pauli operation to the gauge qubits. The transversal gate cannot cause β'' to have support on the first block that strictly contains ω, nor can it have support strictly contained in ω, since $|\omega|$ is minimal. Furthermore, $I \in Z(S)$ so we can ignore blocks other than the first and find an operator $\beta \in Z(S) \setminus S$ that represents $\bar{Z}_p^{(1)}$ and does an arbitrary logical Pauli operation to the gauge qubits in the first block; see Figure 6-3. We also have $\omega = \text{supp}(\alpha) = \text{supp}(\beta)$, $|\omega| = d$. Repeating the argument for $\bar{K}_p^{(1)}$, we obtain an operator γ with support ω that represents $\bar{Y}_p^{(1)}$ up to logical Paulis on the gauge qubits.

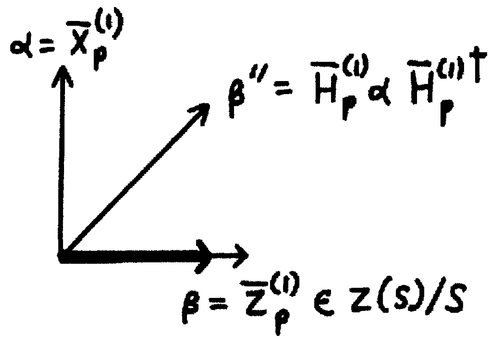

$\alpha = \bar{X}_p^{(1)}$
$\beta'' = \bar{H}_p^{(1)} \alpha \bar{H}_p^{(1)\dagger}$
$\beta = \bar{Z}_p^{(1)} \in Z(S)/S$

Figure 6-3: This figure illustrates the intuition for finding logical Pauli operators on the protected qubit. The main idea is that β'' must have a component along a representative of $\bar{Z}_p^{(1)}$ in the centralizer; otherwise $\bar{H}_p^{(1)}$ does not have the correct action on the code space.

Now we can derive the contradiction. Since we have assumed that the transversal gates are a universal set for some protected qubit p, there must be some coordinate $j \in \omega$ such that U_j is not Clifford. Otherwise, we could not apply any non-Clifford logical gates to encoded qubit p by Theorem 24. The essence of the contradiction is this: the non-Clifford U_j in, say, the tensor product decomposition of $\bar{H}_p^{(1)}$, must fix one of the Pauli operators at coordinate j in the first block, say $U_jZ_1U_j^\dagger = \pm Z_1$ (or $U_jZ_1U_j = I_1$ could happen too, if U_j is non-Clifford, but this is leads to the same contradiction). Therefore, a product of one of the images of α, β, or γ under $\bar{H}_p^{(1)}$ and another logical Pauli operator α, β, or γ will have support strictly contained in ω, but will represent a logical Pauli on the protected qubit as well. This is impossible because α, β, and γ already have minimum weight! The only assumption we have made is that the set of transversal gates is universal for the arbitrarily chosen protected qubit p, so this must not be true, and we obtain:

Theorem 25 (Transversal gates are not computationally universal) *The group of transversal gates $\mathit{Trans}_r(S)$ is not computationally universal on r copies of a stabilizer code $C(S)^{\otimes r}$ for any $r \geq 1$.*

6.3.4 Full automorphisms are not a universal set for any encoded qubit

The group of gates $\mathrm{Aut}(S^{\otimes r})$ is also not computationally universal for any encoded qubit. This automorphism group does not include gates that interact qubits in different blocks, but it does allow permutations of qubits between blocks. Since we can regard $C(S)^{\otimes r}$ as "just another code" with stabilizer $S' := S^{\otimes r}$, it is enough to demonstrate the result for a single block encoded into the code $C(S')$. An element of this group is illustrated for $r = 1$ in Figure 6-4.

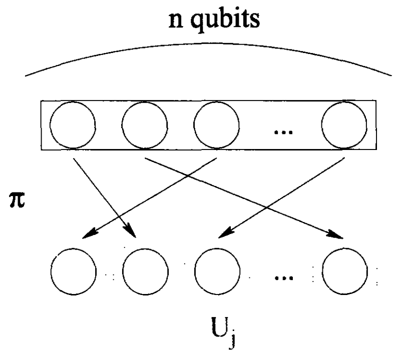


Figure 6-4: Illustration of a code automorphism on 1 block of n qubits. The block is represented by a collection of circles (qubits), grouped into a box. The block undergoes a coordinate permutation π followed by a local unitary gate U whose unitaries U_j act on qubits in the [blue] boxes with rounded edges. U can also be conjugated by P_π to view the automorphism as $P_\pi U'$ where U' is another local unitary gate.

As before, and with the same caveats for subsystem codes, let α be a minimum weight element of $\bar{X}_p^{(1)}S' \cup \bar{Y}_p^{(1)}S' \cup \bar{Z}_p^{(1)}S'$. Let α represent $\bar{X}_p$ without loss of generality and let $\omega := \mathrm{supp}(\alpha)$.

Consider the single qubit gate A defined by

$$X \mapsto \frac{1}{\sqrt{3}}(X+Y+Z),$$
$$Z \mapsto Z',$$

where $\{AXA^\dagger, AZA^\dagger\} = 0$.

As before, assume that $\bar{A}_p$ is implemented approximately, to accuracy ϵ, by some gate $UP_\pi \in \mathrm{Aut}(S')$. Then $\eta := \bar{A}_p \alpha \bar{A}_p^\dagger$ is an element of $\frac{1}{\sqrt{3}}(\bar{X}+\bar{Y}+\bar{Z})\bar{I}$, where $\bar{I}$ is in the generalized stabilizer. Expanding $P_{C(S')}\eta P_{C(S')}$ in the Pauli basis, we again see that there must be representatives α', β', and γ' of $\bar{X}_p$, $\bar{Z}_p$, and $\bar{Y}_p$ in the centralizer that all have support ω', $|\omega'| = |\omega|$. The important fact is that they must have the same support, despite the permutation we applied!

U must be a local equivalence between $C(S'')$ and $C(S')$, where $S'' = P_\pi S P_\pi^\dagger$, so each U_j is either a single qubit Clifford gate or semi-Clifford gate. If all U_j are Clifford, then we are done. Otherwise, one or more gates are semi-Clifford. We can assume that j is in $\omega'' := P_\pi \omega'$ (otherwise $\bar{A}_p$ is Clifford, which is not true). Let δ' be another name for the Pauli operator in $\{\alpha', \beta', \gamma'\}$ whose jth coordinate does not change when we apply $\bar{A}_p$. So $\eta' := \bar{A}_p \delta' \bar{A}_p^\dagger$ yields three new Pauli operators with support ω'', but at least two must have the same Pauli at coordinate j, so their product's support is strictly contained in ω'', a contradiction. Therefore the gate $\bar{A}_p$ cannot be implemented with arbitrary accuracy by a product of gates in $\mathrm{Aut}(S')$. We have proven:

Theorem 26 (Quantum code automorphisms are not computationally universal) *The group of gates $Aut(S^{\otimes r})$ is not computationally universal on r copies of a stabilizer code $C(S)$ for any $r \geq 1$.*

6.4 Non-Clifford single qubit transversal gates for CSS codes

6.4.1 Transversality conditions for diagonal gates

Up to local Clifford equivalence, Corollary 23 says that the unitary part of a code automorphism is a diagonal gate. Therefore, without much loss of generality, we may restrict our discussion of the non-Clifford elements of $\mathrm{Aut}(S)$ to diagonal gates. We could consider the diagonal automorphisms for all locally Clifford equivalent codes, and their permutation equivalent codes, to find all of the non-Clifford automorphisms. For simplicity, and because this case has the greatest known utility for fault-tolerance, we further restrict to the case where the code is CSS.

Lemma 27 *Let $C(S)$ be a CSS code $CSS(C_1, C_2)$ constructed from classical binary codes $C_2^\perp < C_1$. Then*

$$V = \bigotimes_{\ell=1}^{n} diag(1, e^{i\theta_\ell}) \in Aut(S) \tag{6.37}$$

iff

$$\forall a \in C_1/C_2^\perp, \forall c, c' \in C_2^\perp, \sum_{\ell\in\mathrm{supp}(a+c)} \theta_\ell = \sum_{\ell\in\mathrm{supp}(a+c')} \theta_\ell \ mod\ 2\pi. \tag{6.38}$$

Proof The states

$$|\tilde{a}\rangle \propto \sum_{c\in C_2^\perp} |a+c\rangle, a \in C_1/C_2^\perp, \tag{6.39}$$

are a basis for $C(S)$. V is diagonal, so $V|c\rangle = v(c)|c\rangle$ for $c \in C_1$. The factor $v(c) \in \mathbb{C}$ is $e^{i\sum_{j\in\mathrm{supp}(c)}\theta_j}$. V implements a logical gate so $V|\tilde{a}\rangle \in C(S)$ for all $a \in C_1/C_2^\perp$, which is possible for a diagonal gate iff $v(a+c) = v(a+c')$ for all $a \in C_1/C_2^\perp$ and all $c, c' \in C_2^\perp$. □

Rather than solve the system of equations implied by the lemma, we now restrict to angles $\theta_\ell = \theta$ are all equal.

Corollary 28 *Let $C(S)$ be a CSS code constructed from classical binary codes $C_2^\perp < C_1$. A gate $V \in Aut(S)$ is a tensor product of n diagonal unitaries $V_\theta = diag(1, e^{i\theta})$ iff*

$$\forall a \in C_1/C_2^\perp, \forall c, c' \in C_2^\perp, \phi(\mathrm{wt}\,(a+c) - \mathrm{wt}\,(a+c')) \in \mathbb{Z}, \tag{6.40}$$

where $\mathrm{wt}\,(c)$ *denotes the Hamming weight of a classical binary codeword and* $\phi := \frac{\theta}{2\pi}$ *mod* $[0,1)$.

If each coset $C_1/C_2^\perp$ has constant weight, then the condition of the corollary is true for any ϕ. If not, then the condition can only be satisfied for rational angles $\phi \in [0,1) \cap \mathbb{Q}$. Suppose $\phi = \frac{p}{q}$ in lowest terms. The condition becomes

$$\forall a \in C_1/C_2^\perp, \forall c, c' \in C_2^\perp, \mathrm{wt}\,(a+c) \bmod q = \mathrm{wt}\,(a+c') \bmod q. \tag{6.41}$$

In words, the cosets $a + C_2^\perp$ of $C_2^\perp$ in C_1 must be constant weight $w(a)$ modulo q.

Since C_1 and $C_2^\perp$ are linear codes, both contain the all-zeros codeword, so the coset $0 + C_2^\perp$ must have weight 0 modulo q. Therefore, the condition is satisfied only if the codewords of $C_2^\perp$ are divisible by a common divisor q. A classical linear code is said to be **divisible** by Δ if Δ divides the weight of each codeword. A classical linear code is divisible if it has a divisor larger than 1. An $[n, k]$ classical code can be viewed as a pair (V, Λ) where V is a k-dimensional binary vector space and $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ is a multiset of n members of the dual space V^* that serve to encode $v \in V$ as $c = (\lambda_1(v), \dots, \lambda_n(v))$ and the image of V in $\{0,1\}^n$ is k-dimensional. The **b-fold replication** of C is $(V, r\Lambda)$ where $r\Lambda$ is the multiset in which each member of Λ appears r times.

The following theorem, a special case of [War99], *suggests* that implementation angles producing nontrivial logical gates may only be $\theta = \frac{2\pi p}{2^k}$. The resulting local unitary must implement a logic gate that rotates $|\bar{a}\rangle$ by $e^{i\gamma_a}$ where $\gamma_a = (e^{i\theta})^{f(a)}$, where $f(a)$ is an integer that may depend on a. This suggests that the logic gate is in C_k for some k.

Theorem 29 ([War99]) *Let C be an $[n, k]$ classical binary code that is divisible by Δ, and let $b = \Delta/gcd(\Delta, 2^{k-1})$. Then C is equivalent to a b-fold replicated code, possibly with some added 0-coordinates.*

These observations suggest the following conjecture:

Conjecture 30 (Transversality conjecture) *For any $[[n, k, d]]$ stabilizer code S, each $U \in Trans_1(S)$ implements a logical gate $\bar{V} \in C_m^{(k)}$ for some $m \geq 1$.*

6.4.2 A family of CSS codes with non-Clifford transversal gates

The Reed-Muller codes are well-known examples of divisible codes. Furthermore, they are nested and their dual codes are also Reed-Muller codes, which makes them amenable to the CSS construction. In particular,

Theorem 31 (1.10.1, [HP03]) *Let $RM(r, m)$ be the rth order Reed-Muller code with block size $n = 2^m$ and $0 \leq r \leq m$. Then*

(i) $RM(i, m) \subseteq RM(j, m)$, $0 \leq i \leq j \leq m$

(ii) $\dim RM(r, m) = \sum_{i=0}^{r} \binom{m}{i}$

(iii) $d = 2^{m-r}$

(iv) $RM(m, m)^\perp = \{0\}$ *and if* $0 \leq r < m$ *then* $RM(r, m)^\perp = RM(m - r - 1, m)$.

(v) $RM(r, m)$ *is divisible by* $\Delta = 2^{\lfloor m/r \rfloor - 1}$.

Corollary 32 *Let $even(RM^*(r,m)) = C_2^\perp < C_1 = RM^*(r,m)$ where $0 < r \leq \lfloor m/2 \rfloor$. Then $CSS(C_1, C_2)$ is an $[[n = 2^m - 1, 1, d = min(2^{m-r} - 1, 2^{r+1} - 1)]]$ code with a transversal gate $G = \otimes_{j=1}^{n} diag(1, e^{i2\pi/\Delta})$ enacting $\bar{G} = diag(1, e^{-i2\pi/\Delta}) \in \mathcal{C}^{(1)}_{\log_2 \Delta}$ where $\Delta = 2^{\lfloor m/r \rfloor - 1}$.*

For instance, the $[[2^m - 1, 1, 3]]$ CSS codes constructed from $R^*(1, m)$ and its even subcode have the transversal logic gate $\exp(-i\frac{\pi}{2^{m-1}}\bar{Z})$ [ZCCC07, SI05]. The smallest of these has been applied in magic state distillation schemes [BK05] and measurement-based fault-tolerance schemes [RHG07]. The parameters $m = 8$ and $r = 2$ give a $[[255, 1, 7]]$ code with transversal T, but this is not as good as the concatenated $[[15, 1, 3]]$ code. There is a possibility that other families of classical divisible codes may give better CSS codes with $d > 3$ or, particularly, $k > 1$ and transversal non-Clifford gates.

6.5 Conclusion

We have studied what it means to compute on information encoded into a stabilizer code and proven several results about transversal gates. Our technique is based on a subcode method for stabilizer codes that was developed by Rains. We have successfully applied the subcode method to obtain results about the structure of the full automorphism group and the group of transversal gates. Using these results, we have further proven that transversal gates are not a universal set and full automorphisms are not a universal set. Finally, we have explored non-Clifford transversal gates for CSS codes, presenting new examples and proposing an interesting conjecture for future work.

The main open question is how much "transversality" can be strengthened before it becomes universal. In further work beyond the work in this dissertation, we have shown that transversal gates are not universal for codes on higher dimensional systems as well [CCC+08]. However, the case of permutation transversal gates is still open. It would be very interesting if those gates were universal since the architecture of a fault-tolerant quantum computer could become simpler. Allowing input and output to be in different codes certainly gives universality (see Chapter 8), but we have the added need to transform between input and output codes.

We have examples of transversal logical gates in C_k for all k, and these codes can be constructed for any distance d using concantenation. Is there a code with a logical gate that is not in C_k? Are there quantum codes that encode $k > 1$ qubit and have transversal non-Clifford gates? Perhaps codes with $k > 1$ could be chosen based on the frequency of non-Clifford gates in subroutines of a quantum algorithm. For example, we could endeavor to find a code that makes the most commonly occuring gates in an algorithm transversal and implement the rare gates using quantum software methods.

Finally, we wonder how well these techniques apply to nonadditive codes such as CWS codes. Perhaps there is a counterexample for nonadditive codes – such an example would be very interesting (and strange).

Part III

Systems

Chapter 7

Fault-tolerant code architectures

7.1 Introduction

Properties of quantum error-correcting codes are crucial for fault-tolerant quantum computation as has been demonstrated by Chapters 5 and 6. The code parameters have a direct role in determining the accuracy threshold. The type of code together with its parameters roughly indicate the size of the error-correction circuit. Finally, the automorphism group and transversal gate set determine what gates are "easy" to implement fault-tolerantly.

In this chapter, fault-tolerant *systems* are built using several quantum codes. The codes are combined by concatenation and recursive simulation. They are specialized depending on how "close" they are to the device level. Our point of view differs from other code architecture work in that we view concatenation as an essential concept in our vision of an effective architecture. This view is motivated by the idea that maximizing the threshold appears to be of the highest importance, since physical error rates are expected to be relatively high.

Therefore, an important question we address in this chapter is how to evaluate those codes rigorously and compare them fairly without having to build and/or simulate the entire complex system at once. The evaluation method is rooted in a rigorous theory and produces numerical Monte-Carlo estimates of thresholds and pseudothresholds that are closely related to rigorous bounds.

The chapter is organized as follows. Section 7.2 defines an effective code architecture and discusses how codes determine parameters of this architecture. Section 7.3 reviews rectangle design properties, error-correction circuits, and a technique for constructing universal gate sets for any CSS code. Finally, Section 7.4 introduces two Monte-Carlo adaptations of what we call the AGP method [AGP06] for evaluating codes and code architectures. Section 7.5 concludes.

7.2 Code architectures using hierarchies of specialized codes

This section introduces a definition of an effective code architecture and explains how the quantities in this definition are influenced by various code parameters.

7.2.1 Effective code architectures

There are essentially two known approaches to fault-tolerant quantum computation using quantum codes in the circuit model. The first approach uses concatenated codes and was introduced in Chapter 5. The standard approach is of this kind. The other approach is a non-standard approach that uses surfaces codes, which are analogous to classical repetition codes, and for which concatenation is not necessary for a threshold to exist.

The practical goal of both approaches is the same – to obtain an effective code architecture for large scale quantum computation. By **effective code architecture**, we mean a code or hierarchy of codes such that (a) the logical error rates of the fault-tolerant system are comparable to modern digital computers, (b) the noise threshold (or pseudothreshold [SCCA06]) is acceptably high, and (c) the first two goals are achieved with a minimum of overhead as measured by the size and depth blow up of the fault-tolerant circuit relative to the original circuit. There is a tradeoff one can expect between overhead and the logical error rate that mimics the trade-off between distance and rate of quantum codes.

An effective code architecture is very likely to use a concatenated hierarchy of specialized codes, since error levels of physical implementations are expected to be high, optimistically in the range from $O(10^{-2})$ to $O(10^{-6})$. Therefore, it is clear that optimizing the threshold has priority over optimizing overhead. Specific knowledge of the noise process affecting devices may be used to select bottom level, physical, inner codes. Indeed, this can boost the threshold above other thresholds for inner codes designed for general noise [AP07]. After one level of coding, the noise model is expected to be like depolarizing noise, and another high threshold inner code can be used to further reduce the logical error rate. Once the noise has been significantly reduced by one (or at most two) high threshold inner codes, a more efficient outer code can achieve the final logical error rate while limiting further increase in overhead [Ste03]. Furthermore, the outer code may be chosen so that its fault-tolerant gate set is tailored to the specifics of the quantum algorithm; i.e., perhaps having a non-Clifford transversal gate would be beneficial during some parts of the computation.

The envisioned hierarchy of concatenated codes is illustrated in Figure 7-1. One level above the physical level and its code, we use an inner code C_{inner} that is chosen to have a high noise threshold and a reasonable overhead. We will pick some illustrative numbers to argue how one can envision completing the code architecture. We will see in Chapter 8 that one can find an inner code that maps a base error rate of $p_0 = O(10^{-4})$ onto a logical error rate of $p_1 = O(10^{-7})$. To run a reasonable-sized factoring algorithm one may need an logical error rate of, say, $O(10^{-15})$ [1]. Thus one needs an outer code C_{outer} that brings the error rate from $O(10^{-7})$ to $O(10^{-15})$. The desirable features of the outer code are roughly as follows. The top code is a block code $[[n, k, d]]$ with good rate k/n in order to minimize the overhead. The improvement in error rate for a code which can correct t errors is roughly

$$p_1 \approx p_{th} \left(\frac{p_0}{p_{\text{th}}}\right)^t, \tag{7.1}$$

where p_0 is the unencoded error rate and p_{th} is the threshold error rate. Thus in order to get from $p_0 = O(10^{-7})$ to $p_1 = O(10^{-15})$ we could use a code which can correct 5 errors and has a threshold of $O(10^{-5})$. In [Ste99a] Steane studied several block-codes which may meet these demands. Polynomial codes would be an interesting family to study in this respect. Low rate polynomial codes are discussed in Chapter 8.

7.2.2 Approximate threshold scaling with code parameters

We now discuss the global behavior of the noise threshold as a function of block size n, distance, and other code properties. To first approximation the threshold is determined by the equation

$$p_{th} = N p_{th}^{t+1} \implies p_{th} = N^{-1/t}, \tag{7.2}$$

where t is the number of errors that the code can correct and N is the combinatorial factor counting the sets of $t+1$ locations in an encoded gate that lead to the encoded gate failing. Let us consider Eq. (7.2) and see how we can get the best possible threshold. An upper-bound on N is $\binom{A}{t+1}$ where A is the total number of locations in the encoded gate (rectangle). Ideally, a code or code family has a distance that is linear in n; i.e., t is linear in n. Let us assume for simplicity that only some fraction of all locations appears in the malignant fault sets of size $t+1$; i.e., we model $N \approx \binom{A_{\text{mal}}}{t+1}$ where $A_{\text{mal}} < A$. The locations in A_{mal} are in some sense the weak spots in the circuits; overall failure is most sensitive to failure at these locations. A_{mal} may be either linear or super-linear in

[1] An n-bit number can be factored using a circuit with space-time complexity of no more than roughly $360n^4$ [BCDP96], so RSA-1024 could be broken using a circuit with $O(10^{15})$ potential fault locations. Using different architectures, it may be possible to reduce this to $O(10^{11})$ or less; see for example [Met06].

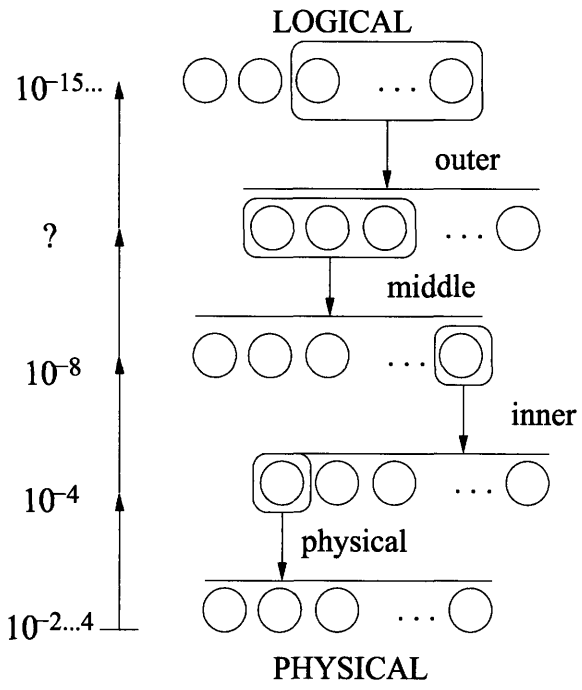


Figure 7-1: A schematic of a concatenated hierarchy of specialized codes. Although this figure shows many levels of coding, we have argued that a few levels of coding are sufficient to solve problems beyond the capabilities of classical computers.

the block size n. In case A_{mal} scales linearly in n, *and* $t = \delta n$ for some $\delta \leq 1/4$, the threshold in Eq. (7.2) *increases* as a function of n and asymptotes in the limit of large n to a finite value. For $A_{\text{mal}} = \alpha n$ and $\delta \ll \alpha$ (which is typically the case since $t \leq n/4$ by the Singleton bound) we get, using Sterling's approximation,

$$p_{\text{th}} = \lim_{n\to\infty} \left(\frac{\alpha n}{\delta n + 1}\right)^{-1/(\delta n)} = \frac{\delta}{e\alpha} + O\left(\frac{\delta^2}{\alpha^2}\right) \tag{7.3}$$

It is also clear that when t is constant, for any polynomial $A_{\text{mal}} = \text{poly}(n)$, the threshold p_{th} in Eq. (7.2) decreases as a function of n. When A_{mal} scales super-linearly with n and t is linear in n we get the following behavior. First, the threshold increases with n (the effect of larger t), then the threshold declines since the effect of a super-linear A_{mal} starts to dominate. For codes and EC circuits with this behavior, it is thus of interest to determine where this peak threshold performance occurs. Figure 7-2 is an illustration of this behavior. We will see this peak for a large set of codes in Figure 8-18 of Chapter 8.

Now let us consider the scaling of A (and A_{mal}) in case we use Steane-EC. In Section 7.3.2 we review how we can bound A for a CSS code with Steane error correction, but a rough estimate is

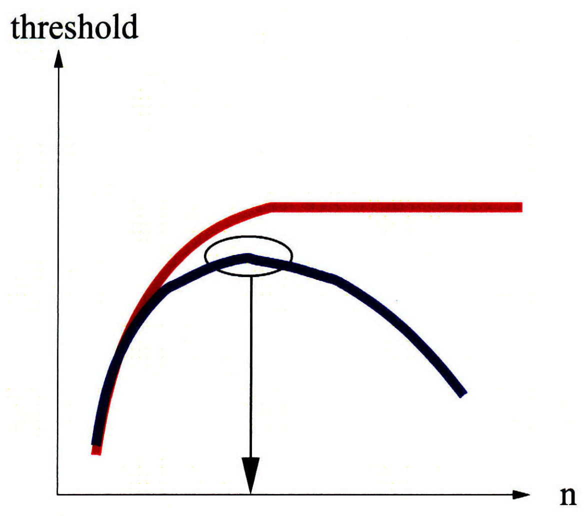


Figure 7-2: Schematic of the behavior of the threshold with parameters of a quantum code. The threshold is plotted on the vertical axis and the block size of the code is plotted on the horizontal axis. The top curve [red] is for a code with distance that scales linearly with block size. The bottom curve [blue] is for a code whose distance scales sublinearly with block size.

that

$$A = c_1 A_{\text{enc}} + c_2 A_{\text{ver}} + c_3 n. \tag{7.4}$$

Here A_{enc} is the number of locations in the encoding of the ancillas for error correction, and A_{ver} is the number of locations in the verification of the ancillas for error correction. The additional term linear in n comes from the transversal encoded gates and the transversal syndrome extractions. For a CSS code and the standard encoding construction (see Section 7.3.2), A_{enc} typically scales as $O(wn)$ where w is the maximum Hamming weight of the rows of the generator matrix of either C_1 or $C_2^{\perp}$ *in standard form.* However this standard construction may be sub-optimal, since by bringing the generator matrix in standard form one can increase the maximum weight of its rows.

For Steane-EC the full verification of the ancilla block requires other ancilla blocks; a fully fault-tolerant verification would give a pessimistic scaling of $A_{\text{ver}} = O(wnt)$. However it is not necessarily desirable to have strict fault-tolerance as long as the total probability of low-weight faults that produce errors with weight $t+1$ or more is low, see the discussion in Section 7.4.1. On the other hand for increasing n the number of verification rounds should at least be increasing with n, perhaps $O(\log n)$ would be sufficient. If we assume that A_{mal} scales similarly as A, it follows that if we look for linear-scaling A_{mal} we need to look at code families which have simple encoders, scaling linearly in n. This seems only possible for stabilizer codes with constant weight stabilizers, such as quantum LDPC codes [MMM04] and surface codes or for the Bacon-Shor codes (which have encoders that use $O(n)$ 2-qubit gates).

For the Bacon-Shor and surface codes the distance t does not scale linearly with n but as $\sqrt{n}$. Nonetheless, the work in [DKLP02] shows that the effective distance for the surface codes does scale linearly in the block size, since there are very few uncorrectable errors of weight $O(t)$. For the Bacon-Shor code family, where one has less syndrome information, this behavior has not been observed; see [AC07] and Chapter 8.

For code families with constant-weight stabilizers an interesting alternative to Steane-EC [Ste97] is the use of Shor-EC [Sho96] where the syndrome corresponding to each stabilizer is extracted using a cat state or simple unencoded qubit ancillas. As for ancilla verification in Steane-EC, the syndrome extraction needs to be repeated to make the circuits more fault-tolerant. It is striking that the surface codes with Shor-EC are the only known examples of a code family with a finite $n \to \infty$ threshold. This is despite the $O(n\sqrt{n})$ scaling of the total number of locations A of the Shor error correction circuit, although the syndrome extraction circuit itself has constant depth and $O(n)$ locations.

7.3 Rectangle constructions

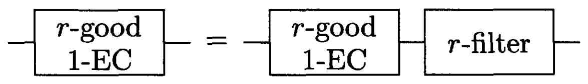
This section reviews rectangle design properties in 7.3.1 and analyzes Steane-EC to support the scaling arguments of Section 7.2.2. We mainly consider CSS codes but we do not assume that the codes are dual containing or doubly even. In 7.3.3, we review how universal gates can be constructed for any CSS code, showing why it is adequate to compute error rates and thresholds for a CNOT rectangle.

7.3.1 Design rules for fault-tolerant rectangles using error-correcting codes

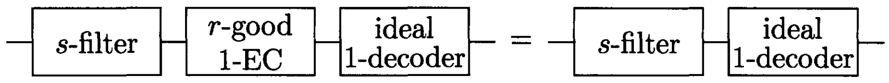
There are some syntactic properties of gates and error-correction circuits that are used in beautiful proofs of the threshold theorem. These properties can be taken as general design rules for fault-tolerant rectangles [AGP06].

An s-**filter** is a orthogonal projection on to the space spanned by all states that can be obtained by acting on a codeword with a Pauli operator of weight no larger than s. A circuit is r-**good** if it contains no more than r faults. The properties are as follows.

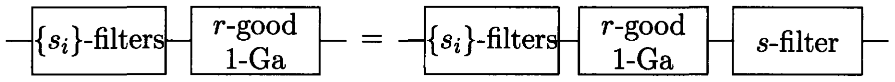
Property 0 ($r \leq t$):



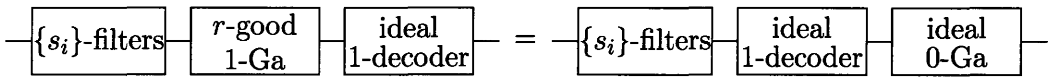
Property 1 ($r + s \leq t$):



Property 2 ($s = r + \sum_i s_i \leq t$):



Property 3 ($r + \sum_i s_i \leq t$):



In properties 2 and 3, the s_i-filter is applied to the ith input block of the 1-Ga. The s-filter on the output is applied to each output block. The properties have obvious analogs for measurement and preparation circuits. Level-1 circuits that satisfy properties $0-3$ obey the exRec-Cor property defined in Chapter 5.

7.3.2 Steane error-correction in detail

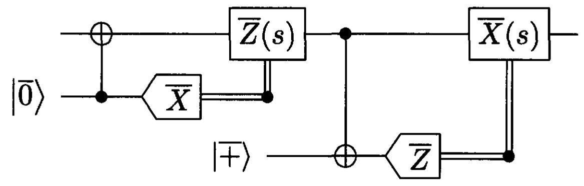


Figure 7-3: Steane's error correction method for CSS codes involves coupling two encoded and verified ancilla's to a block of data qubits. The ancilla qubits are then measured in the logical Z-basis or X-basis and the syndrome s is determined.

The Steane error correction circuit for CSS codes (Steane-EC) is shown in Figure 7-3. The $|\overline{0}\rangle$ and $|\overline{+}\rangle$ ancilla blocks in Fig. 7-3 can be prepared in the following way. First n qubits are encoded using circuits derived from the generator matrix of a classical coset code of $C_2^{\perp}$. The memory locations in the encoder are determined using Steane's Latin rectangle method explained below [Ste02]. Then the ancillas pass through a verification circuit. This error detection circuit measures the X and Z stabilizer generators of the encoded state some number R times. For a $|\overline{0}\rangle$ ancilla each round is given by the circuit in Figure 7-4. The Hadamard-conjugate of the circuit is used for a $|\overline{+}\rangle$ ancilla. If we detect any errors in any of the R rounds, the encoded state is rejected. Otherwise, the state is accepted and used for syndrome extraction. The number of attempted preparations L contributes to the overhead.

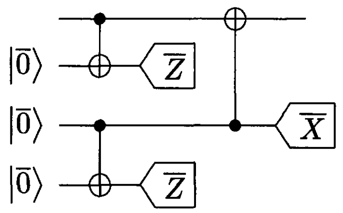


Figure 7-4: The ancilla verification circuit for one round of error detection.

Steane's Latin rectangle method for optimizing encoding circuits

There is a simple method for minimizing the number of memory locations in an ancilla encoding circuit. The generator matrix G of a linear binary code is put into standard form $(I|A)$ using Gaussian elimination. An encoding circuit for the logical zero state can be constructing by looking at the A matrix for the code $C_2^\perp$. Every 1 in the A matrix gives a CNOT gate in the encoder. The control qubits are the 1s in the I part of G and the target qubits are the 1s in the A part of G.

For example, we have $G = (1010101, 0110011, 0001111)$ for the $[7,3,4]$ code, which is the $C_2^\perp$ for Steane's [[7,1,3]] code. Transposing columns 3 and 4 gives the standard form and $A = (1101, 1011, 0111)$. This means there are 9 CNOT gates in the logical zero encoder. We can assign each CNOT a time-step so that no qubit is involved in two gates at once. That constraint makes a time-step assignment the same as finding a partial Latin rectangle. The Latin rectangle to complete is

$$\begin{bmatrix} ? & ? & & ? \\ ? & & ? & ? \\ & ? & ? & ? \end{bmatrix} \tag{7.5}$$

and one possible completion is

$$\begin{bmatrix} 1 & 2 & & 3 \\ 2 & & 3 & 1 \\ & 3 & 1 & 2 \end{bmatrix}. \tag{7.6}$$

The circuit corresponding to this Latin rectangle applies 3 CNOT gates per timestep for 3 timesteps, denoted (timestep, control, target): $(1,1,4)$, $(1,2,7)$, $(1,3,6)$, $(2,1,5)$, $(2,2,4)$, $(2,3,7)$, $(3,1,7)$, $(3,2,6)$, $(3,3,5)$. We have to undo the qubit permutation that occurred in the transformation to standard form, so at the end we should switch back qubits 3 and 4. This is the smallest depth (3) that a circuit for A can have; the maximum row or column sum w of A.

The problem of completing the Latin rectangle and therefore of computing the optimal time-step assignment for a matrix A is equivalent to edge coloring a bipartite graph with the minimum number of colors. We construct the graph in the following way. The left set of vertices corresponds to the control qubits. The right set of vertices corresponds to the target qubits. A control and target vertex are connected by an edge if there is a CNOT between those two qubits. Assign a color to an edge to indicate the time-step of the CNOT gate. Since we cannot have two CNOT gates occur at the same time using the same qubit, all of the edges incident to a given vertex must have different colors. This means that a valid schedule corresponds to an edge coloring of this bipartite graph. The graph is bipartite since there is a set of control vertices that are only connected to target vertices and a set of target vertices that are only connected to control vertices. By Hall's theorem [Ste02], there is a coloring using w colors, and w colors is the minimum possible number of colors. Here $(w+1)$ is maximum weight of the rows of A. An algorithm that finds an edge coloring with w colors in time $O(nN_{\text{CNOT}})$ is given in [KR00]. Here n is the number of qubits that are to be encoded (i.e. number of vertices) and N_{CNOT} is the number of CNOT gates in the encoder (i.e. number of edges).

Locations in Steane error-correction

Steane error correction for a CSS code $CSS(C_1, C_2)$, $C_2^\perp \subseteq C_1$ uses $|\overline{+}\rangle$ and $|\overline{0}\rangle$ ancilla states. These states can be encoded directly from the generator matrices of C_1 and $C_2^\perp$, respectively, according to a well-known procedure. The generator matrix G has n columns and $(n-k)/2$ rows for $C_2^\perp$ or $(n+k)/2$ rows for C_1. Using Gaussian elimination it is put into standard form $G = (I|A)$ where I is an identity matrix and A is a binary matrix. The ith row of the generator matrix specifies the controls and targets of w_i CNOT gates, where w_i is the weight of the row minus one. The depth of the resulting CNOT circuit is $w = \max\{w_i\}$, assuming equal cost for any pair of qubits to communicate. This assumption is generally not true for device models such as that discussed in Chapter 3, but we try to keep the analysis simple by assuming it, with the understanding that a layout could be included in the model. The number of fault locations in an encoder is summarized by the following expressions:

$$A_{\text{enc}}(n,k,w) \leq n + w(n+k)/2, \text{ no memory noise}$$
$$A_{\text{enc}}(n,k,w) \leq n + wn, \text{ memory noise.}$$

For particular states, different scaling is possible. For example, for the Bacon-Shor codes one can make the encoded ancillas using $O(n)$ 2-qubit gates.

One method of verifying the encoded ancilla against low-weight correlated errors is to use transversal gates to perform error detection. A general error detection method consumes three additional ancilla and uses 3 transversal CNOT gates and 3 transversal measurements; see Figure 7-4. If the encoder is considered a black box and the error detection method is fixed, then t rounds of error detection are necessary and sufficient to ensure that the logical error rate scales like $O(p_0^{t+1})$. If we desire error rate scaling like $O(p_0^{R+1})$, $R \leq t$, then we use R rounds. The cost of verifying is:

$$A_{\text{ver}}(n,k,w,R) \leq R(3A_{\text{enc}}(n,k,w)+6n), \text{ no memory noise}$$
$$A_{\text{ver}}(n,k,w,R) \leq R(3A_{\text{enc}}(n,k,w)+6n)+n, \text{ memory noise.}$$

Again, these expressions assume equal cost for any pair of qubits to communicate.

Finally, we can write expressions for the total number of fault locations in a CNOT extended rectangle using Steane error correction:

$$A(n,k,w,R) \leq 8A_{\text{enc}}(n,k,w)+8A_{\text{ver}}(n,k,w,R)+17n.$$

If we set $R=t$, then the total number of fault locations is $A(n,k,w,R) = O(wnt)$ using this method of error correction and assuming equal communication costs between qubits. In the worst case this can be $O(n^3)$.

7.3.3 Universal fault-tolerant gate constructions

If one is able to perform any Clifford group gate transversally, including H and K, we reviewed how to obtain a universal set of gates in Chapter 5. Universality for general CSS codes can in principle be obtained using the technique of *injection-and-distillation* [Kni05a, BK05, Rei06b]. Indeed, magic state distillation gives a constructive proof that C_2 together with any unitary gate outside of C_2 is computationally universal. Let us briefly review how one may perform fault-tolerant computation for CSS codes for which, of the Clifford group gates, only the CNOT and Pauli operations are transversal. Note that a CSS code with only its transversal CNOT gives us the ability to fault-tolerantly prepare the states $\{|\overline{+}\rangle, |\overline{0}\rangle\}$ and perform transversal $\overline{X}$ and $\overline{Z}$ measurements. However we do not necessarily have a fault-tolerant realization of the Hadamard gate H.

In this case the problem of constructing fault-tolerant single qubit Clifford gates can be reduced to the problem of preparing the encoded $|\overline{+i}\rangle \propto |\overline{0}\rangle + i|\overline{1}\rangle$ ancilla, see [AP07]. In particular, the gates K $\propto \exp(-i\pi Z/4)$ and Q $\propto \exp(+i\pi X/4)$ generate the single-qubit Clifford group and can be implemented given a $|{+i}\rangle$ ancilla, see Figure 7-5 and Figure 7-6.

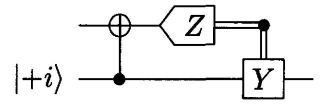


Figure 7-5: The K gate using a $|{+i}\rangle$ ancilla.

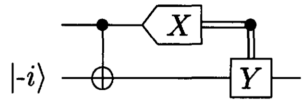


Figure 7-6: The Q gate using a $|{-i}\rangle$ ancilla.

An encoded $|\overline{+i}\rangle$ ancilla can be produced using the method of injection-and-distillation. The specific procedure in this case is shown in Figure 7-7: (a) inject each of seven copies of the state $|{-i}\rangle$ into the [[7,1,3]] Steane code by teleportation, (b) extract the X and Z syndromes of the Steane code and correct the errors, but correct X errors using Y operators, (c) decode the Steane code to yield a single $|\overline{+i}\rangle$ ancilla.

7.4 Evaluating codes for quantum fault-tolerance

This section discusses what we call the AGP method for evaluating fault-tolerant gates. This method is based on evaluating the correctness of an extended rectangle. We extend this method in Sections 7.4.1 and 7.4.2 to allow us to apply it to relatively large stabilizer codes. Furthermore, the same methods are applicable for evaluating codes with $k > 1$, which is essentially equivalent to evaluating a collection of CNOT extended rectangles [SI05].

A simplifying assumption we make is that the inner codes, one level above the physical level, can be evaluated reasonably with a depolarizing noise model. This is perhaps a reasonable assumption after a biased noise code, for example. However, outermost codes are best evaluated with an adversarial noise model due to correlations that are possibly introduced by concatenation. If the distance of the outer codes is relatively small, then this is not a problem for the method.

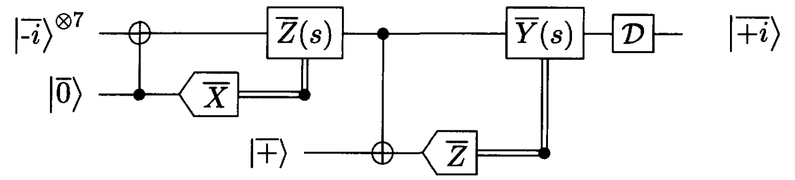
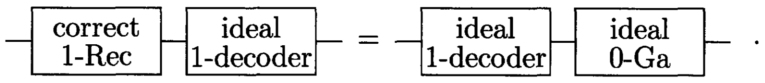


Figure 7-7: A circuit for $|\overline{+i}\rangle$ ancilla distillation.

7.4.1 The Aliferis-Gottesman-Preskill (AGP) method

Recall that a rectangle is correct if the rectangle followed by an ideal decoder is equivalent to the ideal decoder followed by the ideal gate (0-Ga) that the rectangle simulates:



As said before, an extended rectangle (ex-Rec) consists of a 1-Ga along with its leading and trailing error corrections. The extended rectangles make an overlapping covering of the circuit. A set of locations inside an ex-Rec is called benign if the 1-Rec is correct for any set of faults occurring on these locations. If a set of locations is not benign, it is malignant. The design principles of **strict** fault-tolerance are described in Section 7.3.1. If these properties hold for the 1-Ga and 1-EC, these gadgets for a $[[n, 1, d]]$ code with $t = \lfloor (d-1)/2 \rfloor$ are called t-strictly fault-tolerant. The important consequence of these conditions is that for a $[[n, k, d]]$ code with t-strictly fault-tolerant constructions one can show that any set of t or fewer locations in the ex-Rec is benign. A construction is called **weakly fault-tolerant** when, for a code that can correct t errors, sets of $s < t$ locations can be malignant. Weak fault-tolerance is a useful concept in optimizing thresholds since weakly fault-tolerant circuits can be more compact than strictly fault-tolerant circuits, hence allowing for fewer fault locations. At the same time, weak fault-tolerance allows some low-weight faults to be malignant but if the number of such faults is small then the threshold is not much affected.

The fault-tolerant schemes described in Chapter 8 are 1-strictly fault-tolerant implying that single faults can never be malignant. More precisely, any single fault in a 1-EC or a 1-Ga never propagates to become a weight-2 error in a block. In Steane-EC, when we prepare ancillas with at least two attempts ($L \geq 2$) and one error detection stage ($R = 1$), we eliminate malignant faults of weight 1. For $R = 1$ the EC is not 2-strictly fault-tolerant since there is a pair of faults, one in each of the first two encoders, generating a high weight (possibly higher than t) error that passes the error detection circuit undetected. Since the number of these events is quite rare, they will not contribute much to the failure probability. It is possible to show that $R = t$ and $L = t + 1$ is sufficient for t-strict fault-tolerance for a code that can correct t errors.

Let us review why the extended rectangle is the central object in a fault-tolerance analysis. An encoded circuit where the physical gates (0-Ga) have been replaced by rectangles can also be viewed as an encoded circuit with 0-Ga's with a different error model. This can be achieved simply by inserting perfect decoder-encoder pairs between the rectangles, see [AGP06]. In an ex-Rec with malignant faults, the rectangle will correspond to a faulty 0-Ga, whereas for benign faults the rectangle will correspond to a perfect 0-Ga.

The reason that one has to take into account an ex-Rec and not merely a Rec is that faults in the leading 1-EC can combine with faults in the 1-Rec to produce malignant faults. Those faults do not make the gate before the leading 1-EC fail, but they make the gate encoded in the Rec fail. In principle one may think that such arguments would also apply to the input of the ex-Rec, namely an incoming error could combine with a seemingly benign fault in the ex-Rec and give rise to an incorrect rectangle. Thus in principle one has to consider the malignancy of sets of faults given a possible *worst case* input to the extended rectangle.

However one can argue quite generally for stabilizer codes and their error correction that malignancy does not depend on incoming errors to the ex-Rec. Recall that any element of $Z(S)/S$ is a logical operator mapping codewords onto each other. All other Pauli operators $P \notin Z(S)$ anti-commute with at least one element in S and map a code word outside the code space indicated by a non-zero syndrome. Thus the Pauli group G_n can be partitioned into cosets of $Z(S)$ and each of these cosets is labeled by a different syndrome. The lowest-weight member of each coset is called the coset leader. Standard syndrome decoding finds, for each given syndrome, a coset leader with lowest weight and chooses this as the error correction. Thus the low-weight (non-degenerate) correctable errors correspond to distinct syndromes whose coset leader corrects the error. For high-weight errors E_i all we can say is that $E_i E_{\text{correct}} \in Z(S)$ since E_i and E_{correct} have the same syndrome.

Now let us consider the issue of incoming errors to an ex-Rec and assume the following standard properties of stabilizer error correction. First, we assume that the part of the 1-EC circuit that couples any ancillas to the incoming data is deterministic (the choice of which ancillas to couple may depend on some error detection or ancilla verification, but this is independent of incoming errors on the data). This property holds for all known error correction circuits. Furthermore, given a stabilizer S and the incoming error E_{in} on an encoded state, let the 1-EC be such that the syndrome of the 1-EC uniquely determines in which coset of $Z(S)$ in the Pauli group the error E_{in} lies. In this sense the 1-EC must be complete error correction for the code that is used. For example, if for a CSS code the 1-EC only does Z error correction whereas X errors can map the state outside the code space, the syndrome information effectively partitions the Pauli group into cosets of $Z(S_X)P_n(X)$. Here $P_n(X)$ is the subgroup of Pauli operators that only contain X and I

and $Z(S_X)$ is the normalizer of the stabilizer subgroup S_X with only X and I Pauli operators. In this case the syndrome does not uniquely assign the incoming error to a coset of $Z(S)$. Thirdly, upon any incoming error E_{in} a perfect 1-EC determines a syndrome that corrects E_{in} modulo a logical error (given by an element in $Z(S)$). This is a basic property of stabilizer error correction as described above.

Let then the incoming state to an ex-Rec be a state in the code-space of the stabilizer with an additional error E_{in}. We want to show that the state that comes out of the leading 1-EC is again some state in the code space with an additional error E_{out} that only depends on the errors inside the 1-EC, E_{ec}; i.e., $E_{\text{out}} = f(E_{\text{ec}})$ where f is independent of E_{in}. Any error correction 1-EC for stabilizer codes can be implemented with Clifford gates. Given an incoming error E_{in} and error inside the 1-EC E_{ec}, it follows (because a 1-EC for any stabilizer code can be implemented with Clifford gates) that the 1-EC has syndrome $s(E_{\text{in}}h_1(E_{\text{ec}}))$ where h_1 is a function independent of E_{in}. Based on the syndrome the correction step will be some $E_{\text{correct}} = E_{\text{in}}h_1(E_{\text{ec}}) \mod Z(S)$. Before error correction the data has error $h_2(E_{\text{ec}})E_{\text{in}}$ where $h_2(E_{\text{ec}})$ is the part of E_{ec} that has propagated to the data. After error correction the data thus has error $h_2(E_{\text{ec}})h_1(E_{\text{ec}}) \mod Z(S)$. We strip off the logical error in $Z(S)$ and identify $E_{\text{out}} = h_2(E_{\text{ec}})h_1(E_{\text{ec}})$.

7.4.2 Monte-Carlo implementation of method

Statistical bound calculations and pseudothreshold estimation

Here we consider details of the mathematical calculation of the the failure probability of an exRec, i.e., the probability that it is "not correct". We can separate out the contributions to failures due to fault patterns of different weights by writing

$$\mathbb{P}(\text{incorrect}) = \sum_{j=1}^{A} \mathbb{P}(\text{incorrect} \mid \text{weight } j \text{ fault pattern})\mathbb{P}(\text{weight } j \text{ fault pattern}) \tag{7.7}$$

Here there are A locations in the exRec. So long as the failure probability of any location has probability x, then

$$\mathbb{P}(\text{weight } j \text{ fault pattern}) = \binom{A}{j} x^j (1-x)^{A-j} \tag{7.8}$$

In AGP, the failure expression is grouped in this way:

$$\begin{aligned}\mathbb{P}(\text{incorrect}) &= \sum_{j=1}^{A} \alpha_j x^j (1-x)^{A-j}, \\ \alpha_j = \mathbb{P}(\text{incorrect} \mid \text{weight } j \text{ fault pattern})\binom{A}{j} &= p_j \binom{A}{j}\end{aligned} \tag{7.9}$$

For adversarial noise α_j is an integer, since each fault pattern either definitely fails or definitely doesn't; for depolarizing noise α_j is an integer divided by 3^j.

AGP gets a simplified upper bound on this expression using two facts:

$$\begin{aligned} x^2(1-x)^{A-2} &\leq x^2; \\ \sum_{j=3}^{A} p_j x^j (1-x)^{A-j} &\leq \binom{A}{3} x^3,\ p_j \leq \binom{A}{j} \end{aligned} \tag{7.10}$$

This last statement, and its generalization, being proved thus:

$$\binom{A}{k} x^k = \binom{A}{k} x^k \sum_{j=k}^{A} \binom{A-k}{j-k} x^{j-k}(1-x)^{A-j} = \sum_{j=k}^{A} \binom{A}{j}\binom{j}{k} x^j (1-x)^{A-j} \geq \sum_{j=k}^{A} \binom{A}{j} x^j (1-x)^{A-j}. \tag{7.11}$$

So, in AGP the failure probability is bounded as

$$\mathbb{P}(\text{incorrect}) \leq \alpha_2 x^2 + \binom{A}{3} x^3 \tag{7.12}$$

and an upper bound on the noise threshold is given by setting the two sides of this expression equal and setting the probabilities p and x equal; that is, the lower bound p^* is

$$p^* = \alpha_2 (p^*)^2 + \binom{A}{3} (p^*)^3. \tag{7.13}$$

The integer α_2 is obtained exactly by examining every weight-2 fault pattern and counting the number for which the exRec fails, and A is the number of locations in the exRec.

Of course, a much tighter lower-bound equation for p^* can be written down using Eqs. (7.10) differently:

$$p^* = \sum_{j=2}^{w} \alpha_j (p^*)^j (1-p^*)^{A-j} + \binom{A}{w+1} (p^*)^{w+1} \tag{7.14}$$

The problem is that the computation of α_j becomes expensive very quickly as j increases, going like

$3^j\binom{A}{j}$ (the 3^j comes from running through all Pauli errors consistent with a given fault pattern).

We have a problem, therefore, using exactly the AGP approach, particularly because we will be interested in examining codes with large distance d for which j in Eq. (7.14) must be taken to large values (at least $\lfloor (d+1)/2 \rfloor$) to give useful answers.

Our solution to this problem is to compute an accurate *statistical estimate* of the lower bound. To estimate the integer α_j, or the proportion of failures p_j, we draw uniformly and at random from all patterns of j faults, and compute the probability of failure over this sample. This gives an unbiassed estimate $\bar{p}_j$ of p_j. From elementary statistical considerations, the error bar associated with this estimate is $\sqrt{\frac{\bar{p}_j(1-\bar{p}_j)}{N}}$, where N is the number of samples. So, to get an error bar of, say, 1% of the mean, then the number of samples should be $N = 10^4 \frac{1-\bar{p}_j}{\bar{p}_j}$. For small p_j (a common case), this just says that to get 1% accuracy, sampling should be done until about 10,000 failures have been seen. For adversarial noise the running time goes like $3^j N$ (the 3^j coming from the need to run through the Pauli-operator identity of each fault), which is still expensive for large j; but for depolarizing noise, for which the samples can be specific Pauli-operator strings, the running time is $O(N)$, having therefore no direct dependence on j (there is an indirect dependence through the j dependence of p_j). So, with this sampling approach, many more threshold estimates become feasible than with the original AGP approach. Indeed, even for adversarial noise, the sampling method is feasible for the Golay code.

While some of our calculations have been done with exactly the sampling estimate of α_j just described, in fact an even easier sampling approach can be reliably adopted for depolarizing noise, in which we sample over the whole distribution of possible fault patterns at the same time, rather than individually over subdistributions with particular weights. For this, we write a different conditional expression for the total failure probability, more fine-grained than Eq. (7.7):

$$\mathbb{P}(\text{incorrect}) = \sum_{\substack{\text{fault} \\ \text{patterns}}} \mathbb{P}(\text{incorrect } |\text{fault pattern})\mathbb{P}(\text{fault pattern}) \tag{7.15}$$

The probability of any particular fault pattern is $\frac{1}{3^j}x^j(1-x)^{A-j}$, and the conditional probabilities in this expression are simply zero or one: a fault pattern either fails or it doesn't. A fair sample of the fault patterns can be generated by randomly assigning faults, with probability x, to every location in the exRec. The failure probability is estimated by the percentage of failures produced by these samples. As a check, we have seen that estimated failure probabilities obtained in this way agree very closely with that obtained by estimating the individual α_j coefficients.

This estimate, in the end, has a close resemblance to the Monte Carlo calculations that have been used by various authors to estimate the noise threshold for fault tolerance. The difference is that we use a much more rigorous definition of failure, as established by AGP. This involves including faults throughout the exRec, and exploiting the fact that, for the error correction techniques we use, the nature of the errors entering the extended rectangle do not matter for the analysis. This elevates the Monte Carlo approach from a heuristic one to an essentially rigorous one; our approach has no systematic errors, only simple statistical errors.

Specifics of the pseudothreshold calculations

Given the AGP method the numerical problem to be solved is whether a Rec is correct given a set of faults in the ex-Rec containing it. This set of faults is generated using depolarizing noise with error probability p_0 for each location in the circuit. We calculate the failure rate of the ex-Rec, i.e. the probability that the Rec is not correct, for a fixed R and L. This implies that sometimes there are no verified ancillas available for a 1-EC. If this happens for *any* of the 1-ECs inside the extended rectangle, we call this a failure of the extended rectangle.

We will estimate the failure rate of a CNOT ex-Rec, since this is by far the biggest circuit among the Clifford ex-Recs. As we argued in Section 7.3.3, the non-Clifford (and possibly other Clifford) gates will be implemented via injection-and-distillation and they will not affect the threshold. Pauli gates are not applied within a Clifford ex-Rec because they can be stored in classical memory as the Pauli frame and applied only prior to the execution of non-Clifford gates.

Given a fixed R and L, we will estimate the failure rate $p_1(p_0) = \frac{N_{\text{fail}}}{N}$ where N_{fail} is the number of Monte-Carlo samples that fail (i.e. the number of times we simulate the extended rectangle with randomly generated faults and observe that the rectangle is incorrect) and N is the total number of runs. With high probability this estimated p_1 lies within one standard deviation of the real p_1. In this way we collect data points $p_1(p_0)$ for different values of p_0. We then take these points as the mean of a normal distribution for each p_0. We sample from these normal distributions and for each set of samples we determine a small degree polynomial $p_1(p_0)$ fitting the samples. The equation $p_1(p_0) = p_0$ gives us a sample of the threshold and we put an error bar on this result by calculating the standard deviation of the obtained threshold samples.

The way we test for correctness of a rectangle for a given pattern of faults in the ex-Rec is as follows: Let E_{out} be the outgoing error of the leading 1-EC. We use syndrome decoding to determine the coset leader E_{lead} corresponding to the coset of $Z(S)$ in the Pauli group of this E_{out}. We propagate this E_{lead} through the rectangle, let $f(E_{\text{lead}})$ be the outgoing error on the data. We follow the rectangle by an ideal decoder and let E_{correct} be the correction suggested by the ideal decoder. Then we test whether $E_{\text{correct}}f(E_{\text{lead}})$ commutes with both $\overline{X}$ and $\overline{Z}$. If it does, we infer that no logical faults occurred, hence the rectangle was correct. Otherwise we call failure.

In Chapter 8, we use the Monte-Carlo simulation to estimate p_1, but the number of samples required becomes quite large if one wants to estimate p_1 with good relative error for small p_1. In such cases we extrapolate the values for p_1 obtained from larger values of p_0, see Section 8.5.2.

7.4.3 Software and computer use

We have developed software tools implementing the methods in this chapter. The quantum circuits for the CNOT ex-Recs based on CSS codes are highly structured and can be mechanically assembled in $O(n^3)$ time for block-size n given the classical codes C_1 and C_2. We have used MAGMA [WCP97] and/or GAP [GAP07] (using the GUAVA package [CRB$^+$]) to construct quantum codes and compute their parameters. The code stabilizers are copied from the computer algebra programs into our circuit synthesis and simulation programs, where they are again verified to have the required commutation relations.

The simulation and circuit synthesis programs are implemented in C++ and use MPI for communication during embarrassingly parallel tasks. The project is entirely open source and makes use of preexisting open source libraries such as a Galois field implementation and a weighted matching algorithm [Gab74]. Importantly, the same functions and procedures are used in the exact counting simulation and the Monte-Carlo simulation. This gives us increased confidence in the simulation output.

For example, the symmetries of the pair count matrix for some distance-3 code circuits and the lack of single-location malignancies in all circuits strongly suggests that our automated circuit constructions are indeed fault-tolerant. Furthermore, we strictly check all input and intermediate results for consistency at runtime. The programs can be optimized and further improved, but we leave this to future work and encourage development by making the code publicly available.

7.5 Conclusion

This chapter has defined what we mean by an effective code architecture and shown how to evaluate the architecture one layer at a time. We have argued that such an architecture likely consists of multiple specialized codes. In particular, the rough estimates in this chapter suggest that a three level code architecture, consisting of a physical, inner and outer code, will be effective for applications beyond the reach of modern digital computers. We have explained how code parameters determine the quantitative properties of a code architecture; i.e., the threshold scaling with block size and minimum distance, scaling of overhead with rectangle parameters, and expected logical error rates at each level of coding. Furthermore, the chapter reviewed general design rules for error-correcting code rectangles and explained why, for CSS codes, it is adequate to study CNOT rectangles. Finally, we gave two new Monte-Carlo adaptations of the methods of Aliferis, Gottesman, and Preskill that allow us to evaluate larger codes and rectangles than the original method permitted. The results we obtain from applying this method are closely related to rigorous bounds [AGP06].

Chapter 8

Comparative study of inner codes

8.1 Introduction

Chapter 7 argued the importance of evaluating a code architecture one level at a time, so that the whole architecture need not be simulated at once. The chapter also introduced a Monte-Carlo method for numerically estimating thresholds that is tied to rigorous theory. In this chapter, we apply those methods and ideas to complete a comparative study of inner codes in a concatenated code hierarchy [CDT07].

Figure 8-1 reminds us of the purpose of inner codes in the proposed hierarchy. They are used after and above physical codes to suppress logical error rates below 10^{-6} while maintaining a high threshold. Below error rates of 10^{-6}, more efficient error-correcting codes can be used to reach error rates demanded by applications. Our goal is to identify inner codes with high thresholds and low overheads that can attain logical CNOT error rates of around 10^{-7} from base error rates of 10^{-4}.

This chapter is organized as follows. Section 8.2 presents the codes we have chosen as inner codes, why they were selected, and roughly how their gate and error-correction circuitry is implemented. Section 8.3 describes constructions for low rate polynomial codes and explains how to map circuits on high-dimensional subsystems down to gates on qubits. Section 8.4 gives many circuit construction details for the inner codes. Finally, Section 8.5 discusses the results of the comparative inner code survey and Section 8.6 concludes.

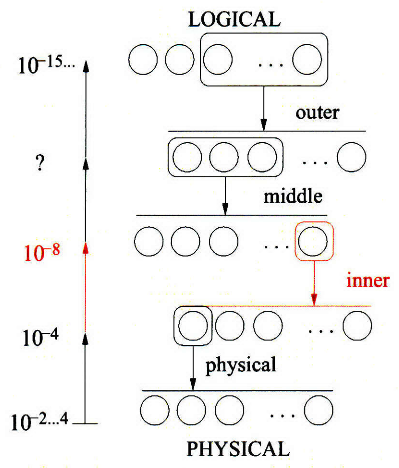


Figure 8-1: A schematic of a concatenated hierarchy of specialized codes. The inner codes that are the subject of this chapter are highlighted [in red online].

8.2 Selection of inner codes

For our study it is necessary to select a subset of quantum codes. We focus on codes that are likely to have a good threshold, possibly at the cost of a sizeable but not gigantic overhead. Recalling Eq. (7.2), it is clearly desirable to minimize the number of locations N and maximize t. This consideration has led us to primarily consider Calderbank-Shor-Steane (CSS) codes. The advantage of a transversal CNOT is that it minimizes the size of the encoded CNOT; the bulk of the CNOT rectangle will be taken up by error correction (EC). This is favorable for the noise threshold of C_{inner}. Secondly, minimizing the error rate of the encoded gate $C_{\text{inner}}(\text{CNOT})$ will be useful at the next level of encoding, because CNOTs occur frequently in EC and their error rates play a large role in determining whether error rates are below the threshold (of C_{outer}). However, to demonstrate that this restriction to CSS codes is warranted we also consider the non-CSS 5-qubit code $[[5,1,3]]$ which is the smallest code that can correct a single error. We indeed find that this code performs *worse* than Steane's 7-qubit code $[[7,1,3]]$; see Section 8.5 and the Data Tables in Appendix B.

Section 8.2.1 discusses our specific code selections and Section 8.2.2 is an overview of the most important or previously unpublished construction details.

8.2.1 Choice of codes

The codes that we have studied are listed in Table 8.1. All codes in this table are CSS codes with the exception of the [[5,1,3]] code. Some of these codes have been previously analyzed by Steane in Ref. [Ste03]. There exist various families of binary CSS quantum codes; the families are the quantum Reed-Muller codes, the quantum Hamming codes, the quantum BCH codes, the surface codes and the sub-system Bacon-Shor codes. In our study we consider only a single member of the quantum Reed-Muller family, a $[[15, 1, 3]]$ code, since these codes typically don't have very good distance versus block-size [Ste99b]. The $[[15, 1, 3]]$ was first constructed in [KLZ96] from a punctured Reed-Muller code RM$(1, 4)$ and its even sub-code. It is the smallest known distance-3 code with a transversal T gate.

We study various quantum Hamming and quantum BCH codes (see a complete list of quantum BCH codes of small block-size in [GB99]) which are constructed from self-orthogonal classical Hamming and BCH codes respectively. We have chosen those codes that encode a single qubit and have maximum distance for a given block size. We have included the previously studied Bacon-Shor codes and the surface codes in our study. We have also included the concatenated 7-qubit code $[[49, 1, 9]]$ which we use in the way that was proposed by Reichardt in [Rei04], see the details in Section 8.2.2.

Another family of codes that has been proposed for fault-tolerance [ABO97, ABO99] are the quantum Reed-Solomon codes or polynomial codes. These are codes that are naturally defined on qudits. In this study we consider them as candidates for inner codes. An alternative use is to consider them as outer codes, perhaps with high rate (not discussed in this dissertation), where one uses a good inner code to map the qubits onto qudits. In our study we assume that quantum information is presented in the form of qubits and hence we will consider these codes as binary stabilizer codes. We specifically chose to include the $[[21, 3, 5]]$ (a $[[7, 1, 4]]_8$) and $[[60, 4, 10]]$ (a $[[15, 1, 8]]_{16}$) from the family of dual-containing polynomial codes over $GF(2^m)$, because they are the smallest error-correcting polynomial codes in this family.

We find it impractical to simulate the encoded CNOT gate for BCH codes in this table which have block-size larger than $[[47, 1, 11]]$. The threshold for these bigger codes will benefit considerably from the fact that t/n is quite high. Some semi-analytical values for the thresholds of these codes have been given in [Ste03]. Even with good thresholds, these bigger BCH codes have limited applicability due to their large overhead. The inner code should be picked to obtain a logical error rate that is well below the threshold of some good block code but only at the price of a moderate overhead.

PARAMETERS	NOTES
$[[5,1,3]]$	non-CSS five qubit code [LMPZ96]
$[[7,1,3]]$	Steane's 7-qubit code [Ste96]
$[[9,1,3]], [[25,1,5]],$ $[[49,1,7]], [[81,1,9]]$	Bacon-Shor codes [AC07]
$[[15,1,3]]$	Quantum Reed-Muller code [SI05, KLZ96]
$[[13,1,3]], [[41,1,5]],$ $[[85,1,7]]$	Surface codes [BK98, FM01]
$[[21,3,5]]$	Dual-containing polynomial code on $GF(2^3)$ [GGB99]
$[[23,1,7]]$	Golay code (cyclic) [Rei06a]
$[[47,1,11]]$	Doub.-even dual-cont. quadratic-residue code [GB99]
$[[49,1,9]]$	Concatenated $[[7,1,3]]$ Hamming code [Rei04]
$[[60,4,10]]$	Dual-containing polynomial code on $GF(2^4)$ [GGB99]
$[[79,1,15]], [[89,1,17]],$ $[[103,1,19]], [[127,1,19]]$	BCH codes, not analyzed [GB99]

Table 8.1: A list of the codes included in our study.

8.2.2 Specific code considerations

Specific properties of a quantum code can often be used to simplify the error-correcting circuits. This section discusses each family of codes and the optimizations that have and have not been implemented, or the code properties that have been used to modify the EC and 1-Ga circuits.

In general, we have opted to focus on the error-correcting properties of the codes rather than the possible simplifications to the Steane-EC network. One of the reasons for this approach is that it is not clear whether verification circuits that perform the minimal number of checks are superior to verification circuits that perform many thorough tests. Furthermore, changes to the network are difficult to parameterize and systematically study because there are many possible choices and few are clearly the best.

Reichardt has suggested a generic optimization that uses different encoders for each logical ancilla in the verification circuit [Rei04]. This optimization can reduce the number of necessary rounds of verification and possibly decrease the probability of correlated errors at the output of the verification circuit, conditioned on acceptance. We do not use this optimization for any of the codes in this study.

The Steane and Golay codes are constructed from perfect classical codes. Perfect codes have the property that every syndrome locates a unique error of weight $w \leq t$. As Ref. [AGP06] observed, some parts of the error detection circuit can be removed for a CSS code constructed from perfect classical codes and the construction remains strictly fault-tolerant. Again we do not use this optimization.

For the Bacon-Shor codes we don't use Steane's Latin rectangle encoding method, but rather the simpler method described in [AC07]. We do use the standard verification method for the bigger Bacon-Shor codes and not the simpler verification method in [AC07]; nevertheless the simple verification networks have been computed and are described in Section 8.4.4.

For the surface codes we consider both Steane-EC and Shor-EC to understand their effects on the threshold. We use Shor-EC using bare ancillas as in [DKLP02]. This is fault-tolerant for surface codes on a 5×5 lattice or larger as long as the syndrome measurements are repeated enough times. The number of repeated measurements could in principle be varied, but we choose to repeat the measurements ℓ times for a $\ell \times \ell$ surface code, following [DKLP02].

The $[[49,1,9]]$ concatenated Steane code is one of the CSS codes in our study whose network deviates from the construction described in the previous section. The preparations of $|\overline{0}\rangle$ and $|\overline{+}\rangle$ do not include a verification circuit. Instead each 7-qubit block has an error detection after each [[7,1,3]]-encoded logical gate [Rei04]. A 49-qubit ancilla is rejected if any of these error-detections detects an error. This implies that any single fault will be detected, so the circuit is fault-tolerant. In fact, any pair of faults is also detected, so that a third order event is necessary to defeat the error-detections. The method of using $[[49,1,9]]$ is the one which Reichardt proposed. Unlike in his simulations we restrict ourselves to a finite number of ancilla preparation attempts L, since we care about the total overhead.

The polynomial codes we consider are non-binary codes over 2^m-dimensional qudits. We can choose the parameters of these codes so that when we consider each qudit as a block of m qubits, the Fourier transform and controlled-SUM gates are implemented by bitwise application of Hadamard and CNOT, respectively. In this setting, the code is simply a binary CSS code encoding m qubits that is constructed from a non-binary dual-containing classical code by concatenation with a self-dual basis of $GF(2^m)$. The self-dual basis can be interpreted as concatenation with an $[[m,m,1]]$ code. The advantage of such a construction is that we can decode the syndromes as if they were vectors over $GF(2^m)$, allowing us to correct more higher-order errors than we could otherwise correct as a binary code. To use this advantage, we do not need to change the way we construct the rectangles at all, only the way we interpret the classical measurement outcomes. Section 8.3 reviews and clarifies constructions for these codes.

The $[[5,1,3]]$ code does not have a transversal two-qubit gate, but it does have a 3-qubit gate

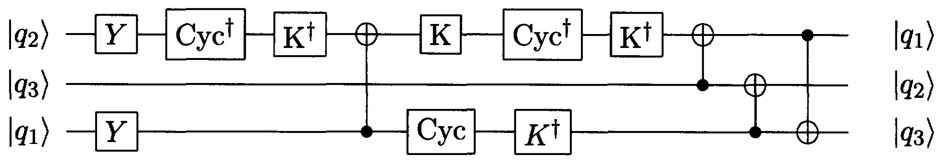


Figure 8-2: The encoded implementation of M3 (with an additional permutation of the blocks q_1, q_2, q_3) using the gates CNOT, K, Cyc and Y (and inverses). Each gate in the circuit is applied transversally. The circuit is only a logical operation after completing all of the gates, i.e. CNOT and K are not valid transversal gates for $[[5,1,3]]$.

M3 [1]. The M3 gate is a Clifford gate that can be combined with stabilizer-state preparations and transversal Pauli measurements to yield any gate in the Clifford group [Got97]. Specifically, CNOT, K, and Cyc gates (and their inverses) can be constructed from the M3 gate in this way. Here Cyc = KHKH acts as $X \to Y \to Z \to X$. The fault-tolerant implementation of M3 is shown in Figure 8-2. In our study we analyze the M3 extended rectangle.

The $[[5,1,3]]$ construction also differs from other CSS constructions because we use Knill (or teleported) error correction (Knill-EC) [Kni05a]. $[[5,1,3]]$ is the smallest distance-3 quantum error-correcting code and it is a perfect quantum code. Gottesman has shown how to compute fault-tolerantly with this code [Got97], and there have been some numerical studies of logical error rates using Shor-EC [Fow05], but to our knowledge the threshold for this code has never been published. We also simulate the extended M3-rectangle assuming that the logical Bell pairs of Knill-EC are perfect, in order to show that even in that case the threshold is not very good, see Section 8.5. For [[5,1,3]] the R and L parameters are replaced by NC and NB, denoting the number of cat state preparation attempts per Pauli measurement and the number of logical Bell state preparation attempts per error correction, respectively. A circuit to prepare and verify encoded Bell pairs for Knill error correction for [[5,1,3]] is shown in Figure 8-7.

The construction for the $[[15,1,3]]$ Reed-Muller code is entirely standard. Since this code is not constructed from a dual-containing classical code, the $|\overline{0}\rangle$ and $|\overline{+}\rangle$ encoders are not simply related by a transversal Hadamard gate. For the same reason, the code can correct more X errors than Z errors.

[1]M3 is defined by the following action on Pauli operators: $XII \to XYZ, IXI \to YXZ, IIX \to XXX, ZII \to ZXY, IZI \to XZY, IIZ \to ZZZ$.

8.3 Constructions for polynomial codes

The quantum polynomial codes are an interesting family of quantum codes derived from classical Reed-Solomon codes. Polynomial codes were used in the first proofs of the threshold theorem and have several interesting properties [ABO99, AG03]. We review them carefully here and introduce some new material about mapping fault-tolerant gates on qudit codes down to gates on qubits. The polynomial codes constructed here have low rate, but we hope that the discussion may be useful for future work in which polynomial codes may be considered for other levels of a code hierarchy.

8.3.1 Generalized Reed-Solomon codes

This section reviews the definition of generalized Reed-Solomon codes [HP03]. Let q be the size of an alphabet $\mathbb{F}_q$ that is a finite field with $q = p^m$ for some prime p and non-negative integer m. Let n be any integer block size with $1 \leq n < q$. Choose an n-tuple $\gamma := (\gamma_0, \gamma_1, \dots, \gamma_{n-1})$ of distinct elements of $\mathbb{F}_q$ and an n-tuple $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$ of nonzero elements of $\mathbb{F}_q$. Let k be an integer with $1 \leq k \leq n$, and $\mathcal{P}_k$ denote the set of polynomials of degree less than k in $\mathbb{F}_q[x]$ including the zero polynomial. The **generalized Reed-Solomon codes** are

$$\mathrm{GRS}_k(\gamma, \mathbf{v}) := \{(v_0 f(\gamma_0), v_1 f(\gamma_1), \dots, v_{n-1} f(\gamma_{n-1})) \mid f \in \mathcal{P}_k\}. \tag{8.1}$$

Choosing the standard basis $\{1, x, x^2, \dots, x^{k-1}\}$ for $\mathcal{P}_k$, the polynomial $f(x) = \sum_{i=0}^{k-1} f_i x^i \in \mathcal{P}_k$ is evaluated at a set of distinct points $\{\gamma_0, \gamma_1, \dots, \gamma_{n-1}\}$. The **evaluation map** $ev : \mathcal{P}_k \to \mathbb{F}_q^n$ for a set of evaluation points $\{\gamma_i\}$ maps a polynomial f to $(f(\gamma_0), \dots, f(\gamma_{n-1}))$. It is clear that $ev(f+g) = ev(f) + ev(g)$ so it is a linear map. Furthermore, $ev(fg) = ev(f) \cdot ev(g)$. The evaluation map has a matrix representation

$$ev f(x) = \begin{bmatrix} 1 & \gamma_0 & \gamma_0^2 & \cdots & \gamma_0^{k-1} \\ 1 & \gamma_1 & \gamma_1^2 & \cdots & \gamma_1^{k-1} \\ \cdots & & & & \cdots \\ 1 & \gamma_{n-1} & \gamma_{n-1}^2 & \cdots & \gamma_{n-1}^{k-1} \end{bmatrix} \mathbf{f} \tag{8.2}$$

called a **Vandermonde matrix** V. Acting from the left with V on a column vector $\mathbf{f}$ of polynomial coefficients $(f_0, f_1, \dots, f_{k-1})$ gives the evaluation of that polynomial on the evaluation points. One important property of the Vandermonde matrix follows from its determinant [Gar04]

$$\det A \propto \prod_{i>j}(\gamma_i - \gamma_j). \tag{8.3}$$

In particular, V is nonsingular iff the γ_i are distinct.

From the preceeding discussion, it is clear that a generator matrix for $\mathrm{GRS}_k(\gamma, \mathbf{v})$ is given by

$$G = \begin{bmatrix} v_0 & v_1 & \cdots & v_{n-1} \\ v_0\gamma_0 & v_1\gamma_1 & \cdots & v_{n-1}\gamma_{n-1} \\ v_0\gamma_0^2 & v_1\gamma_1^2 & \cdots & v_{n-1}\gamma_{n-1}^2 \\ & & \vdots & \\ v_0\gamma_0^{k-1} & v_1\gamma_1^{k-1} & \cdots & v_{n-1}\gamma_{n-1}^{k-1} \end{bmatrix}. \tag{8.4}$$

The interpolation problem is to determine f from the vector of samples $V\mathbf{f}$ [HJ85]. The problem arises in decoding Reed-Solomon codes and in fault-tolerant operations on these codes. The Lagrange's solution to the problem inverts V to find polynomials $L_i(x)$ such that

$$f(x) = f(\gamma_0)L_0(x) + f(\gamma_1)L_1(x) + \cdots + f(\gamma_{n-1})L_{n-1}(x). \tag{8.5}$$

It can be shown that

$$L_i(x) = \frac{\prod_{j=0,j\neq i}^{n-1}(x-\gamma_j)}{\prod_{j=0,j\neq i}^{n-1}(\gamma_i-\gamma_j)}, i = 0, 1, \ldots, n-1 \tag{8.6}$$

is the solution.

Recall that the Reed-Solomon codes have parameters $[n, k, n-k+1]_q$. Indeed, $\mathcal{P}_k$ is k-dimensional, so dim $\mathrm{GRS}_k(\gamma, \mathbf{v}) \leq k$. Suppose there exists $u, w \in \mathcal{P}_k$, $u \neq w$, such that $ev(u) = ev(w)$. Then $\mathbf{v}\cdot ev(u) = \mathbf{v}\cdot ev(w) \in \mathrm{GRS}_k(\gamma, \mathbf{v})$ and $\mathbf{v}\cdot ev(u-w) \in \mathrm{GRS}_k(\gamma, \mathbf{v})$ is the zero codeword. But $u - w \in \mathcal{P}_k$ so it has degree less than k. This contradicts the fact that $ev(u-w) = \vec{0}$, i.e. that $u - w$ has $n \geq k$ roots. Therefore, distinct $u, w \in \mathcal{P}_k$ map to distinct codewords under the evaluation map and dim $\mathrm{GRS}_k(\gamma, \mathbf{v}) = k$. The minimum distance of a linear code C satisfies $d = \min_{c\in C, c\neq 0}\mathrm{wt}(c)$. A nonzero $f \in \mathcal{P}_k$ has at most $k-1$ roots so $\mathrm{wt}(\mathbf{v}\cdot ev(f)) \geq n-(k-1) = n-k+1$. Therefore $d \geq n-k+1$. By the Singleton bound, $d = n-k+1$.

The dual codes of GRS codes are also GRS codes. Since the dual of an MDS code is also MDS,

$\mathrm{GRS}_{n-1}(\gamma,\mathbf{v})^{\perp}$ must be an $[n,1,n]$ code, so its basis vector $\mathbf{w}$ has $\mathrm{wt}(\mathbf{w})=n$. Since

$$\begin{aligned}\mathrm{GRS}_1(\gamma,\mathbf{w}) &= \{(w_0 f(\gamma_0),\ldots,w_{n-1}f(\gamma_{n-1})) \mid f\in\mathcal{P}_1\}\\ &= \{f_0\mathbf{w} \mid f_0\in\mathbb{F}_q\}\end{aligned}$$

we have $\mathrm{GRS}_{n-1}(\gamma,\mathbf{v})^{\perp}=\mathrm{GRS}_1(\gamma,\mathbf{w})$. If $h\in\mathcal{P}_{n-1}$ then $\mathbf{v}\cdot ev(h)\in\mathrm{GRS}_{n-1}(\gamma,\mathbf{v})=\mathrm{GRS}_1(\gamma,\mathbf{w})^{\perp}$. Therefore $\mathbf{w}\cdot\mathbf{v}\cdot ev(h)=\sum_{i=0}^{n-1}w_iv_ih(\gamma_i)=0$. Let $0\le k\le n-1$. If $f\in\mathcal{P}_k$ and $g\in\mathcal{P}_{n-k}$ then $h=fg\in\mathcal{P}_{n-1}$ and $\mathbf{w}\cdot\mathbf{v}\cdot ev(fg)=\mathbf{w}\cdot ev(g)\cdot\mathbf{v}\cdot ev(f)=0$ which implies $\mathrm{GRS}_k(\gamma,\mathbf{v})^{\perp}\subseteq\mathrm{GRS}_{n-k}(\gamma,\mathbf{w})$. Since $\dim\mathrm{GRS}_k(\gamma,\mathbf{v})^{\perp}=n-k$, $\mathrm{GRS}_k(\gamma,\mathbf{v})^{\perp}=\mathrm{GRS}_{n-k}(\gamma,\mathbf{w})$.

8.3.2 Quantum Reed-Solomon codes (aka polynomial codes)

The CSS construction applies to codes over larger alphabets as well as to binary codes [KKKS06]. Let C_1 and C_2 denote two classical linear codes with parameters $[n,k_1,d_1]_q$ and $[n,k_2,d_2]_q$ such that $C_2^{\perp}\subseteq C_1$. Then there exists an $[[n,k_1+k_2-n,d]]_q$ stabilizer code with minimum distance $d=\min\{\mathrm{wt}(c)\mid c\in(C_1-C_2^{\perp})\cup(C_2-C_1^{\perp})\}$. In the more familiar stabilizer language, $C_1^{\perp}$ corresponds to the X-like generators of the stabilizer and $C_2^{\perp}$ corresponds to the Z-like generators. Elements of C_1 and C_2 that are not in the dual codes corresponding to the logical Z and X operators, possibly multiplied by elements of the stabilizer or by each other.

Aharonov and Ben-Or originally defined quantum polynomial codes [ABO99]. We define the same codes using language from coding theory to clarify the relationship between these codes and the GRS codes. First, choose a GRS code $C_1=\mathrm{GRS}_k(\gamma,\mathbf{1})$ with parameters $[n,k,n-k+1]$ such that γ has no zero components. Choosing γ in this way will guarantee that $C_2^{\perp}$ is also MDS. All the elements of γ must be distinct elements of $\mathbb{F}_q^*$, so $n\le q-1$ since $|\gamma|=n$. Define the code $C_2^{\perp}\subset C_1$ as

$$C_2^{\perp}:=\{(f(\gamma_0),\ldots,f(\gamma_{n-1}))\mid f\in\mathcal{P}_k, f(0)=0\} \tag{8.7}$$

where $k>1$. The parameters of $C_2^{\perp}$ are $[n,k-1,n-k+2]$ so C_2 has parameters $[n,n-k+1,k]$.

Quantum polynomial codes are the CSS codes constructed from these C_1 and C_2. The codes are spanned by

$$|\bar{a}\rangle_k:=\frac{1}{\sqrt{q^{k-1}}}\sum_{f\in\mathcal{P}_k,f(0)=a}|f(\gamma_0)\ldots f(\gamma_{n-1})\rangle \tag{8.8}$$

where $a\in C_1/C_2^{\perp}\cong\mathbb{F}_q$ and the subscript k denotes that the code uses polynomials of degree less than k. These are $[[n,1,\tilde{d}]]_q$ quantum codes with $\tilde{d}=\min\{k,n-k+1\}$. The polynomial codes will saturate the quantum Singleton bound if C_1 has $k = (n+1)/2$. Let C^k Denote a quantum polynomial code that uses polynomials of degree less than k.

The dual codes are defined by a choice of interpolation coefficients. Suppose interpolation coefficients $\mathbf{a} = (a_0, \dots, a_{n-1})$ are chosen such that $\sum_{i=0}^{n-1} a_i f(\gamma_i) = f(0)$ for all $f \in \mathcal{P}_n$, so $a_i = L_i(0)$ for each i. C_2 is a GRS code with parameters $[n, n-k+1, k]_q$ and if $f \in \mathcal{P}_k$ with $f(0) = 0$ then $\sum_{i=0}^{n-1} a_i f(\gamma_i) g(\gamma_i) = (fg)(0) = f(0)g(0) = 0$ for all g such that $\deg(fg) < n$. Therefore, $C_2 = \{(a_0 g(\gamma_0), \dots, a_{n-1} g(\gamma_{n-1})) \mid g \in \mathcal{P}_{n-k+1}\}$. Similarly, $C_1^\perp = \mathrm{GRS}_k(\gamma, \mathbf{1})^\perp = \mathrm{GRS}_{n-k}(\gamma, \mathbf{w})$ where $\mathbf{w}$ satisfies $\sum_{i=0}^{n-1} w_i \gamma_i^j = \delta_{j,n-1}$. The interpolation coefficients satisfy $\sum_{i=0}^{n-1} a_i \gamma_i^j = \delta_{j,0}$, therefore we can choose $w_i = a_i \gamma_i$ since $\gamma_i^n = 1$.

8.3.3 Transversal gates

The polynomial codes admit several transversal gates that we now review [ABO99].

The **generalized Pauli group** $\langle X_c, Z_d, c, d \in \mathbb{F}_{q=p^m}\rangle$ has elements that satisfy the commutation relation $Z_d X_c = \omega^{\mathrm{Tr}\, cd} X_c Z_d$, where ω is a primitive pth root of unity. The generators are defined by $X_c|a\rangle = |a+c\rangle$ and $Z_c|a\rangle = \omega^{\mathrm{Tr}\, ac}|a\rangle$. The **trace function** is defined by

$$\mathrm{Tr}\, \alpha = \mathrm{Tr}_m(\alpha) = \sum_{i=0}^{m-1} \alpha^{p^i} \tag{8.9}$$

for all $\alpha \in \mathbb{F}_q = \mathbb{F}_{p^m}$. If a field $\mathbb{F}_q$ has characteristic p, then $(\alpha+\beta)^p = \alpha^p + \beta^p$ by binomial expansion. This fact can be used to show that $\mathrm{Tr}_m \alpha \in \mathbb{F}_p$ and also the additional properties that (i) it is not identically zero, (ii) $\mathrm{Tr}\, \alpha + \beta = \mathrm{Tr}\, \alpha + \mathrm{Tr}\, \beta$ for all $\alpha, \beta \in \mathbb{F}_{p^m}$, and (iii) $\mathrm{Tr}\, a\alpha = a \,\mathrm{Tr}\, \alpha$ for all $\alpha \in \mathbb{F}_{p^m}$ and $a \in \mathbb{F}_p$.

The logical $\bar{X}_c$ defined by $|\bar{a}\rangle \mapsto |\overline{a+c}\rangle$ for $c \in \mathbb{F}_q$ is transversal,

$$\begin{aligned}
X_c^{\otimes n}|\bar{a}\rangle &= \frac{1}{\sqrt{q^{k-1}}} \sum_{f \in \mathcal{P}_k, f(0)=a} X_c^{\otimes n} |f(\gamma_0), \dots, f(\gamma_{n-1})\rangle \\
&= \frac{1}{\sqrt{q^{k-1}}} \sum_{f \in \mathcal{P}_k, f(0)=a} |f(\gamma_0)+c, \dots, f(\gamma_{n-1})+c\rangle \\
&= \frac{1}{\sqrt{q^{k-1}}} \sum_{g \in \mathcal{P}_k, g(0)=a+c} |g(\gamma_0), \dots, g(\gamma_{n-1})\rangle \\
&= |\overline{a+c}\rangle\,.
\end{aligned}$$

The logical $\bar{Z}_c$ defined by $|\bar{a}\rangle \mapsto \omega^{\mathrm{Tr}\, ac}|\bar{a}\rangle$ for $c \in \mathbb{F}_q$ is also transversal but it is implemented by $\bigotimes_{i=0}^{n-1} Z_{c_i}$ where the $c_i = ca_i \in \mathbb{F}_q$ are interpolation coefficients chosen as before such that $cf(0) = c\sum_{i=0}^{n-1} a_i f(\gamma_i)$ for all $f \in \mathcal{P}_n$. Indeed,

$$\begin{aligned}
(\bigotimes_{i=0}^{n-1} Z_{c_i})\,|\bar{a}\rangle &= \frac{1}{\sqrt{q^{k-1}}} \sum_{f\in\mathcal{P}_k, f(0)=a} (\bigotimes_{i=0}^{n-1} Z_{c_i})\,|f(\gamma_0),\ldots,f(\gamma_{n-1})\rangle \\
&= \frac{1}{\sqrt{q^{k-1}}} \sum_{f\in\mathcal{P}_k, f(0)=a} (\prod_{i=0}^{n-1} \omega^{\mathrm{Tr}\, f(\gamma_i)c_i})\,|f(\gamma_0),\ldots,f(\gamma_{n-1})\rangle\,, \\
&= \frac{1}{\sqrt{q^{k-1}}} \sum_{f\in\mathcal{P}_k, f(0)=a} \omega^{\mathrm{Tr}\sum_{i=0}^{n-1} f(\gamma_i)c_i}\,|f(\gamma_0),\ldots,f(\gamma_{n-1})\rangle\,, \\
&= \frac{1}{\sqrt{q^{k-1}}} \sum_{f\in\mathcal{P}_k, f(0)=a} \omega^{\mathrm{Tr}\, cf(0)}\,|f(\gamma_0),\ldots,f(\gamma_{n-1})\rangle\,, \\
&= \omega^{\mathrm{Tr}\, ac}\,|\bar{a}\rangle\,.
\end{aligned}$$

Since the quantum polynomial codes are CSS, the **SUM gate** $|a,b\rangle \mapsto |a,a+b\rangle$ is transversal as well. The **multiplication gate** S_c defined by $|a\rangle \mapsto |ca\rangle$ is also transversal, as can be shown by direct calculations similar to the $\mathbf{X_c}$ calculation.

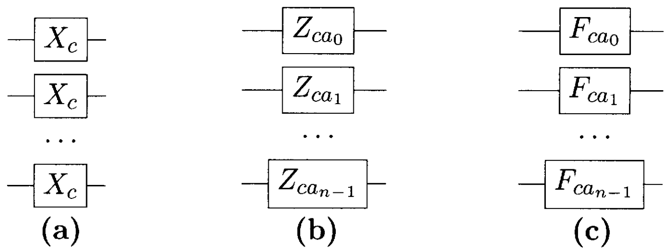


Figure 8-3: Fault-tolerant generalized Pauli and Hadamard gates: (a) $\bar{X}_c$ is transversal, implemented by applying X_c to each qudit (b) $\bar{Z}_c$ is transversal, implemented by applying Z_{ca_i} to the ith qudit (c) $\bar{F}_c$ is code-transforming transversal, implemented by F_{ca_i} on the ith qudit. The $\{a_i\}$ are suitably chosen interpolation coefficients.

8.3.4 Code-transforming transversal gates

Code-transforming transversal gates can be applied transversally to a quantum polynomial code C^k but produce output encoded in a different polynomial code $C^{k'}$ [ABO99]. The Hadamard and Toffoli gates are two such gates.

The **generalized Hadamard gates** F_c are discrete Fourier transforms defined by $|a\rangle \mapsto \frac{1}{\sqrt{q}}\sum_{b\in\mathbb{F}_q} \omega^{\mathrm{Tr}\, abc}\,|b\rangle$. Since $S_c X_a S_c^\dagger = X_{ac}$ and $F_1 X_a F_1^\dagger = Z_a$, $F_1 X_{ac} F_1^\dagger = F_1 S_c X_a S_c^\dagger F_1^\dagger = F_c X_a F_c^\dagger = Z_{ac}$ so $F_c = F_1 S_c$. Consider F_1 and fix interpolation coefficients $\{a_i\}$ such that $f(0) = \sum_{i=0}^{n-1} a_i f(\gamma_i)$

as before. The code-transforming transversal implementation shown in Figure 8-3 is $\bar{F}_1 = \bigotimes_{i=0}^{n-1} F_{a_i}$,

$$
\begin{aligned}
(\bigotimes_{i=0}^{n-1} F_{a_i}) |\bar{a}\rangle &= \frac{1}{\sqrt{q^{k-1}}} \sum_{f\in\mathcal{P}_k, f(0)=a} (\bigotimes_{i=0}^{n-1} F_{a_i}) |f(\gamma_0)\dots f(\gamma_{n-1})\rangle \\
&= \frac{1}{\sqrt{q^n}} \frac{1}{\sqrt{q^{k-1}}} \sum_{b_0,b_1,\dots,b_{n-1}\in\mathbb{F}_q} \sum_{f\in\mathcal{P}_k, f(0)=a} (\prod_{i=0}^{n-1} \omega^{\mathrm{Tr}\, a_i f(\gamma_i) b_i}) |b_0 b_1 \dots b_{n-1}\rangle \\
&= |\beta\rangle + \left|\beta^\perp\right\rangle.
\end{aligned}
$$

$|\beta\rangle$ is the superposition of those $|b_0\dots b_{n-1}\rangle$ with $\deg b(x) \le n-k$, i.e. $b(x)\in\mathcal{P}_{n-k+1}$, and $\left|\beta^\perp\right\rangle$ is the superposition of those $b(x)\in\mathcal{P}_n - \mathcal{P}_{n-k+1}$. If $b(x)\in\mathcal{P}_{n-k+1}$ then $h(x) := f(x)b(x)\in\mathcal{P}_n$ so $\sum_{i=0}^{n-1} a_i f(\gamma_i) b(\gamma_i) = \sum_{i=0}^{n-1} a_i h(\gamma_i) = h(0) = f(0)b(0)$. This gives

$$
\begin{aligned}
|\beta\rangle &= \frac{1}{\sqrt{q^{n+k-1}}} \sum_{b(x)\in\mathcal{P}_{n-k+1}} \sqrt{q^{2(k-1)}} \omega^{\mathrm{Tr}\, ab(0)} |b(\gamma_0)\dots b(\gamma_{n-1})\rangle \\
&= \frac{1}{\sqrt{q^{n-k+1}}} \sum_{b\in\mathbb{F}_q} \omega^{\mathrm{Tr}\, ab} \sum_{b(x)\in\mathcal{P}_{n-k+1}, b(0)=b} |b(\gamma_0)\dots b(\gamma_{n-1})\rangle \\
&= \frac{1}{\sqrt{q}} \sum_{b\in\mathbb{F}_q} \omega^{\mathrm{Tr}\, ab} \left|\bar{b}\right\rangle_{n-k+1},
\end{aligned}
$$

so $|\beta\rangle$ is encoded in a code using polynomials of degree less than $n-k+1$, i.e. in the code spanned by C_2. Because $\langle\beta|\beta\rangle = 1$ (since it is the Fourier transform of a unit vector), $\left|\beta^\perp\right\rangle = 0$ and $\bar{F}_1 |\bar{a}\rangle_k = \frac{1}{\sqrt{q}} \sum_{b\in\mathbb{F}_q} \omega^{\mathrm{Tr}\, ab} \left|\bar{b}\right\rangle_{n-k+1}$. Therefore $\bar{F}_1$ is a code-transforming transversal gate that takes inputs in C^k and produces outputs in C^{n-k+1}. The distance of the output code is equal to the distance of the input code, and the codes are. actually the same if $k = n-k+1$, i.e. if $C_1 = C_2$ and the CSS code is dual-containing. The $\bar{F}_c$ gate can be implemented as $\otimes_{i=0}^{n-1} F_{ca_i}$, which is a transversal multiplication operator $\bar{S}_c$ followed by $\bar{F}_1$.

The **generalized Toffoli** $\mathrm{TOF}|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|ab+c\rangle$ is also code-transforming transversal. By direct calculation,

$$
\begin{aligned}
\mathrm{TOF}^{\otimes n} |\bar{a}\rangle_k \left|\bar{b}\right\rangle_k |\bar{c}\rangle_k &= |\bar{a}\rangle_k \left|\bar{b}\right\rangle_k \frac{1}{\sqrt{q^{k-1}}} \sum_{h'\in\mathcal{P}_{2(k-1)+1}, h'(0)=ab+c} |h'(\gamma_0)\dots h'(\gamma_{n-1})\rangle \\
&= |\bar{a}\rangle_k \left|\bar{b}\right\rangle_k \left|\bar{a}\bar{b}+\bar{c}\right\rangle_{2k-1}.
\end{aligned}
$$

Therefore, $T\bar{O}F$ is implemented by applying TOF to corresponding qudits of three blocks. The input and output codes of the first two blocks correspond but the third block takes inputs in C^k

to outputs in C^{2k-1}. The distance of the third output code block increases, stays the same, or decreases if $k < \frac{n+2}{3}$, $k = \frac{n+2}{3}$, or $\frac{n+2}{3} < k \leq n-k+1$, respectively. Therefore, it is not possible for quantum polynomial codes to have transversal Hadamard gates and simultaneously have a code-transforming Toffoli gate whose third input and output blocks are encoded in codes with the same minimum distance.

8.3.5 Fault-tolerant degree reduction

In the setting of quantum polynomial codes, **degree reduction** refers to the process of mapping a codeword of $C^{k'}$ to the corresponding codeword of C^k where $k' > k$. Aharonov and Ben-Or gave fault-tolerant degree reduction methods in their original work [ABO99]. Teleportation is an important primitive in fault-tolerant quantum computing that allows for converting between quantum codes [GC99]. This gives a direct means of implementing fault-tolerant degree reduction. The circuit is shown in Figure 8-4 [CGS02]. Following the state through the circuit,

$$\begin{aligned}
|\bar{a}\rangle_{k'} \otimes F_1 |\bar{0}\rangle_k &= \frac{1}{\sqrt{q}} \sum_{b\in\mathbb{F}_q} |\bar{a}\rangle_{k'} |\bar{b}\rangle_k \\
&= \frac{1}{\sqrt{q}} \sum_{b\in\mathbb{F}_q} \left[\frac{1}{\sqrt{q^{k'-1}}} \sum_{f\in\mathcal{P}_{k'}, f(0)=a} |ev(f)\rangle \right] \cdot \left[\frac{1}{\sqrt{q^{k-1}}} \sum_{g\in\mathcal{P}_k, g(0)=b} |ev(g)\rangle \right] \\
&\mapsto \frac{1}{\sqrt{q}} \sum_{b\in\mathbb{F}_q} \left[\frac{1}{\sqrt{q^{k'-1}q^{k-1}}} \sum_{f\in\mathcal{P}_{k'}, f(0)=a, g\in\mathcal{P}_k, g(0)=b} |ev(f+g)\rangle |ev(g)\rangle \right] \\
&\mapsto \frac{1}{\sqrt{q^{k-1}}} \sum_{g\in\mathcal{P}_{k-1}, g(0)=b} |d_0 \ldots d_{n-1}\rangle |g(\gamma_0) \ldots g(\gamma_{n-1})\rangle \\
&\mapsto \frac{1}{\sqrt{q^{k-1}}} \sum_{g\in\mathcal{P}_{k-1}, g(0)=(a+b)-b=a} |g(\gamma_0) \ldots g(\gamma_{n-1})\rangle = |\bar{a}\rangle_k
\end{aligned}$$

The controlled-SUM gate is a logical gate because $\mathcal{P}_k \subseteq \mathcal{P}_{k'}$. Therefore the resulting codeword in the first register is a codeword of $C^{k'}$. Measuring the first register in the computational basis fixes the set of outcomes $\{d_i\}, d_i = f(\gamma_i) + g(\gamma_i)$. Taking the Euclidean inner product with the interpolation coefficients for $\mathcal{P}_{k'}$ and $\{\gamma_i\}$ gives $d = \sum_{i=0}^{n-1} a_i d_i = f(0) + g(0) = a + b$, indicating the appropriate correction operation.

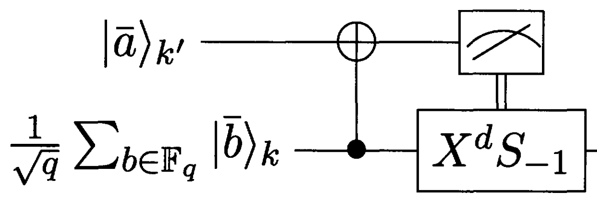

$|\bar{a}\rangle_{k'}$
$\frac{1}{\sqrt{q}}\sum_{b\in\mathbb{F}_q}|\bar{b}\rangle_k$
X^dS_{-1}

Figure 8-4: Fault-tolerant degree reduction through quantum teleportation. The input state is a codeword encoded using a polynomial code based on polynomials in $\mathcal{P}_{k'}$ (shown here as a basis state $|\bar{a}\rangle_{k'}$). The ancilla state is encoded in a polynomial code based on polynomials in $\mathcal{P}_k$ where $k < k'$. The controlled-SUM gate is a valid gate when the target has degree greater than or equal to the control degree. The computational basis measurement outcomes can be used to infer the value of d in the generalized correction operator.

8.3.6 Fault-tolerant error-correction

The polynomial codes are CSS codes, so Steane's error-correction method applies as shown in Figure 8-5. The logical states used are encoded by a standard network [GRB03].

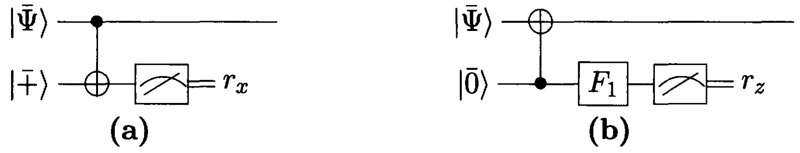


Figure 8-5: Steane syndrome extraction networks for CSS qudit codes: (a) bit-shift syndrome extraction network (b) phase-shift syndrome extraction network derived by conjugating the input of the bit-shift network by F_1. We can infer the locations and types of errors from the measurement outcomes $r_x \in C_1$ and $r_z \in C_2$ by decoding the outcomes as the appropriate classical Reed-Solomon codes.

8.3.7 Qudits as products of systems

It may be desirable to decompose each qudit into a product of subsystems. The Hilbert space $\mathcal{H}_q$ of a q-qudit is a q-dimensional space spanned by basis vectors that we label by elements of a finite field, i.e. $\mathcal{H}_q = \mathrm{span}_{x\in\mathbb{F}_q}|x\rangle$ and $q = p^m$ for some m. We would like to decompose $\mathcal{H}_q$ into a tensor product of Hilbert spaces $\mathcal{H}_r$. Let $r = p^\ell$ be the size of a subfield of $\mathbb{F}_q$, where $\ell|m$, so that there is a natural Pauli group on each $\mathcal{H}_r$.

In this case, each gate on the original qudits must be decomposed into gates on the new subsystems. Choosing a basis of $\mathbb{F}_q$ over $\mathbb{F}_r$ defines a (vector space) isomorphism B from $\mathbb{F}_q$ to $\mathbb{F}_r^{m/\ell}$ that relabels the standard basis of $\mathcal{H}_q$ like $|x\rangle \mapsto |B(x)\rangle \in \mathcal{H}_r^{\otimes(m/\ell)}$. Let $P(c)$ denote $P_{c_1}\otimes\cdots\otimes P_{c_{m/\ell}}$ for $c \in \mathbb{F}_r^{m/\ell}$. B also induces a relabeling of the generalized Pauli group by $X_c \mapsto X(B(C))$ and

$Z_c \mapsto Z(y_c)$ where $y_c \in \mathbb{F}_r^{m/\ell}$ satisfies

$$\mathrm{Tr}_q(cx) = \mathrm{Tr}_r\left(\sum_{i=1}^{m/\ell}(y_c)_i(B(x))_i\right) = \mathrm{Tr}_r(y_c \cdot B(x)), \tag{8.10}$$

for all $x \in \mathbb{F}_q$ so that the mapping is a group isomorphism.

When constructing fault-tolerant gates, a basis that gives a simple representation is desirable. One particularly simple representation is where the Fourier transform $F_{1,(q)}$ on a q-qudit decomposes into m/ℓ Fourier transform gates $F_{1,(r)}$ on the r-qudits. Since $F_1 X_c F_1^\dagger = Z_c$, the condition $F_{1,(q)} = F_{1,(r)}^{\otimes(m/\ell)}$ is equivalent to $y_c = B(c)$. Therefore, B should satisfy

$$\mathrm{Tr}_q(xy) = \mathrm{Tr}_r(B(x) \cdot B(y)),\ \forall x, y \in \mathbb{F}_q. \tag{8.11}$$

If gates are ultimately implemented on qubits, consider an example where $q = 2^m$ so that $\mathbb{F}_q$ has a basis over $\mathbb{F}_2$. Taking the canonical basis $\{1, \omega, \ldots, \omega^{m-1}\}$, the associated $B : \mathbb{F}_{2^m} \longrightarrow \mathbb{F}_2^m$ takes field elements to their vector representation over $\mathbb{F}_2$. We are free to choose to represent X_c as $X(B(c))$ for each $c \in \mathbb{F}_q$. There is an operator $Z(y_c)$ that has the same action as Z_c for each $c \in \mathbb{F}_q$; the vector $y_c \in \mathbb{F}_2^m$ must be such that $Z_c\,|x\rangle = Z(y_c)\,|B(x)\rangle$. This is true if $y_c \cdot B(x) = \mathrm{Tr}\, cx$ for all $x \in \mathbb{F}_{2^m}$, where the multiplication on the left hand side is a dot product of binary vectors, and the multiplication on the right hand side is multiplication in $\mathbb{F}_{2^m}$. The vector y_c exists for each $c \in \mathbb{F}_{2^m}$ by dimensionality.

Table 8.2 gives examples of this decomposition for Pauli operators over small fields of size 2^m for $m = 2, 3, 4$. The table gives the generators of the Pauli group that satisfy the correct commutation relations. For example, in $\mathbb{F}_4$, $X_1 Z_1 = (e^{i\pi/2})^{\mathrm{Tr}\ 1} Z_1 X_1 = (e^{i\pi/2})^0 Z_1 X_1$ in $\mathbb{F}_4$, and this is observed in the Table since $[XI, IZ] = 0$.

The SUM gate corresponds to a transversal CNOT because the field has characteristic 2. However, the Hadamard gates are not necessarily transversal. We know that $F_c X_a F_c^\dagger = Z_{ca}$, so the decomposition of Hadamards into gates on qubits must be a Clifford circuit. This Clifford circuit must exchange $X(B(a))$ and $Z(y_{ca})$, so we can proceed in two stages. In the first stage, we apply a circuit P_c consisting of CNOT gates and Pauli X's such that $P_c X(B(a)) P_c^\dagger = X(y_{ca})$. The second stage is a transversal Hadamard gate on the block of m qubits. Examples of some decompositions are shown in Figure 8-6.

There is no basis for which all of the Hadamards F_c are simply qubit relabelings followed by

Field	Element c	X(B(c))	Z($\mathbf{y_c}$)
$\mathbb{F}_4$	1	XI	IZ
	ω	IX	ZZ
$\mathbb{F}_8$	1	XII	ZII
	ω	IXI	IIZ
	ω^2	IIX	IZI
$\mathbb{F}_{16}$	1	XIII	IIIZ
	ω	IXII	IIZI
	ω^2	IIXI	IZII
	ω^3	IIIX	ZIIZ

Table 8.2: Some decompositions of Pauli operators corresponding to the field $\mathbb{F}_{2^m}$ generated by $\omega^m + \omega + 1 = 0$. The canonical basis $\{1, \omega, \ldots, \omega^{m-1}\}$ is used to represent each field element, and this basis determines the relationship between the field elements and their X representations. The Z representations are derived based on the relationship $y_c \cdot B(x) = \operatorname{Tr} cx \ \forall x \in \mathbb{F}_q$.

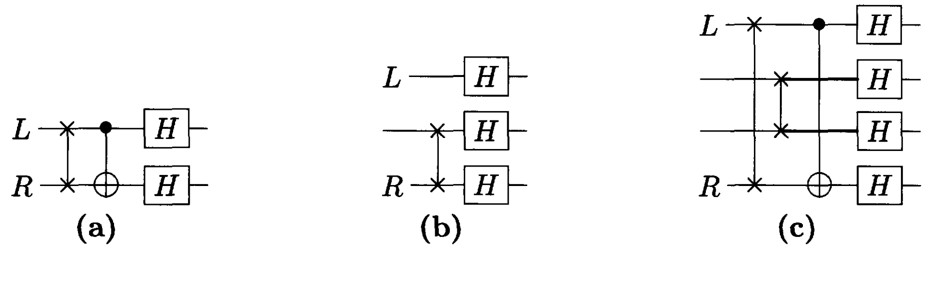


Figure 8-6: Fourier transform gates F_1 implemented on 2^m level systems that have been decomposed into qubits in the canonical basis for $m = 2, 3, 4$, respectively. The qubits are ordered from top to bottom in the circuits as they occur in Table 8.2 from left to right. The thicker lines in (c) indicate that the qubits must wait during the controlled-NOT gate.

transversal Hadamard gates. To see this, choose an arbitrary B such that F_1 is transversal up to relabeling. The set of Hadamards is $F_1 S_c$ where S_c is the constant multiplication gate. There is a Hadamard taking X_i to Z_j for each nonzero i and j. Therefore, at least one such Hadamard must change the weight of an operator, i.e. $\exists F_c$ such that for some field element a we have $\mathrm{wt}\ X(B(a)) \neq \mathrm{wt}\ Z(y_{ca})$.

However, it is possible to choose the GRS codes such that the logical Fourier transform $\bar{F}_1$, used in error-correction, decomposes into nm transversal Hadamards. $\bar{F}_1$ is implemented by F_{a_i} for each of the interpolation coefficients $\{a_i\}$. First, we would like to implement $\mathbf{F_1}$ as n transversal F_1 gates. Recall that the interpolation coefficients are chosen such that $\sum_{i=0}^{n-1} a_i f(\gamma_i) = f(0)$ for all $f \in \mathcal{P}_n$. Expanding $f(x)$, this condition becomes

$$\sum_{i=0}^{n-1} a_i \gamma_i^j = \delta_{j0} \tag{8.12}$$

where δ_{j0} equals one if $j = 0$ and zero otherwise. Equation 8.12 is satisfied iff $\gamma_i = \omega^i$ for a primitive element ω, since $n \equiv 1 \bmod 2$ when $j = 0$ and the field has characteristic two. If we choose $\gamma_i = \omega^i$ for a primitive element ω and take $n = 2^m - 1$ then $\sum_{i=0}^{n-1} \omega^i = 0$ and $\sum_{i=0}^{n-1} \omega^{ji} = 0$. Codes with these properties are the narrow-sense Reed-Solomon codes, which happen to be self-orthogonal BCH codes.

Next, we would like to find a basis of $\mathbb{F}_{2^m}$ such that F_1 can be implemented by m transversal Hadamards. Therefore, the basis should satisfy $B(x) \cdot B(y) = \mathrm{Tr}\, xy$ for all x and y in the field. This is a bilinear equation so we need only satisfy the condition for an orthogonal basis. For such a basis, $B(x) \cdot B(y) = \delta_{xy}$, so choose elements such that $\mathrm{Tr}\, xy = \delta_{xy}$. The basis elements are such that $\mathrm{Tr}\, x^2 = 0$ and $\mathrm{Tr}\, xy = 1$. This is what is known as a self-dual basis [JMV90, GGB99]. For example, $\{\omega, \omega^2\}$ where $\omega^2 + \omega + 1 = 0$ is a self-dual basis of $\mathbb{F}_4$, $\{\omega^3, \omega^6, \omega^5\}$ where $\omega^3 + \omega + 1 = 0$ is a self-dual basis of $\mathbb{F}_8$, $\{\omega^3, \omega^7, \omega^{13}, \omega^{12}\}$ where $\omega^4 + \omega + 1 = 0$ is a self-dual basis of $\mathbb{F}_{16}$, and $\{\omega^3, \omega^{20}, \omega^{13}, \omega^{12}, \omega^{26}\}$ where $\omega^5 + \omega^2 + 1 = 0$ is a self-dual basis of $\mathbb{F}_{32}$.

Therefore, we have understood that the quantum codes constructed from narrow-sense primitive non-binary BCH codes are contained in the family of polynomial codes. They are MDS codes like the other polynomial codes and they are self-orthogonal so the Fourier transform is transversal. By choosing a self-dual basis of the field $\mathbb{F}_{2^m}$, each Fourier transform as a tensor product of Hadamards.

Concatenated quantum Reed-Solomon codes have been constructed in the literature [GGB99]. If a basis for the field $\mathbb{F}_{2^m}$ is chosen such that two codewords $b, c \in \mathbb{F}_{2^m}^n$ orthogonal in the Euclidean inner product, $\sum_{i=0}^{n-1} b_i c_i = 0$, remain orthogonal over $\mathbb{F}_2^{mn}$, $\sum_{i=0}^{mn-1} B(b)_i B(c)_i = 0$, then this corresponds to choosing a self-dual basis of the field. This choice of basis corresponds to an $\mathbb{F}_2$-linear mapping from $\mathbb{F}_{2^m}$ into the $[m, m, 1]_2$ code $\{0, 1\}^m$ [HP03]. It allows to construct $[[mn, mk, d' \geq d]]_2$ quantum codes from $[n, k, d]_{2^m}$ self-orthogonal classical codes. Therefore, the arguments of the previous section led to a concatenated code construction with a simple (poor) inner code. It is quite possible that different choices of bases and/or inner codes produce better quantum codes, or even that subfield constructions based on Reed-Solomon codes produce binary quantum codes with desired parameters [BE97].

To summarize, we have understood that the quantum codes constructed from narrow-sense primitive non-binary BCH codes are contained in the family of polynomial codes. They are MDS codes like the other polynomial codes, and they are self-orthogonal so the Fourier transform is transversal. By choosing a self-dual basis of the field $\mathbb{F}_{2^m}$, the Fourier transform is a tensor product of Hadamards and the code is a concatenated code with an $[[m, m, 1]]$ inner code.

8.4 Detailed constructions and circuits

This section provides many details about the fault-tolerant circuit constructions for the codes in Table 8.1. Specifically, the section explains how to construct the syndrome decoders, $[[5, 1, 3]]$ encoder, polynomial code circuits, and Bacon-Shor verifiers.

8.4.1 Syndrome decoding procedures

Best known general algorithms for constructing the classical circuits to decode measurement outcomes obtained in Steane error correction require exponential time and/or space. Therefore, we consider each code's syndrome decoder separately, essentially finding a special-purpose algorithm for each to make the decoding feasible.

Table 8.3 lists all of the codes we consider in this study and their syndrome decoders. There are six distinct decoding algorithms that we use to compute the error locations and type of error from the syndrome measurements: a generic table lookup algorithm, a table lookup algorithm for cyclic codes over arbitrary fields, a majority voting algorithm for Bacon-Shor codes, a minimum weight matching algorithm for surface codes, a simple message passing algorithm for the concatenated Hamming code, and an algebraic decoder for the $[[47, 1, 11]]$ quadratic residue code.

Rather than use a general table-lookup algorithm, we use a so-called Meggitt decoder which

CODE	DECODER
$[[5,1,3]]$	Table Lookup
$[[7,1,3]]$	Table Lookup (cyclic)
$[[9,1,3]], [[25,1,5]], [[49,1,7]], [[81,1,9]]$	Majority
$[[15,1,3]]$	Table Lookup
$[[13,1,3]], [[41,1,5]], [[85,1,7]]$	Min. Wt. Matching
$[[21,3,5]]$	Table Lookup (cyclic)
$[[23,1,7]]$	Table Lookup (cyclic)
$[[47,1,11]]$	Algebraic [CTC+07]
$[[49,1,9]]$	Table Lookup with Message Passing
$[[60,4,10]]$	Table Lookup (cyclic)

Table 8.3: The decoders that we use for the codes in our study.

uses the fact that the polynomial codes and the Hamming, Golay, and quadratic residue (QR) codes are constructed from cyclic classical codes. Cyclic codes have a compact description in terms of a generating polynomial whose coefficients give one of the code words and whose cyclic shifts generate a basis for the code. The Meggitt decoding algorithm stores a table of syndromes and their associated error corrections [HP03]. For non-binary codes such as the polynomial codes, the table stores both error locations and error-type (the so-called amplitude). Only $\binom{n}{w-1}$ syndromes need to be stored for a weight w error, since one of the coordinates can be fixed by the cyclic symmetry. Finding the appropriate recovery requires at most n table lookups. If we fail to find a recovery in the table, a subroutine is triggered that applies some syndrome-dependent correction mapping the state back into the code space.

For cyclic codes with larger distance where table lookup is impractical, for example $[[47,1,11]]$, algebraic decoding techniques can be used. The generator polynomial's roots are used to compute a sequence of syndromes from the received vector from which we can locate errors. BCH codes are easy to decode because their generator polynomials have a contiguous sequence of roots so the Berlekamp-Massey algorithm can find the error-locator polynomial whose roots give the error locations. Sometimes decoding up to the full minimum distance of the code is challenging because the generator polynomial may not have a long sequence of roots, so some syndromes are missing and the Berlekamp-Massey algorithm cannot be directly applied. In this case, unknown syndromes can sometimes be computed from algebraic equations involving the known syndromes. Algebraic decoding of the $[[47,1,11]]$ proceeds this way. For each error weight from zero to t, we compute any missing syndromes, construct a polynomial whose roots are the error locations, and find the roots of the polynomial. If the polynomial has enough roots, we correct those errors and stop. If we do not find enough roots for each of the locators, we return a "failed" result, triggering a subroutine that applies some syndrome-dependent correction that maps the state to some (possibly logically incorrect) state in the code space. The implementation details can be found in [CTC+07].

The Bacon-Shor codes are essentially concatenated quantum repetition codes. Since the code stabilizer is preserved by bitwise Hadamard composed with a 90 degree rotation of the square lattice, one syndrome decoder is sufficient for both X and Z error correction. Imagine a vector of n^2 syndrome bits placed on an n by n square lattice. Let s_x be the syndrome vector for X errors and s_z be the syndrome vector for Z errors. Let R be the map on vectors of length n^2 that rotates them by 90 degrees on the square lattice. The same syndrome decoder is applied to s_x and Rs_z. The syndrome decoder decodes a variation on the classical repetition code on n bits. First, the decoder computes the parity of each column of the lattice and stores each column parity as an element of a vector p. Next, the decoder computes the repetition code parity check $h = Hp$. This parity check H is expressed in standard form $[I_{n-1}\ 1]$ where 1 is the all ones column vector. Finally, the decoder infers the error locations from the parity check. If the weight of the parity check is greater than t, we must assume that the rightmost bit of p was incorrect so that $h \oplus 1$ gives the error locations on the first $n-1$ bits of p. Otherwise, we infer that the rightmost bit of p was correct so that h gives the error locations on the first $n-1$ bits of p.

The surface code is decoded using Edmond's minimum-weight matching algorithm. The approach differs slightly depending on whether Steane-EC or Shor-EC is used but is essentially the same as [DKLP02]. Steane-EC gives a 2D matching problem whereas Shor-EC gives a 3D matching problem. The mapping from syndrome information to a matching problem is as follows.

Nonzero syndrome bits are called defects and are located somewhere in the $\ell \times \ell$ plane. We construct a complete weighted graph whose vertices represent defects and whose edge weights indicate the distance between defects. The surface code's syndrome may be such that there are lone defects which are not caused by error patterns connecting two defects, but by an error pattern connecting an edge-defect on the boundary to an inner defect. X and Z errors constitute separate matching problems and X-defects can be matched with, say, the horizontal boundaries and Z-defects with the vertical boundaries.

We can design an algorithm for decoding the surface code for, say, Z errors, as follows:

- Imagine cutting the lattice vertically in two halves, left (L) and right (R). Let $N_{L/R}(i)$ be the number of defects in row i of the left/right part of the lattice. For each row of the lattice, add a $N_{L/R}(i)$ edge defects on the ith row on the left (right) boundary.

- Assign weight of the edges between *any* edge defects as zero and assign the distance as the weight between edge defects and inner defects.
- Compute the minimum-weight perfect matching of the graph of defects.
- The recovery operation consists of applying phase flips on the qubits that are along the edges of each pair of matched vertices in the graph.

Note that the algorithm enforces the property that the graph has an even number of vertices, so that every vertex can be matched.

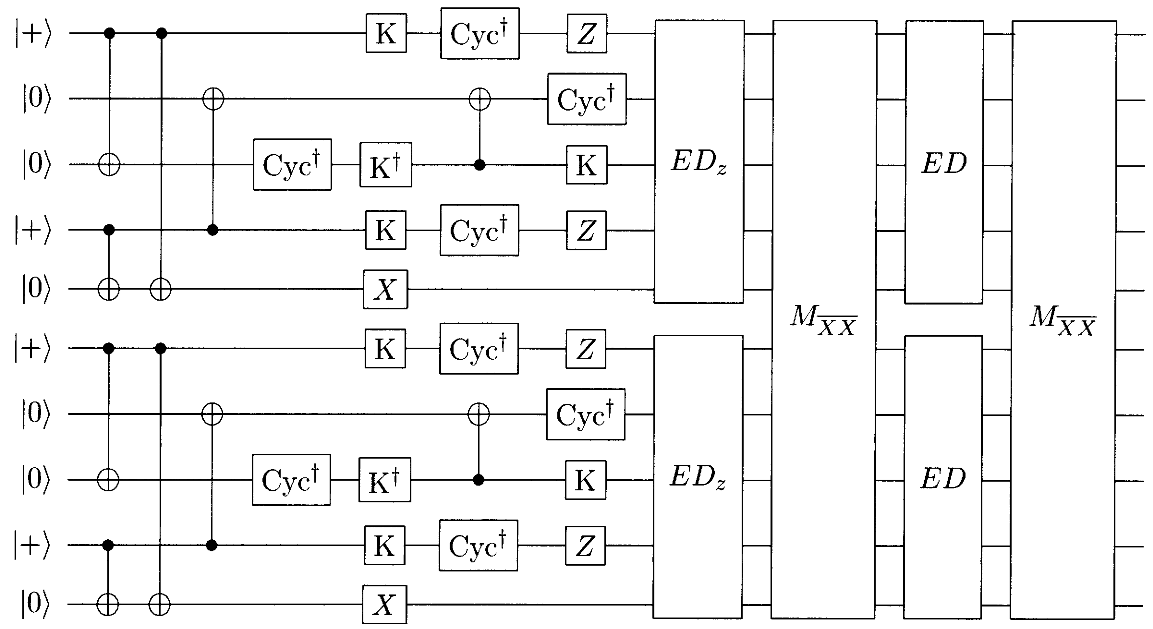


Figure 8-7: A fault-tolerant circuit for preparing logical Bell pairs for Knill error correction of [[5, 1, 3]]. The sub-circuit ED_z measures the stabilizer of $|\overline{0}\rangle$, $\langle XZZXI, IXZZX, XIXZZ, ZXIXZ, ZZZZZ\rangle$, using 4 and 5 qubit cat states, and the sub-circuit ED makes the same measurement without measuring $\overline{Z} = ZZZZZ$. If any measurement outcome is nonzero, the Bell state is rejected. The sub-circuit $M_{\overline{XX}}$ measures $\overline{XX} = X^{\otimes 10}$ using a 10 qubit cat state. The Bell state is rejected if the $\overline{XX}$ measurements disagree, but if they are both 1 then $\overline{Z}_1$ is applied to the output Bell pair. The cat states are verified so that if a cat state is accepted then a single fault in its preparation cannot produce a correlated error.

The concatenated [[7,1,3]] code, that is, the [[49,1,9]] code, can be decoded to distance 7 if we treat it as a concatenated code. However, decoding the code to distance 9 requires a slight modification of the algorithm so that a simple message is passed from level-1 to level-2.

Suppose the 49 transversal measurement outcomes are organized into 7 registers of 7 bits each. We use these registers as temporary storage to compute the appropriate correction. First, we compute the level-1 syndromes for each register as we would normally do. These syndromes indicate errors e_i in the ith level-1 register. We correct each level-1 register according to the e_is and "flag" those registers for which $e_i \neq 0$. Next, we compute the level-2 (logical) syndrome of the resulting 49 bit register, which now has trivial level-1 syndrome in each 7 bit register. This level-2 syndrome indicates a logical correction $\bar{e}$ that is constant on each level-1 register (but two level-1 registers can take different values). The correction $c_1 := (\bigoplus_i e_i) \oplus \bar{e}$ corrects all errors of weight 4 or less, except for one problem case. This case occurs when a pair of errors occurs in one level-1 register and another pair of errors occurs in a different level-1 register. The problem is overcome by comparing the register positions where $\bar{e}$ is 1 with the positions of the flags whenever two flags are raised. If they disagree, apply the correction $c_2 := (\bigoplus_i e_i) \oplus \bar{f}$ where $\bar{f}$ is a logical correction on the flagged registers. Otherwise, apply the original correction c_1. This procedure corrects all errors of weight 4 or less and returns the input to the codespace in all cases.

All decoding algorithms have been tested exhaustively for the codes in this chapter and are found to correct all errors of weight t or less.

8.4.2 The [[5, 1, 3]] code

We want to find an encoding circuit for $|\bar{0}\rangle$ for the [[5, 1, 3]] with the minimum number of 2-qubit gates. The first step is to row reduce the stabilizer

$$S_{|\bar{0}\rangle} = \left[\begin{array}{c|ccccc|ccccc} + & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ + & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ + & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ + & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{c|ccccc|ccccc} - & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ + & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ + & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ - & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right]. \quad (8.13)$$

Applying H_5 brings all X's along the diagonal, and Z_1Z_4 clear the negative signs. The stabilizer $S_{Z_1Z_4H_5|\bar{0}\rangle}$ now corresponds to a graph state shown in Figure 8-8.

Local complementation of vertices 1 and 4 reduces the edge count by 2, leaving a ring graph. **Local complementation** of a vertex v applies the Clifford $Q_v \otimes \bigotimes_{v' \in N(v)}(Z_{v'}K_{v'})$ where Q : $(X, Y, Z) \rightarrow (X, Z, -Y)$. Therefore, the result of applying the local unitary

$$U := (Q_1Z_1)(Z_2K_2)(Z_3K_3)(Q_4Z_4)(Z_5K_5Z_5K_5H_5) \quad (8.14)$$

to $|\bar{0}\rangle$ is the state described by the ring graph on vertices ordered 1, 3, 2, 4, 5. Exhaustive search

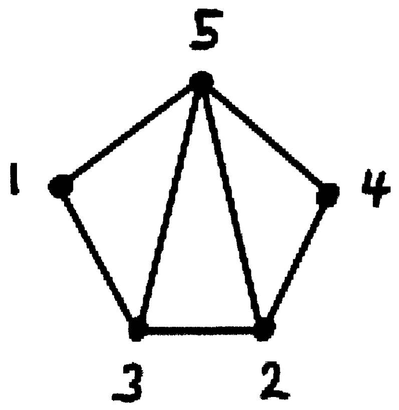


Figure 8-8: Graph for $|\bar{0}\rangle$ of 5 qubit code.

over local complementations shows that this is the minimum number of edges (5).

We write the local unitary in terms of a fault-tolerant set of gates Q, CYC for the [[5, 1, 3]] using the identities $Q_1 Z_1 = -Y_1 Q_1$, $Z_5 K_5 Z_5 K_5 H_5 = Z_5 H_5$, $ZK = K^\dagger$, $ZH = K \cdot \mathrm{CYC}^\dagger$, $H = K^\dagger \cdot \mathrm{CYC}^\dagger$ and $Q = K^\dagger \cdot \mathrm{CYC}$ where $CYC : X \to Y \to Z \to X$. The fault-tolerant set Q and CYC are written $Q = HKH$ and $CYC = KHKH = KQ$ terms of standard Clifford group generators. Figure 8-7 is an example of one possible circuit with 4 CNOT gates per $|\bar{0}\rangle$.

8.4.3 Polynomial codes

Polynomial codes constructed from codes C_1 that are narrow-sense Reed-Solomon codes, i.e. over $\mathbb{F}_q$ with $q = n + 1$, whose block size is $n = 2^m - 1$ and whose designed distance is $\delta = (n+1)/2$ will have transversal syndrome extraction circuits on qubits (except for the ancilla encoders). The codes are self-orthogonal and cyclic with some generator polynomial.

Rather than view these codes as cyclic codes, there is a simple way to view polynomial codes so as to construct encoders for them. The matrix for the evaluation map on $\mathcal{P}_k$ is a generator matrix for C_1

$$G = \left[\begin{array}{cccc} 1 & 1 & \dots & 1 \\ \hline \gamma_0 & \gamma_1 & \cdots & \gamma_{n-1} \\ \dots & & & \dots \\ \gamma_0^{k-1} & \gamma_1^{k-1} & \cdots & \gamma_{n-1}^{k-1} \end{array}\right]. \tag{8.15}$$

The first $k-1$ rows of these generator matrices generate $C_2^\perp$ and the last row of these generator matrices is a coset representative in $C_1/C_2^\perp$, so the matrices are already in a standard form to construct encoders for the quantum code [GRB03]. The generator for $C_2^\perp$ is obtained by deleting the all-ones row. For the narrow-sense RS family, $C_1 = C_2$, so we do not need additional generator matrices and $\gamma_i = \alpha^i$ for some primitive element α

Example $[[7,1,4]]_8$ **or** $[[21,3,5]]_2$

This code is the first in this family that can correct one error. It is a $[7,4,4]_8$ code with $(n,\delta,q) = (7,4,8)$ generated by $x^3+\beta^6x^2+\beta x+\beta^6$ where $\beta^3+\beta+1=0$, $\beta \in \mathbb{F}_8$. This code yields a $[[7,1,4]]_8$ polynomial code or a $[[21,3,\geq 4]]_2$ binary concatenated code.

The $[[7,1,4]]_8$ code has generator matrix

$$G_{[[7,1,4]]_8} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & \beta^6 & \beta^5 & \beta^5 & \beta^2 \\ 0 & 1 & 0 & \beta & \beta^2 & \beta^4 & 1 \\ 0 & 0 & 1 & \beta^6 & \beta^6 & \beta^3 & \beta \\ \hline 0 & 0 & 0 & 1 & \beta^2 & 1 & \beta \end{array}\right] \tag{8.16}$$

where $\beta^3+\beta+1=0$. The qudit order is not changed when moving from the generator polynomial to this generator matrix representation of the code. Using the self-dual basis $B = \{\beta^3,\beta^6,\beta^5\}$, in which $B(\beta) = 011$ for example, we can construct the following matrix representation of $\mathbb{F}_8$:

$$\beta \rightarrow \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \qquad \beta^2 \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\beta^3 \rightarrow \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \qquad \beta^4 \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$\beta^5 \rightarrow \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \beta^6 \rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Using this representation, we can decompose the qudit gates in the encoder shown in Figure 8-9 into products of qubit gates by finding a circuit for a generator and simplifying the powers of this generator. Figure 8-10 shows the complete set of rules for transforming CNOT circuits that we would use to simplify the powers of the generator [IY03]. Let the first bit be the coefficient of β^3, the second be the coefficient of β^6, and the third be the coefficient of β^5. The multiplication gates are given by

$$\begin{aligned}
S_\beta &= SWP[2,3]CNOT[1,3]CNOT[2,1] & S_{\beta^2} &= SWP[1,3]CNOT[2,1]CNOT[3,2] \\
S_{\beta^3} &= SWP[1,2]CNOT[2,3]CNOT[3,1] & S_{\beta^4} &= SWP[1,2]CNOT[3,2]CNOT[1,3] \\
S_{\beta^5} &= SWP[1,3]CNOT[1,2]CNOT[2,3] & S_{\beta^6} &= SWP[2,3]CNOT[3,1]CNOT[1,2]
\end{aligned}$$

where the gates are applied from right to left as they appear in these expressions and $CNOT[c,t]$ means that c is the control bit and t is the target bit. Notice that $\beta^4, \beta^5, \beta^6$ are the Hadamard conjugates of β^3, β^2, β. This discussion implies that the SUM gates in the decoder decompose into a total of 45 CNOT gates (3 CNOT gates each), and the multiplications gates decompose into 32 CNOT gates (2 CNOTs and a SWAP each).

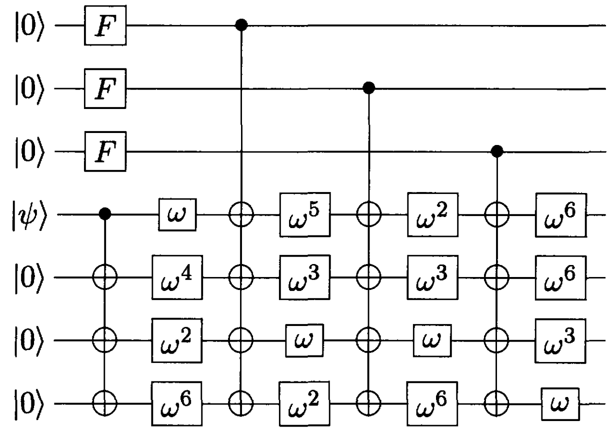


Figure 8-9: Encoder for the $[[7,1,4]]_8$ code expressed using gates over eight level systems. This encoder is found directly from the generator matrix of C_1 using known methods. The encoder in this figure has been simplified by combining multiplication gates. The SUM gates decompose into 45 CNOT gates and the multiplication gates decompose into 32 CNOT gates for a total of 77 CNOT gates. Each Fourier transform decomposes into 3 Hadamard gates. Therefore, there are a total number of 104 binary gates in this circuit excluding SWAP gates and including the state preparations. To prepare the $|\bar{0}\rangle$ state, we do not need the first 3 SUM gates (9 CNOT gates) nor the first 4 multiplication gates (8 CNOT gates) so there are 87 binary locations in this circuit. Finally, if we consider the Fourier transform gates to be part of state preparation, there are 78 binary locations. This does not include waiting locations.

We can also express this code as a binary stabilizer code. The generator matrix of this code is

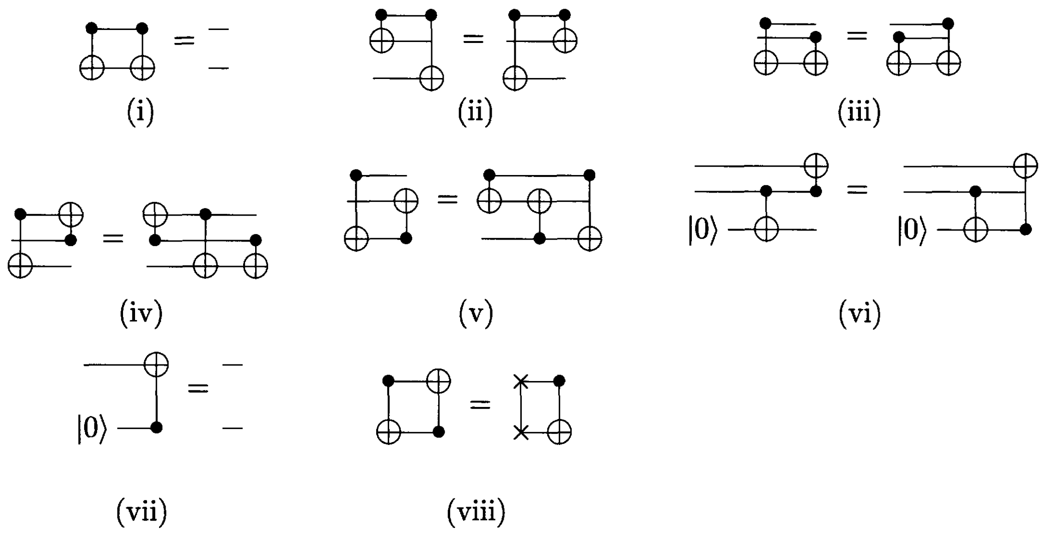


Figure 8-10: Complete transformation rules for CNOT circuits. Rule (i) is a cancellation rule because CNOT is self-inverse. Rules (ii) and (iii) express the fact that CNOT gates commute if a control and target do not touch the same bit. Rules (iv) and (v) give the commutators of CNOT gates when a control and target touch the same bit. Rules (vi) and (vii) give equivalences when an input is known to be zero. Finally, rule (viii) expresses the fact that the SWAP gate is a sequence of three CNOT gates. This last rule is included because SWAP gates can be ignored in some models of fault-tolerant computation. Rules (i), (vii), and (viii) reduce the number of gates in a CNOT circuit, while the other rules may produce circuits to which rules (i), (vii), and (viii) can be applied.

$$G_{[[21,3,\geq 4]]} = \left[\begin{array}{cc} I_9 & A_{21} \\ 0 & B_{21} \end{array} \right] \text{ where}$$

$$A_{21} = \left[\begin{array}{ccc|ccc|ccc|ccc}
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ \hline
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \hline
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0
\end{array} \right] \tag{8.17}$$

and

$$B_{21} = \left[\begin{array}{ccc|ccc|ccc|ccc}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0
\end{array} \right]. \tag{8.18}$$

The encoder can again be found by standard methods and is shown in Figure 8-11.

Example $[[15,1,8]]_{16}$ **or** $[[60,4,10]]_2$

This code is derived from a $[[15,8,8]]_{16}$ BCH code (GAP command `BCHCode(15,8,GF(16))`) with generator polynomial $g(x) = x^7 + z^6x^6 + z^{13}x^5 + z^{12}x^4 + zx^3 + z^5x^2 + z^{11}x + z^{13}$. The Conway polynomial for $GF(16)$ is $x^4 + x + 1$ and $\{z^3, z^7, z^{13}, z^{12}\}$ is a self-dual basis of the field.

$$G_{[[15,1,8]]_{16}} = \left(\begin{array}{ccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & z^{13} & z^4 & z^{14} & z^8 & z^{13} & z^1 & 1 & z^{13} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & z^{11} & z^{14} & z^6 & z^8 & z^7 & z^2 & z^{12} & z^{12} \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & z^{10} & z^6 & z^{10} & z^9 & z^1 & z^5 & z^7 & z^3 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & z^1 & z^6 & z^3 & z^{14} & z^3 & 1 & z^{11} & z^{14} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & z^{12} & z^9 & 1 & z^4 & z^5 & z^{14} & z^3 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & z^{13} & z^6 & z^4 & z^2 & z^{11} & z^2 & z^3 & z^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & z^6 & z^1 & z^{10} & 1 & z^3 & z^2 & 1 & z^2 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & z^{13} & z^{12} & z^3 & z^{14} & 1 & z^8 & z^2
\end{array} \right). \tag{8.19}$$

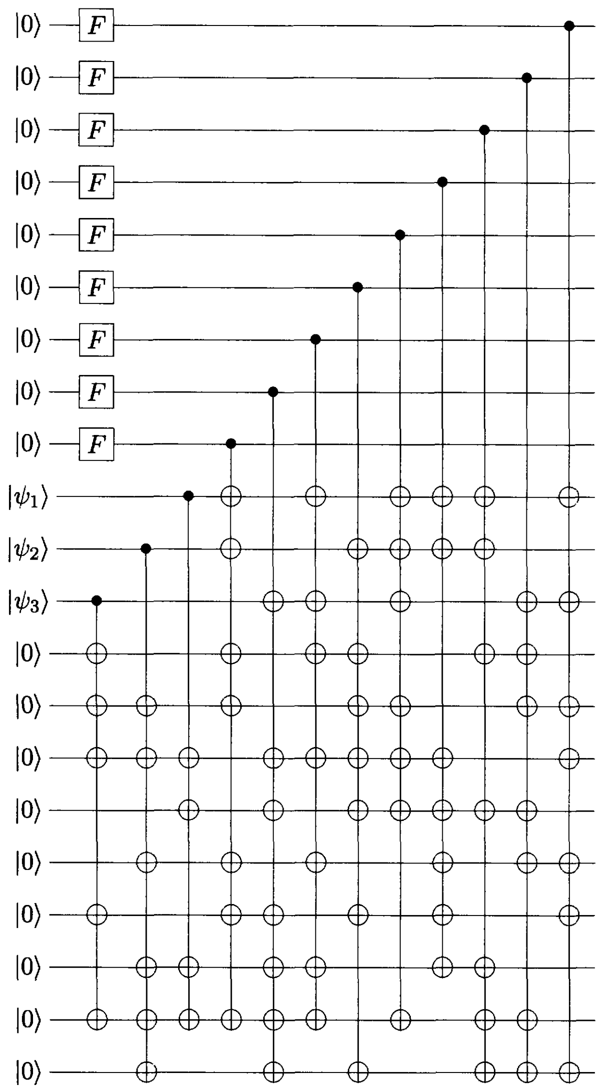


Figure 8-11: Encoder for the $[[7,1,4]]_8$ code expressed using qubit gates. This encoder is found directly from the binary generator matrix of C_1. It has 78 CNOT gates, 9 Hadamard gates, and 18 preparations for a total of 105 binary locations. The first 15 CNOT gates are not necessary to prepare $|\bar{0}\rangle$ ancilla states, so this operation has 90 binary locations. If we consider the Hadamard gates to be part of the state preparation, then there are 81 binary locations. This does not include waiting locations. Here $F = H$.

The following is matrix representation of the field over the self-dual basis:

$$z = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad z^2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad z^3 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad z^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$z^5 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad z^6 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad z^7 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad z^8 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$z^9 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \quad z^{10} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad z^{11} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad z^{12} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$z^{13} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad z^{14} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

The representation can be used to construct the binary encoder, or the binary encoder can be constructed using Steane's methods after concatenation.

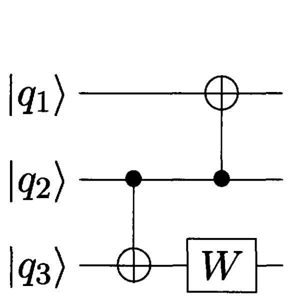
8.4.4 Bacon-Shor codes

Compact ancilla verification circuits

The Bacon-Shor ancilla states for error-correction can be simplified to products of cat states or their conjugates [AC07]. Specifically, an n qubit cat state is the state $|\text{cat}\rangle \propto |\underbrace{00\ldots0}_{\text{n}}\rangle + |\underbrace{11\ldots1}_{\text{n}}\rangle$. The encoded computational basis states are $|\bar{0}\rangle \propto \bigotimes_{n \text{ columns}} H^{\otimes n}|\text{cat}\rangle$ and $|\bar{+}\rangle \propto \bigotimes_{n \text{ rows}} |\text{cat}\rangle$ where H is the Hadamard gate.

When verifying $|\text{cat}\rangle$ states, it is sufficient for fault-tolerance to ensure that $w \leq t = \lfloor (n-1)/2 \rfloor$ faults do not produce an error at the output of the state preparation with weight greater than w. Because Bacon-Shor codes are CSS, X and Z errors may be considered separately. For cat states, all Z errors are equivalent to a single Z error modulo the cat state stabilizer, so Z errors satisfy the condition. Further, any error of weight $t+1$ or greater is equivalent to an error of weight t or less modulo the stabilizer. For codes with odd n, any error of weight t or greater is equivalent to an error of weight t or less modulo the stabilizer.

A standard way of verifying ancilla states for CSS codes was given in Chapter 7. This "full" verification method uses X or Z syndrome extraction. Full verification uses many more gates than necessary for small block sizes but may create ancilla with favorable statistical properties. The ancilla states for Bacon-Shor codes only require verification against one type of error, X errors for the $|+\rangle$ state and Z errors for the $|0\rangle$ state, because Z errors (X errors) of any weight reduce to weight 1 errors modulo the cat (conjugate cat) stabilizer.



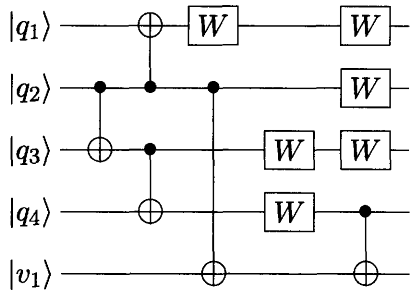
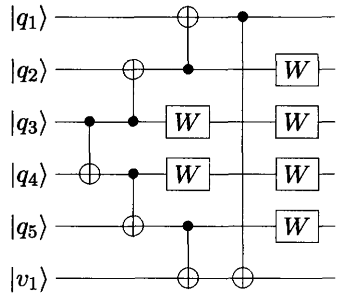


Figure 8-12: Networks to verify 3 qubit (6 locations) and 4 qubit cat states (17 locations)

On the other hand, it is possible to verify cat states using a minimal number of parity checks involving pairs of qubits. At least $t-1$ parity checks are necessary to verify a cat state. A single fault in the preparation network can create a weight t error prior to verification. With the remaining $t-1$ faults, up to $t-1$ additional parity checks can be fooled. If there are fewer than $t-1$ parity checks, then a weight t error can pass verification when there are fewer than t faults. Therefore, at least $t-1$ parity checks are necessary.



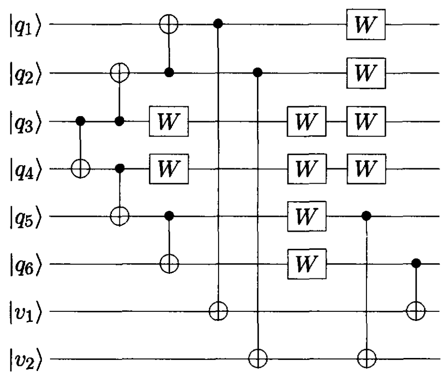


Figure 8-13: Networks to verify 5 qubit (19 locations) and 6 qubit cat states (29 locations)

Minimal verification networks exist that meet or nearly meet this lower bound for small n. Networks for the first few values of n are given in Figures 8-12, 8-13, 8-14 and 8-15. The number of fault locations in these networks are 5 to 10 times smaller than full verification. The networks are not unique and were found by exhaustive computer search. A few complications are worth mentioning. First, it is not surprising that $t-1$ parity checks are not always sufficient. In cases where $t-1$ parity checks were not sufficient, successively larger numbers of parity checks were searched until a sufficient set of checks was found. Second, when an error pattern of weight $w' > w$ occurs for a weight w error, $w' - w$ parity checks must cover that error pattern. Otherwise fewer than w' faults could produce this error pattern. This additional constraint modestly complicates the search. Third, there are many sets of 4 parity checks that cover all of the error patterns for $n = 8$ and $n = 9$. However, there are no disjoint sets of four parity checks, i.e. at least one pair of parity checks involves a common qubit. This creates a situation where faults within the verification circuit can cause a single high weight error pattern to pass verification and at least one additional parity check is required.

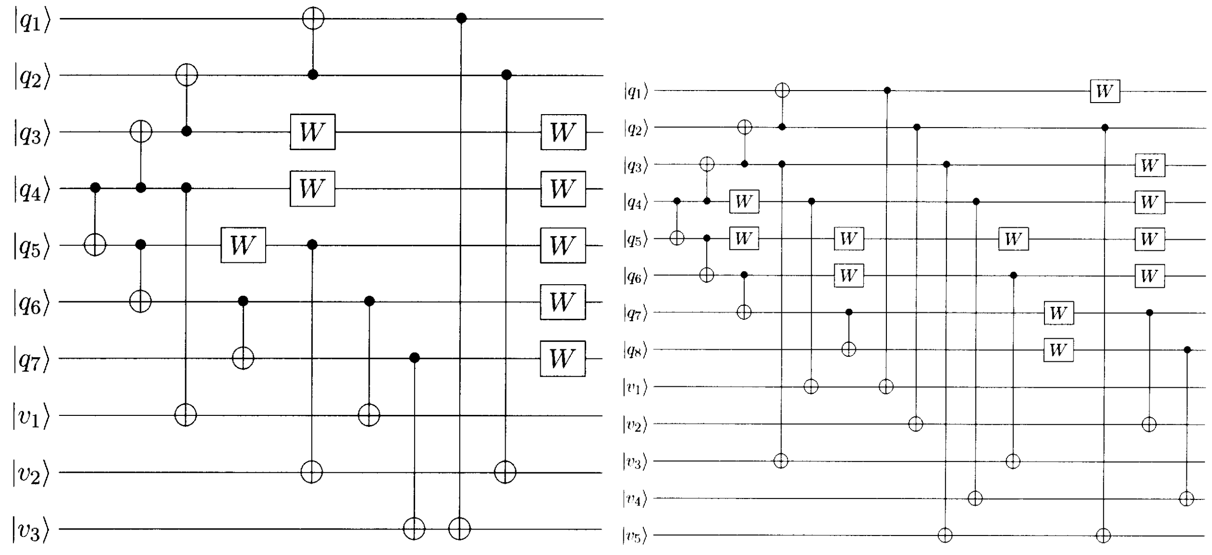


Figure 8-14: Networks to verify 7 qubit (33 locations) and 8 qubit cat states (47 locations)

Observation about large block size behavior

It is natural to ask if the asymptotic threshold is nonzero for the Bacon-Shor codes. If $t+1$ bitflip errors occur, they produce a weight $t+1$ error pattern that the code can correct unless each fault is in a different column of the lattice. We can create an uncorrectable pattern by choosing $t+1$ of n columns in which to place the fault, and we can do this in n ways for each of the $t+1$

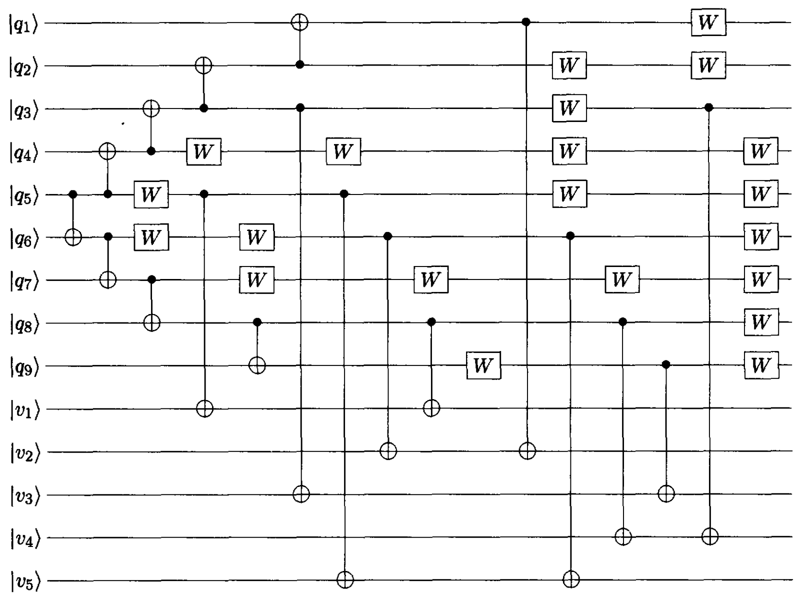


Figure 8-15: Prepare a 9 qubit cat state (58 locations)

faults. Therefore, there are $n^{t+1}\binom{n}{t+1}$ ways to choose an uncorrectable configuration of $t+1$ faults. The total number of patterns produced by $t+1$ faults is $\binom{n^2}{t+1}$, so we can see that the fraction of uncorrectable patterns drops off exceedingly fast.

Now we will try to estimate the asymptotic value of the bit-flip fixed point. An approximation to the threshold is given by

$$\left\{(2t+1)^{t+1}\binom{2t+1}{t+1}\right\}^{-1/t}. \tag{8.20}$$

This approximation neglects terms of order $t+2$ and higher, so it is not a bound of any sort. From [Wor94] equation 2.10,

$$\lim_{t\to\infty}\binom{2t+1}{t+1}^{-1/t} = \lim_{t\to\infty} Q(2,t)^{-1/t} = \frac{1}{4}\lim_{t\to\infty}\sqrt[2t]{\pi t} = \frac{1}{4} \tag{8.21}$$

where

$$Q(a,n) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{n}}\sqrt{\frac{a^2}{a-1}}\left\{a^{1/a}\left(\frac{a}{a-1}\right)^{(a-1)/a}\right\}^n. \tag{8.22}$$

It follows that

$$\lim_{t\to\infty}\left\{(2t+1)^{t+1}\binom{2t+1}{t+1}\right\}^{-1/t} = \frac{1}{4}\lim_{t\to\infty}(2t+1)^{-(t+1)/t} = 0. \tag{8.23}$$

This suggests, though it is not a proof, that the number of uncorrectable errors grows too quickly by a factor of n for the limiting value to be nonzero.

8.5 Results

This section presents the results of the inner code study using the methods from Chapter 7. Tables B.1, B.2, B.3, B.4, and B.5 in the Appendix list the complete set of results of our studies. Our results are obtained assuming that all locations including memory locations suffer from noise at the same noise rate, unless specified otherwise.

8.5.1 Perfect Ancillas

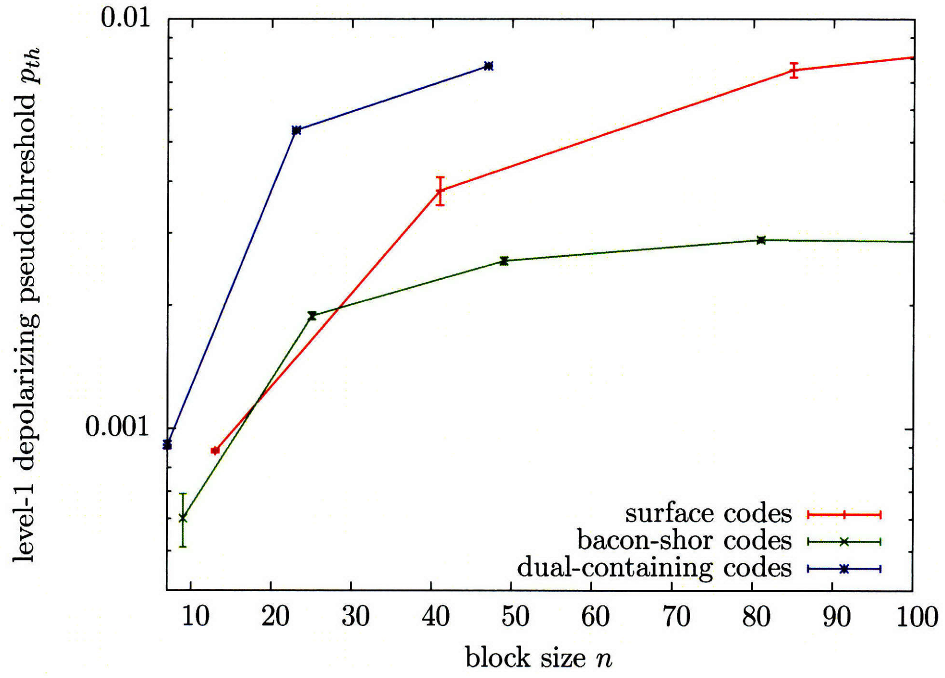


Figure 8-16: Level-1 depolarizing pseudo-threshold for three families of codes with perfect ancillas for Steane-EC: surface codes, dual-containing codes, and Bacon-Shor codes. This plot indicates that under no circumstances can thresholds reach 1% for the codes in our study. The data points are connected by lines merely as a guide to the eye.

In our first study, and only in this section, we assume that ancillas for Steane error correction can be prepared flawlessly, see Figure 8-16. In such a scenario, the threshold is largely determined by the error correction properties of the code (see also the analysis in [Eas07]), in particular its (effective) distance. For families of quantum error-correcting codes in which the effective distance is linear in the block-size, we expect the threshold to be monotonically increasing as a function of n, see Section 7.2.2. In Figure 8-16 and Figure 8-17 we have plotted the pseudo-thresholds for three families of codes: surface codes, some dual-containing codes, and Bacon-Shor codes. The surface codes and Bacon-Shor codes apparently have fairly good distance properties, even though there is some decline in the Bacon-Shor code family for large n. Figure 8-16 shows we cannot expect a threshold over 1% for the codes we have studied using Steane-EC – introducing noise realistically into the ancilla preparation circuits cannot increase the pseudo-threshold. Note that if we do Shor-EC on the surface codes we cannot expect thresholds exceeding about 3%, see the arguments in [DKLP02].

When we assume that the logical Bell pairs of Knill's circuit can be prepared flawlessly, the level-1 pseudo-threshold [2]. of the $[[5,1,3]]$'s M_3 gate is $(2.0 \pm 0.1) \times 10^{-4}$. This is roughly an order of magnitude below the Steane code with perfect ancillas.

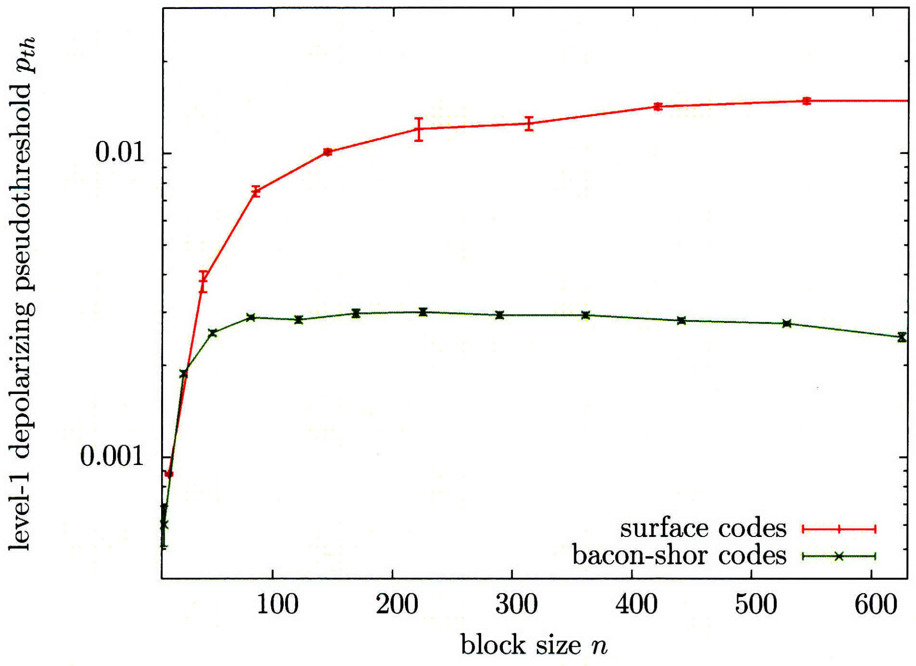


Figure 8-17: Level-1 depolarizing pseudo-threshold for surface codes and Bacon-Shor codes using perfect ancillas for Steane-EC.

8.5.2 Pseudo-Thresholds

In Figure 8-18 we tabulate for each code the maximum pseudo-threshold over the various choices of R and L. The maximum overall pseudo-threshold $(2.25 \pm 0.03) \times 10^{-3}$ is attained by the Golay code with $L = 30$ and $R = 1$.The two code families, Bacon-Shor and surface, both attain a peak

[2]The pseudo-threshold in this case is the point at which the failure rate p_1 of a M_3 ex-Rec is the same as the base error rate p_0 of all elementary gates in the ex-Rec.

threshold and then decline when we use Steane-EC. The peak Bacon-Shor code is the [[49, 1, 7]] at $(1.224 \pm 0.005) \times 10^{-3}$ with $L = 9$ and $R = 1$. The peak surface code (using Steane error correction) is [[41, 1, 5]] at $(1.008 \pm 0.008) \times 10^{-3}$ at $L = 30$ and $R = 1$. Interestingly when we use Shor-EC for the surface codes the performance is quite different. Shor-EC does not do as well as Steane-EC for small block sizes, but for larger block size Shor-EC gives a threshold that asymptotes to a finite value in the limit of large n, see Figure 8-19. For small block size the thresholds of the surface codes are not as good as of some other codes such as the Golay code and the Bacon-Shor codes.

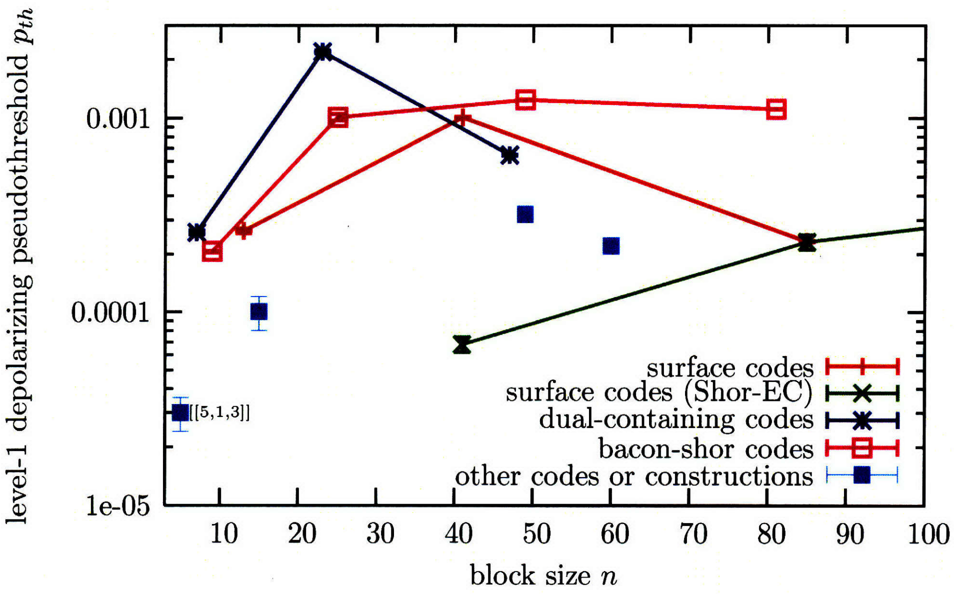


Figure 8-18: Level-1 depolarizing pseudo-threshold versus block size. The other codes are the [[5, 1, 3]] non-CSS code, the [[15, 1, 3]] Reed-Muller code, the [[49, 1, 9]] (dual-containing) concatenated Steane code using $L = 15$ attempts to prepare using error detection at level-1, and the [[60, 4, 10]] (dual-containing) concatenated polynomial code using $L = 20$ attempts to prepare ancillas.

It is clear from the data that the pseudo-threshold increases with increasing L. Our main interest in this study is in circuits with small overhead and hence with a relatively small number of preparation attempts L. In various cases the thresholds stated for finite L will be thus be lower than the one in the $L \to \infty$ limit. Notably, this occurs for the [[49, 1, 9]] code, where we expect thresholds approaching 1×10^{-2} with many more ancilla preparation attempts [Rei04]. In other cases one can take the perfect ancilla results in Figure 8-16 and the Tables as upper bounds on the $L \to \infty$ pseudo-threshold.

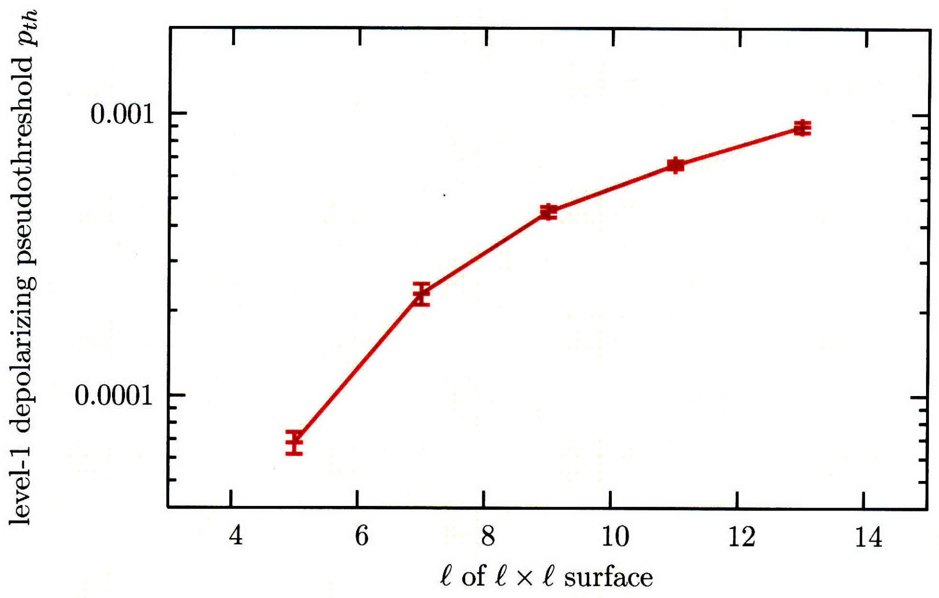


Figure 8-19: Surface code level-1 depolarizing pseudo-threshold versus ℓ for $\ell \times \ell$ surface code (the block-size $n = \ell^2 + (\ell - 1)^2$). The ex-Rec is a transversal CNOT gate with ℓ sequential Shor-EC steps per EC. The pseudo-threshold increases with ℓ and is expected to approach a constant value in the limit of large ℓ, unlike the other codes in this study.

8.5.3 Influence of Storage Errors

In Figure 8-20 we replot the pseudo-threshold versus block-size when storage error rates (on memory locations) are zero. The peak pseudo-threshold increases to $(3.33 \pm 0.02) \times 10^{-3}$. The Figure shows that storage errors do not influence the pseudo-threshold appreciably. The Bacon-Shor codes are least affected by storage errors because the encoding circuits are extremely simple. The non-CSS $[[5,1,3]]$ code is most greatly affected because storage errors can enter into the M_3 gate sub-circuit, the $|\overline{0}\rangle$ encoders, and the cat-state encoders at many locations.

8.5.4 Logical Error Rate versus Overhead

The threshold is an extremely important figure of merit for fault-tolerant circuit constructions. But practically speaking, we are also interested in how quickly the error rate decreases if the initial error rates are low enough for a given overhead. Figures 8-21, 8-22, and 8-23 plot the probability of failure of a CNOT ex-Rec (defined in Section 7.4.2) versus the number of physical CNOTs in a rectangle at $p_0 = 10^{-4}$. The Golay code achieves the lowest logical error rate for codes with fewer than $O(10^4)$ CNOT gates per rectangle, and that rate can be further reduced by increasing the number of verification rounds to $R = 2$. There is a clear tradeoff between the number of physical CNOTs

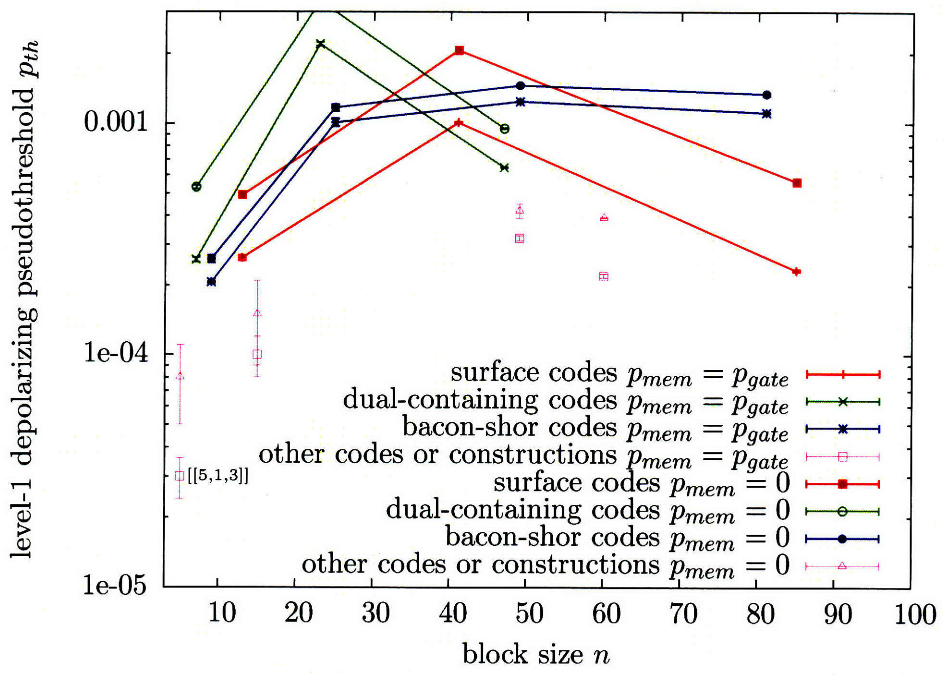


Figure 8-20: Pseudo-thresholds versus block size for Steane-EC and Knill-EC circuits, comparing the case where the memory failure rate equals the gate failure rate with the case where the memory failure rate is zero. Naturally the difference is smallest where we have taken advantage of simple encoders as those for the Bacon-Shor codes.

per rectangle and the logical error rate. We note that given the lack of code specific optimizations, the achievable overheads for various codes may be less than what is estimated here. For the Golay code and the Bacon-Shor codes for example, the overhead may come down by at least a factor of 2 by using simplified verification circuits. We also see in Figure 8-22 that the approximate expression for the failure rate, Eq. (7.1), gives a pretty good estimate of the actual failure rate.

Some of the error rates plotted in Figure 8-21 were extrapolated from error rates at higher values of p_0. For small values of p_0 the logarithm of the error rate $p_1(p_0)$ is expected to be approximately linear in p_0. We extrapolate from a least-squares fit to this line. Tables B.1, B.2, and B.3 indicate these extrapolated rates by enclosing them in square brackets. The extrapolations are plotted for the 5×5 surface code and the 9×9 Bacon-Shor code.

For the Golay code we have looked at the behavior of the threshold for $R = 1, 2, 3$. One important empirical observation is the following. The pseudo-threshold can increase slightly while the logical error rate for $p_0 = 10^{-4}$ remains the same. This happens for the Golay code when $R = 1$ and L is increased from 10 to 20. Furthermore, the pseudo-threshold can decrease while the logical error rate decreases too. This also happens for the Golay code when $L = 10$ and R is increased

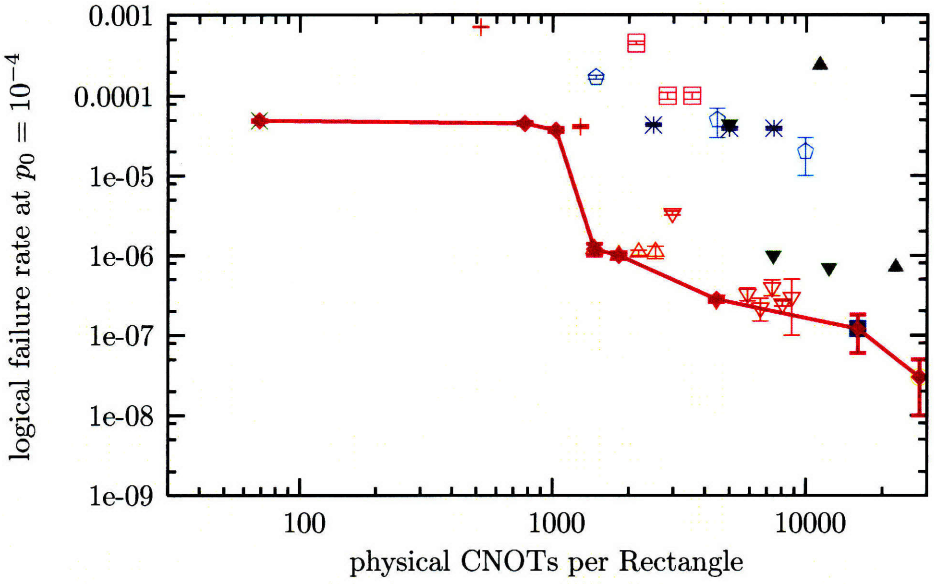


Figure 8-21: Level-1 logical error rate (probability of failure of a CNOT ex-Rec) versus the number of CNOTs per rectangle. The line connects the points of the best performing codes. Points with the same shape (color online) belong to the same code but have different circuit parameters. The error rates are evaluated at a fixed $p_0 = 10^{-4}$.

from 1 to 2. This suggests that the pseudo-threshold value is sensitive to higher order effects that quickly become negligible at lower error rates. Thus a desired logical error rate may be achievable with significantly fewer ancilla resources L than are necessary to maximize the pseudo-threshold, provided the initial error rate p_0 is not too close to the pseudo-threshold.

In Figure 8-22 we have also added Knill's C_4/C_6 Fibonacci scheme [Kni05a] at 2 and 3 levels of encoding. These data points are derived from his paper [3]. At level 2 the detected error rate of the logical CNOT is $(1.06 \pm 0.01) \times 10^{-5}$ and at level 3 the detected error rate is $(2.18 \pm 0.02) \times 10^{-8}$.

The plot shows that [[9,1,3]] is still better than the C_4/C_6 scheme in terms of overhead, but the C_4/C_6 Fibonacci scheme definitely beats [[7,1,3]]. The next two Bacon-Shor codes fill a void between C_4/C_6 level 2 and C_4/C_6 level 3.

For the surface codes (see Fig. 8-23) we note that the error rates are relatively high compared to other error-correcting codes with comparable numbers of CNOTs per rectangle. However one should remember that the circuits for the surface codes are already spatially local in two dimensions whereas the circuits for any of the other codes, for example, the Golay code, are not.

[3]Note that his error model is slightly different from ours but we take the dominating physical CNOT error rate to be the same.

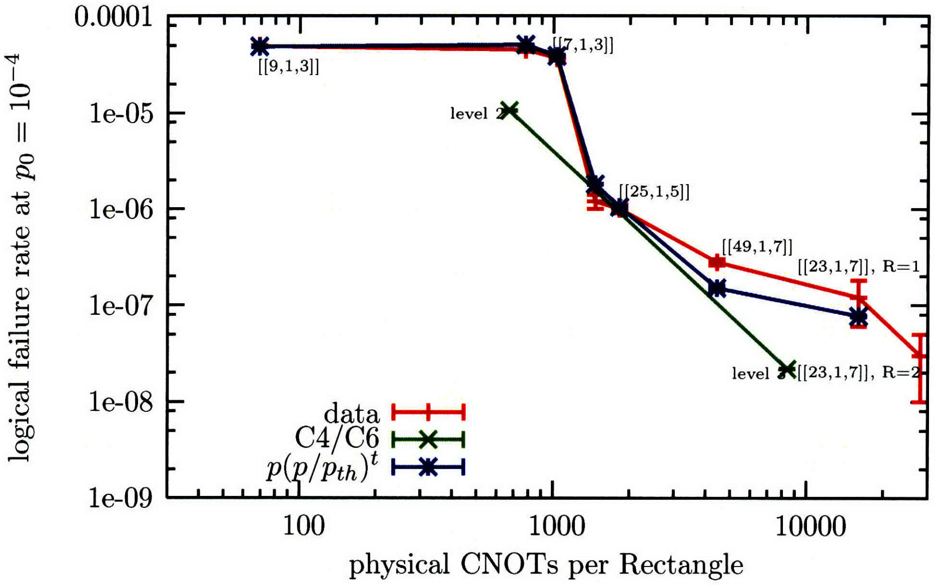


Figure 8-22: Level-1 logical error rate (probability of failure of CNOT ex-Rec) versus the number of CNOTs per rectangle for the best performing codes. The subset of data plotted here was chosen so that the error rate decreases monotonically with the rectangle size and there is no code with lower error rate at a given rectangle size. The error rates are evaluated at a fixed $p_0 = 10^{-4}$. The results for the C4/C6 scheme of [Kni05a] are shown for comparison.

8.5.5 Computer use

The simulations were carried out on a relatively small allocation of Blue Gene L at the IBM T. J. Watson Research Center. Typically we used between 64 and 256 PowerPC 440 700 MHz CPUs. Each pair of CPUs had access to 512 Mb of local memory. Using 256 CPUs gave us roughly a factor of 50 speed-up over a typical single-processor desktop machine. The entire process of development and debugging took many months, but we estimate that all of the data could be retaken in several weeks with these computing resources.

8.6 Conclusion

In our study we have considered inner codes and their performance in a physical-inner-outer code architecture. Our best threshold around 2×10^{-3} is seen for the Golay code, and many other codes both larger and smaller were studied and found to have much worse thresholds. An important figure of merit is the logical error rate versus overhead curve which shows that the Bacon-Shor codes are competitive with Knill's C_4/C_6 scheme at base error rate 10^{-4}.

In this landscape of codes and their performances, one of the missing players is the surface

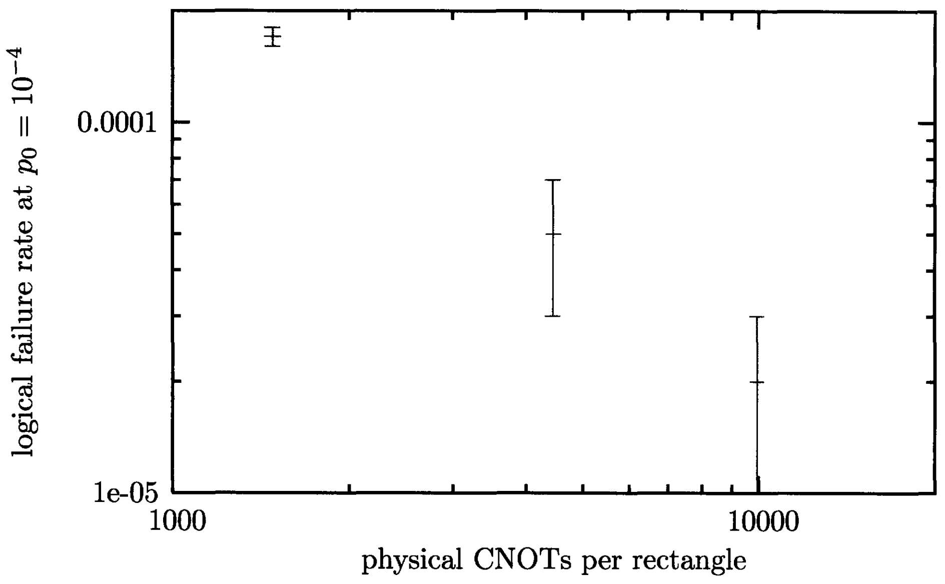


Figure 8-23: Level-1 logical error rate versus the number of CNOTs per rectangle for the $\ell \times \ell$ surface codes, $\ell = 5, 7, 9$. It is expected that the error rate decreases exponentially as ℓ increases for fixed $p_0 = 10^{-4}$.

code scheme of [RHG07] in which many qubits are encoded in one surface code and the CNOT gate is done in a topological manner. In principle, the possible advantage of this scheme is that if one uses enough space (meaning block size) one would reach the asymptotic threshold of a simple EC rectangle (no 1-Ga). We have in fact analyzed an ex-Rec where the Rec is only Shor-EC on a $\ell \times \ell$ surface and we find that this asymptotic memory threshold for $\ell \to \infty$ is about 3.5×10^{-3}. This is a factor of two lower than the number stated in [RHG07]. For finite block size one could analyze a CNOT ex-Rec for this topological scheme just as for the other codes. Like all the other codes, the topological scheme will have a trade-off between overhead and logical error rate. It will be interesting to see whether topology and block coding provide an efficient way of using resources and how it compares to a local version of a physical-inner-outer architecture.

For a physical-inner-outer architecture it will be important to study the performance of outer codes in order to understand at what error rate one should switch from inner to outer code and what total overhead one can expect. Concerning a choice of outer code we expect the following. First of all, given the constructions of [SI05], one can expect that a $[[n, k, d]]$ block code has a threshold comparable to a $[[n, 1, d]]$ code. Secondly, the networks in [SI05] show how to do logical gates on qubits inside the block codes using essentially gate-teleportation and Knill-EC. One issue of concern for block codes is the complexity of the encoding circuit as a function of block size. It would be highly desirable to consider block codes with EC circuits that are linear in n; otherwise, one would expect the threshold to decline as a function of n.

There is another desirable property of outer codes which relates to the transversality of gates. In order to minimize overhead, it is desirable that the T gate is transversal for the outer code. The reason is as follows. In order to have maximal freedom in picking a inner code we will only require that it is has a transversal CNOT. Thus all other gates, in particular $\mathrm{T} = e^{i\pi Z/8}$ and the phase gate K need to be performed by alternative means, namely the injection-and-distillation scheme. The obtained error rates of the encoded and distilled ancillas will be limited by the noise rates on the Clifford gates which distill the ancillas, since the Clifford distillation circuit is not fault-tolerant. Assume we teleport the ancillas into $C_{\text{outer}} \circ C_{\text{inner}}$ and get Clifford gates with $O(10^{-15})$ error rate. Since a circuit such as Bravyi-Kitaev distillation uses $O(10^3)$ gates, the error rates of the distilled ancillas can be as high as $O(10^{-11})$. Thus by these schemes the T error rate is always trailing the transversal gate error rates. But assume that the T gate is transversal for the outer code and thus we only inject the T ancillas into C_{inner}. Then even though the once encoded gate $C_{\text{inner}}(\mathrm{T})$ has an error rate of, say, $O(10^{-4})$, the twice-encoded gate $C_{\text{outer}} \circ C_{\text{inner}}(\mathrm{T})$ will mostly likely have an error rate similar to other Clifford gates since there are very few $C_{\text{inner}}(\mathrm{T})$ in the twice-encoded gate compared to the EC parts.

Chapter 9

Conclusion

This dissertation has revealed limitations of standard approaches to fault-tolerant quantum computing, proposed methods to evaluate quantum computer architectures built using concatenated hierarchies of error-correcting codes, and discovered additional structure of quantum codes. First, it is clear from Chapter 6 that quantum computer architectures are fundamentally different from classical computer architectures due to limitations on fault-tolerant gates. In particular, the need for mechanisms to create and distribute quantum software appears to be unavoidable, since we cannot hope to use *only* transversal gates; see Figure 9-1. Second, the results of the inner code survey in Chapter 8 strongly suggest that the universe of effective inner codes is limited. Of all the codes we surveyed, only the Bacon-Shor codes, surfaces codes, and Golay code stand out as possible inner code candidates in a three level physical-inner-outer hierarchy. Furthermore, only the Bacon-Shor and surface codes appear to be competitive with non-standard fault-tolerance methods using concatenated error-detecting codes and message passing. However, more positively, a three level architecture appears to be effective; i.e, with only three levels of coding, it *is* possible to suppress very high error rates down to $O(10^{-15})$ and solve otherwise intractable problems. Furthermore, we now have much more knowledge about the universe of available quantum codes, having discovered a large new family of codes in Chapter 4, and having proven new results about the symmetries of stabilizer codes in Chapter 6.

Let us revisit what has been accomplished in each chapter of this dissertation, beginning with Chapter 2. Chapter 2 reviewed the quantum circuit model, stabilizer circuit model, and the essential elements of the theory of open quantum systems, so that the concept of decoherence and the need for techniques to combat decoherence could be appreciated. Chapter 3 reviewed a model for a

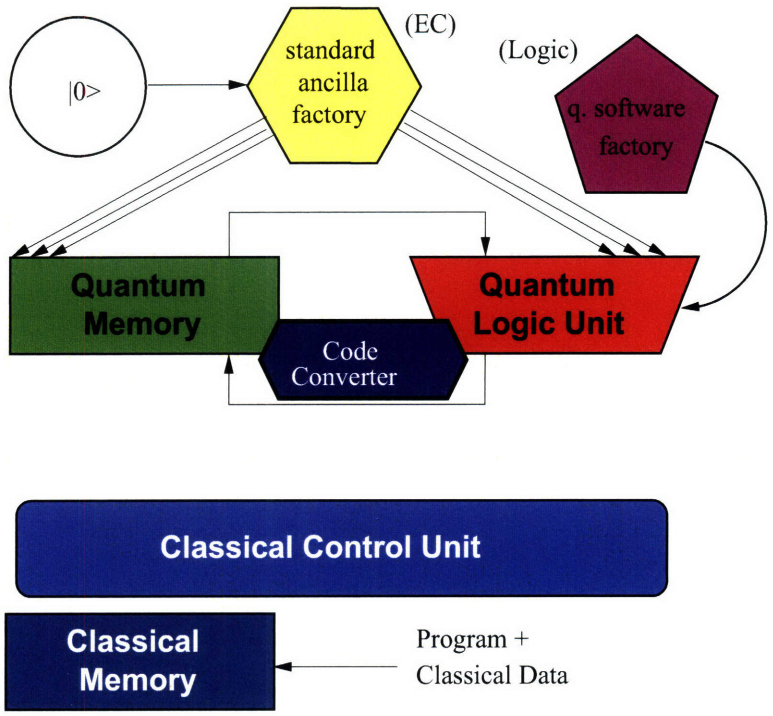


Figure 9-1: The stored-program architecture introduced in Chapter 1 is inadequate for fault-tolerant quantum computation in light of the no-go theorems presented in Chapter 6. These no-go theorems strongly suggest that specialized "quantum software" states are necessary to implement a universal set of gates. These states differ from ancilla used in error-correction, are somewhat more difficult to construct, and are used in a different way. Therefore, an additional unit, the quantum software factory, is an essential part of this vision of a fault-tolerant quantum computer architecture.

trapped-ion quantum charge-coupled-device architecture. This model is a basic device level model together with notions of geometry and specific noise parameters. Such a model can be constructed for any candidate physical system and incorporated into the circuit constructions of Chapter 8, for example. We did not use such a model in our evaluation because good models require expertise in the physics of a particular system and access to experimental data about noise and systematics. However, the concept was important for us to emphasize since incorporating such a model is crucial for realistic application of the ideas in this dissertation.

Chapter 4 reviewed the stabilizer formalism and gave examples of important stabilizer codes that were used in Chapters 6 and 8. Our first new result was described in this chapter – a new family of quantum codes called *codeword stabilized codes* that contains all of the stabilizer codes as well as many new codes. All known codes with good parameters are codeword stabilized codes, and several optimal codes are as well. Furthermore, we presented methods for systematically understanding and constructing these codes from $GF(4)$-additive codes and (nonlinear) classical codes. We gave an algorithm for finding all CWS codes with small block sizes. In further work, beyond the results presented in this dissertation, we have investigated the structure of CWS codes and connected these codes to the framework of Aggarwal and Calderbank [CCS$^+$08].

A promising question for future research is the following: How can the systematic codeword stabilized framework can be suitably expanded and/or relaxed to include codes that correct different physically motivated noise models? The work in this dissertation only concerns codes that correct arbitrary low weight errors. However, codes that correct specific physically motivated noise models are very interesting because they may be used as physical level codes in a hierarchy of codes. For example, biased noise can be treated using asymmetric CSS codes, for example. Amplitude dampling noise has been treated with stabilizer codes as well [LNCY97, Fle07]. Interestingly, nonadditive codes with better parameters can be constructed for these models [LS07].

There is another promising question related to CWS codes: Does there exist a family of non-additive codeword stabilized codes with a high degree of symmetry, so that members of the family can be efficiently encoded and error-corrected? In general it seems that the error-correction circuits for these codes will have exponential size. However, there are probably specific constructions that have simple circuits that relate in an elegant way to error-correction circuits for stabilizer codes. For example, one family of interesting nonadditive codeword stabilized codes has already been discovered. The quantum Goethals-Preparata CWS codes were found by Grassl and Roetteler using an idea of *codespace* stabilization [GR08]. One reason these codes are interesting is that they have a large subgroup contained in the set of word operators. Perhaps this near-additivity will yield simple circuits. Finally, one may wonder if the usual methods of obtaining new codes from existing codes, such as lengthening, shortening, puncturing, and concatenating, lend themselves to a simple, useful graphical interpretation in the CWS framework.

Next, Chapter 5 provided a pedagogical introduction to code-based fault-tolerance in preparation for later chapters. Fault-tolerant quantum computation and the threshold theorem provide the main reason for our continued hope that large scale quantum computation will one day be achieved. The chapter defined and introduced a standard approach to classical and quantum fault-tolerant computing, while reminding us that there are other (non-standard) approaches and significant open questions in quantum fault-tolerance.

In Chapter 6, we studied computation on stabilizer codes and proved several new results about the form and limitations of transversal gates. Our mathematical techniques were based on a subcode method originally introduced by Eric Rains. We successfully applied the subcode method to find the generic structure of elements of the full automorphism group and the group of (basic) transversal gates. Using these results, we proved that (basic) transversal gates and full automorphisms are not a universal sets. Finally, we showed that CSS codes with non-Clifford transversal gates are built from classical divisible codes. This allowed us to construct new examples and propose the interesting conjecture that all non-Clifford transversal logic gates are so-called C_k gates – exactly the gates that appear in the study of quantum teleportation! Will this conjecture be proven or refuted? Is there a stabilizer code with a transversal gate not in C_k, such as a transveral $\pi/3$-gate?

An open question beyond the work in Chapter 6 is: How much can the notion of "transversality" be strengthened before universal quantum computation is possible with such gates? In particular, we know that if transversal measurements are allowed, universal gates can be constructed using quantum software methods. Furthermore, allowing the gate input and output to be encoded in different codes certainly gives universality, as we have seen in Chapter 8. However, in this case, we have the added need to transform the output code back into the input code space at some point, which again uses quantum software methods. In further work, we have shown that transversal gates are not universal for codes on higher dimensional systems as well as on qubits [CCC$^+$08]. However, we still do not know if more complicated permutation transversal gates can be universal. The main obstacle to proof using subcode methods is that this set of gates is not a group.

Another open question related to transversality is: does there exist a stabilizer code encoding multiple qubits where a non-Clifford gate can be performed transversally on some of the logical qubits? We have examples of transversal logical gates in C_k for all k, and these codes can be constructed for any distance d using concantenation. However, we have no examples encoding more than one qubit! CSS codes with encoding multiple qubits, with transversal (or even almost-transversal) non-Clifford gates, could be chosen based on the frequency of non-Clifford gates in subroutines of a quantum algorithm. For example, we could endeavor to find a code that makes the most commonly occuring gates in an algorithm transversal and implement the rare gates using quantum software methods.

A final, broad open question on this topic: what is the mathematical form of full automorphisms for nonadditive codes such as the codeword stabilized codes? This area is almost entirely unexplored. For CWS codes, we easily know that the logical Pauli X gates are transversal by definition. Other gates are more difficult to apprehend. Although it does not seem likely to us, perhaps there is a strange counterexample lurking among nonadditive codes that has a universal set of transversal gates.

Chapter 7 defined what we mean by an effective code architecture and showed how to evaluate the architecture one layer at a time. Such an architecture, we argued, consists of multiple specialized codes. Rough estimates suggest that a three level code architecture, consisting of a physical, inner and outer code, will be effective for cryptographic applications. We have explained how code parameters determine the quantitative properties of a code architecture; i.e., the threshold scaling with block size and minimum distance, scaling of overhead with rectangle parameters, and expected logical error rates at each level of coding. We reviewed general design rules for error-correcting code rectangles and explained why, for CSS codes, it is adequate to study CNOT rectangles. Finally, our new result in this chapter was two new Monte-Carlo adaptations of the methods of Aliferis, Gottesman, and Preskill that allow us to evaluate larger codes and rectangles than the original method permitted. The results we obtain from applying this method are closely related to rigorous bounds [AGP06].

There are many open engineering, computer systems, and computer architecture problems in the area of quantum computer architecture design and evaluation. If history is any indication, the task of designing a quantum computer can be greatly accelerated by computer-aided design tools and compilers. In other work, we have proposed a software architecture for quantum computing design tools [SCA$^+$06]. Some elements of this tool chain have been implemented, such as the evaluation tools discussed in Chapter 7, simulators, quantum circuit compilers, layout tools, and physical operation schedulers [MTC$^+$06]. Further work is needed to develop this open source tool chain. Additionally, the evaluation methods we use can be further extended to codes that encode multiple qubits, although there may be some minor technical details to overcome.

Our study in Chapter 8 considered inner codes and their performance in a physical-inner-outer code architecture. The best threshold of all inner codes surveyed was 2×10^{-3}, seen for the Golay code. Unfortunately, many other codes both larger and smaller were studied and found to have much worse thresholds. This negative result is important, however, since we can focus on those codes and techniques that are most likely to contribute to an effective architecture. The logical error rate versus overhead curve is perhaps the most practical figure of merit, and this figure shows that the Bacon-Shor codes (using standard methods) are competitive with Knill's C_4/C_6 scheme (using non-standard methods) at base error rate 10^{-4}.

In this landscape of inner codes, one of the missing players is the surface code scheme of [RHG07] in which many qubits are encoded in one surface code and the CNOT gate is done in a topological manner. In principle, the possible advantage of this scheme is that if one uses enough space (meaning block size) one would reach the asymptotic threshold of a simple EC rectangle (no 1-Ga). We have in fact analyzed an ex-Rec where the Rec is only Shor-EC on a $\ell \times \ell$ surface and we find that this asymptotic memory threshold for $\ell \to \infty$ is about 3.5×10^{-3}. This is a factor of two lower than the number stated in [RHG07], probably because we limit the syndrome history of our decoder to order ℓ parallel syndrome extractions. For finite block size one could analyze a CNOT ex-Rec for this topological scheme just as for the other codes. Like all the other codes, the topological scheme will have a trade-off between overhead and logical error rate. It will be interesting to see whether topology and block coding, together with inner physical coding, provide an efficient way of using resources and how it compares to a local version of a physical-inner-outer architecture proposed in this dissertation.

For some additional perspective on the results of the inner code survey, consider the history of threshold calculations from the discovery of fault-tolerant quantum circuits in 1996 to the present (2008), shown in Figure 9-2. The figure includes three kinds of threshold results: analytical estimates of the threshold based on rough counting of fault paths, numerical estimates based on Monte-Carlo estimates or computer-aided counts, and rigorous upper and lower bounds. First, the figure reveals a progression in the community from early estimates, to more optimistic numerical estimates, and finally to a rigorous understanding of the accuracy threshold value (although there are still two orders of magnitude between upper and lower bounds). Second, we see a trend toward higher threshold values over the last four years. The results in this dissertation are responsible for several numerical points and one rigorous bound in this figure.

For a physical-inner-outer architecture, an important problem is to study the performance of outer codes using methods like those in Chapter 7. Concerning a choice of outer code we expect the following. First of all, given the constructions of [SI05], one can expect that a $[[n,k,d]]$ block code has a threshold comparable to a $[[n,1,d]]$ code. Secondly, the networks in [SI05] show how to do logical gates on qubits inside the block codes using essentially gate-teleportation and Knill-EC. One issue of concern for block codes is the complexity of the encoding circuit as a function of block size. It would be highly desirable to consider block codes with EC circuits that are linear in n; otherwise, one would expect the threshold to decline as a function of n. However, as our calculation have suggested, a threshold of 10^{-5} or perhaps even 10^{-6} would be acceptable for an outer block code.

The choice of outer codes also relates to one of the open problems about transversality and gives us more reason to look for new, interesting outer codes. In order to minimize overhead, it

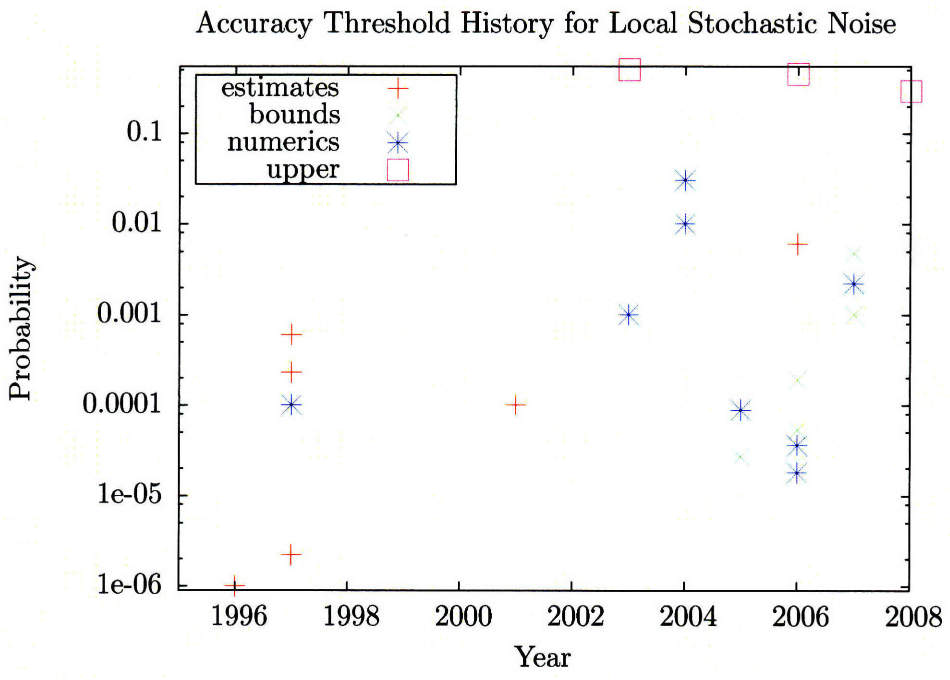


Figure 9-2: A summary of several quantum accuracy threshold estimates, numerics, and rigorous bounds appearing in the literature [CDT07, ABO97, ABO99, SDT07, AC07, AGP08, AGP06, AP07, DKLP02, Got97, KLZ96, Kni05a, Kni04b, Rei04, RHG07, Ste03, KLZ98, Pre98, SCCA06, Ali08]. Furthermore, we show upper bounds and the year those bounds were discovered [Raz04, BCL+06, KRUdW08].

is desirable that the T gate is transversal for the outer code. The reason is as follows. In order to have maximal freedom in picking a inner code we will only require that it is has a transversal CNOT. Thus all other gates, in particular $\mathrm{T} = e^{i\pi Z/8}$ and the phase gate K need to be performed by alternative means, namely the injection-and-distillation scheme. The obtained error rates of the encoded and distilled ancillas will be limited by the noise rates on the Clifford gates which distill the ancillas, since the Clifford distillation circuit is not fault-tolerant. Assume we teleport the ancillas into $C_{\text{outer}} \circ C_{\text{inner}}$ and get Clifford gates with $O(10^{-15})$ error rate. Since a circuit such as Bravyi-Kitaev distillation uses $O(10^3)$ gates, the error rates of the distilled ancillas can be as high as $O(10^{-11})$. Thus by these schemes the T error rate is always trailing the transversal gate error rates. Assume that the T gate is transversal for the outer code and thus we only inject the T ancillas into C_{inner}. Then, even though the once encoded gate $C_{\text{inner}}(\mathrm{T})$ has an error rate of, say, $O(10^{-4})$, the twice-encoded gate $C_{\text{outer}} \circ C_{\text{inner}}(\mathrm{T})$ will mostly likely have an error rate similar to other Clifford gates since there are very few $C_{\text{inner}}(\mathrm{T})$ in the twice-encoded gate compared to the EC parts.

In addition to overcoming lagging non-Clifford gate error rates, new outer codes with interesting transversal gates could be adapted to the quantum algorithm, as can, perhaps, the error-correction circuits. Roughly speaking, it may be desirable to use a code with transversal Toffoli during, say, modular exponentiation. However, during a quantum Fourier transform, one may want a code with higher order transversal $\pi/2^m$-gates. Furthermore, it may happen that one type of error is suppressed within a largely "classical" circuit like modular exponentiation, or, more generally, that one type of error commutes with the action of the circuit, as in some quantum simulations.

Our hope is that this dissertation encourages continued work on fault-tolerant quantum computer architectures. Noise-adapted physical codes, together with composite pulses and decoherence free subspaces, may transform the noise model so that it is amenable to inner coding using Bacon-Shor codes, surface codes, or error-detection based fault-tolerance. Incorporating notions of layout will further indicate the most promising codes. Work on outer levels of the code hierarchy will give a more complete picture of the requirements to solve challenging problems that are well beyond what modern digital computers can solve. Greater understanding of fault-tolerant gate techniques, perhaps using topological methods, may give us ways to reduce overhead and adapt outer codes to algorithms. Finally, developments in software tools and simulation methods may allow us ultimately to evaluate such a fault-tolerant architecture from end-to-end, so that we can move closer to realizing a large-scale quantum computer.

Appendix A

Notation

$a \in A$	a is contained in the set A
$\exists$	exists
$\forall$	for all
C_4/C_6	Knill's error-detection based fault-tolerance scheme using [[4, 2, 2]] and [[6, 2, 2]] codes
$O(g(x))$	$f(x)$ is $O(g(x))$ if $\exists x_0, \exists M > 0$ s.t. $\|f(x)\| \leq M\|g(x)\|$ for $x > x_0$
$O(10^{-n})$	shorthand for quantity roughly between 10^{-n} and 9×10^{-n}
$\mathcal{H}$	Hilbert space
$\|\psi\rangle$	vector in a Hilbert space
$\|0\rangle, \|1\rangle$	computational basis states for a qubit
$\langle\psi_1\|\psi_2\rangle$	inner product
$\|\|\psi\|\|$	norm of $\|\psi\rangle$
$\|\pm\rangle$	states $(\|0\rangle \pm \|1\rangle)/\sqrt{2}$
$\{0, 1\}^n$	set of n-bit strings
$\otimes$	tensor product
$A^{\otimes n}$	tensor product of A with itself n times
I	identity matrix
U	unitary matrix
U^T	matrix transpose of U
U^*	complex conjugate of U
$U^\dagger$	transpose conjugate
H	Hamiltonian $H = H^\dagger$ or Hadamard gate $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, or check matrix

$U(2^n)$	unitary group of $2^n \times 2^n$ unitary matrices
$\|\psi\rangle\langle\psi\|$	outer product
$g[q_1, \ldots, q_m]$	quantum gate g acting on an ordered list of qubits
$\oplus$	exclusive OR (XOR) or direct sum
$\Lambda(X)$ or $\Lambda(X)[1,2]$	controlled-NOT gate $\Lambda(X)\|ab\rangle = \|a(a \oplus b)\rangle$
$\Lambda(Z)$	controlled-Z gate $\Lambda(Z)\|ab\rangle = (-1)^{ab}\|ab\rangle$
$\Lambda^2(X)$	Toffoli gate $\Lambda^2(X)\|abc\rangle = \|ab(ab \oplus c)\rangle$
$\mathrm{diag}(a, b, \ldots, z)$	diagonal matrix with the given diagonal
K	Phase gate $\mathrm{diag}(1, i)$
T	$\pi/8$ gate $\mathrm{diag}(1, e^{i\pi/4})$
$\mathbf{L}(\mathcal{H})$	linear operators on $\mathcal{H}$
$\|\|A\|\|$	norm of an operator
X, Y, Z	Pauli matrices
$\langle A_1, A_2, \ldots, A_m\rangle$	group generated by A_i
$\mathcal{G}_n$	Pauli group on n qubits
$\mathrm{T}r\ A$	trace of a matrix or trace map on a field
$iZXZII$	shorthand for tensor product of Pauli matrices with phase
$[n]$	shorthand for the set of integers $\{1, 2, \ldots, n\}$
$[A, B]$	commutator of A and B, $AB - BA$; can be defined on sets
$\{A, B\}$	anticommutator of A and B, $AB + BA$
$f : A \rightarrow B$	a map f from A to B
$[A\|B]$	a block matrix or vector with an explicit separator
$\mathbf{a}$	binary vector
$\cdot$	dot product
$\odot$	symplectic inner product
$\mathcal{C}_2^{(n)}$	n-qubit Clifford group
G_1/G_2	factor group G_1 modulo G_2
$Sp(2n, 2)$	symplectic group
S	stabilizer group
$\hat{S}$	generating set of S
$\|S\|$	order of S
$C(S)$	stabilizer subspace associated with S

$\cup$,$\cap$	union, intersection
ρ	density matrix
$\mathrm{Tr}_A(\rho_{AB})$	partial trace over subsystem A
$\subseteq$	subset with possible equality
$F(\rho, \sigma)$	fidelity
$\mathcal{E}(\rho)$	quantum operation
(n, K, d)	binary classical code encoding K symbols into n bits with distance d
$[n, k, d]$	binary classical linear code encoding k bits into n bits with distance d
C	classical code
$C^{\perp}$	dual code to C
t	number of correctable errors
$\mathcal{C}$	quantum code
$((n, K, d))$	binary quantum code encoding K levels into n qubits with distance d
$[[n, k, d]]$	binary additive quantum code encoding k qubits into n qubits with distance d
$[[n, k, l, d]]$	binary subsystem code with l gauge qubits
$P_{C(S)}$	projector onto stabilizer code $C(S)$
$Z(S)$	centralizer of S in the Pauli group
$\bar{X}_i$, $\bar{Z}_i$	logical Pauli operators on a code
$GF(q)$, $\mathbb{F}_q$	finite field with $q = p^m$ elements, p prime
$:=$	definition
$\equiv$	equivalent
$\lvert\bar{\mathbf{a}}\rangle$	logical basis state a
$RM(r, m)$	Reed-Muller code of order r and length 2^m
$RM^*(r, m)$	punctured Reed-Muller code
$\lvert S\rangle$	state stabilized by S
$A^{\mathbf{c}}$ or $A(\mathbf{c})$	tensor product of A wherever $c_i = 1$ and I wherever $c_i = 0$
$\wedge$, $\vee$	Boolean AND, OR
$\bar{U}$	gate acting logically like U on a code space
$\mathbb{P}(\text{event})$	probability of an event
$\mathbb{P}(\text{event}\lvert\text{condition})$	conditional probability of an event
$\binom{C}{m}$	binomial coefficient C choose m
$\mathrm{Trans}_r(S)$	transversal gates on r blocks of $C(S)$

π_p	permutation where p is written using cycle notation
$C_k^{(n)}$	n qubit C_k hierarchy of quantum teleportation
$\mathcal{I}(S)$	generalized stabilizer of $C(S)$
$\mathrm{Aut}(S)$	full automorphism group of stabilizer code $C(S)$
$\ltimes$	semidirect product of groups
S_n	symmetric group on n items
$\mathrm{PAut}(S)$	permutation automorphisms of S
$\mathbb{F}[A]$	vector space spanned by elements of A over field $\mathbb{F}$
wt R	weight of a local operator R
$\mathrm{supp}(R)$	support of a local operator R
$\mathfrak{m}(S)$	union of minimal supports of S
$M(S)$	subgroup generated by all minimal elements of S
$\bar{\omega}$	complement of ω in $[n]$
ρ_ω	partial trace of ρ over all coordinates in $\bar{\omega}$
S_ω	elements of S with support strictly contained in ω
$[[n_1, k_1, d_1]] \otimes [[n_2, k_2, d_2]]$	shorthand for tensor product of two quantum codes
A_ω	number of nonidentity elements in S_ω
$SO(n)$	special orthogonal group of $n \times n$ matrices
$\subsetneq$	contained in but not equal to
$S \setminus S_j$	set difference, all elements in S not in S_j
$\mathbb{Z}$, $\mathbb{Q}$	integers, rationals
$C_2^\perp < C_1$	$C_2^\perp$ is a subspace of C_1
$\gcd(a, b)$	greatest common divisor of a and b
$\ll$	much less than
$\lvert \pm i\rangle$	$(\lvert 0\rangle \pm i\lvert 1\rangle)/\sqrt{2}$
$\propto$	proportional to
$\mathcal{P}_k$	polynomials of degree less than k in $\mathbb{F}_q[x]$
det A	determinant of A
$\mathrm{GRS}_k(\gamma, \mathbf{v})$	generalized Reed-Solomon code using $\mathcal{P}_k$ and evaluation points γ
$ev(f)$	evaluation of f at a set of points γ
$C_{\text{outer}} \circ C_{\text{inner}}$	recursive simulation of C_{inner} followed by C_{outer}
$\Omega(g(x))$	$f(x)$ is $\Omega(g(x))$ if $\exists x_0, \exists M > 0$ s.t. $\lvert f(x)\rvert \geq M\lvert g(x)\rvert$ for $x > x_0$

Appendix B

Data Tables

$[[n,k,d]]$	L[a]	R[b]	CX/REC[c]	$p_1(p_{\text{mem}}=0, p_0=10^{-4})$	$p_1(p_{\text{mem}}=p_0=10^{-4})$	$p_{\text{th}}(p_{\text{mem}}=0)$	$p_{\text{th}}(p_{\text{mem}}=p_0)$
$[[5,1,3]]$	2	2	2,160	–	–	$(3.9 \pm 0.7) \times 10^{-5}$	$(2.5 \pm 0.4) \times 10^{-5}$
$[[5,1,3]]$	3	3	5,117	–	–	$(9.2 \pm 0.5) \times 10^{-5}$	$(3.7 \pm 0.3) \times 10^{-5}$
$[[5,1,3]]$	5	5	14,775	–	–	$(9.2 \pm 0.5) \times 10^{-5}$	$(3.3 \pm 0.6) \times 10^{-5}$
$[[5,1,3]]$	10	3	18,536	–	–	$(8.8 \pm 0.5) \times 10^{-5}$	$(4.3 \pm 0.3) \times 10^{-5}$
$[[5,1,3]]$	10	10	60,760	–	–	$(8 \pm 3) \times 10^{-5}$	$(3.0 \pm 0.6) \times 10^{-5}$
$[[7,1,3]]$	2	1	519	$(5.34 \pm 0.07) \times 10^{-4}$	$(7.05 \pm 0.08) \times 10^{-4}$	$(1.85 \pm 0.05) \times 10^{-5}$	$(1.46 \pm 0.05) \times 10^{-5}$
$[[7,1,3]]$	3	1	775	$(2.3 \pm 0.2) \times 10^{-5}$	$(4.5 \pm 0.2) \times 10^{-5}$	$(3.11 \pm 0.02) \times 10^{-4}$	$(1.98 \pm 0.01) \times 10^{-4}$
$[[7,1,3]]$	4	1	1,031	$(1.9 \pm 0.1) \times 10^{-5}$	$(3.7 \pm 0.2) \times 10^{-5}$	$(4.97 \pm 0.07) \times 10^{-4}$	$(2.56 \pm 0.06) \times 10^{-4}$
$[[7,1,3]]$	5	1	1,287	$(1.8 \pm 0.1) \times 10^{-5}$	$(4.1 \pm 0.2) \times 10^{-5}$	$(5.3 \pm 0.1) \times 10^{-4}$	$(2.58 \pm 0.06) \times 10^{-4}$
$[[9,1,3]]$	1	1	69	–	$(4.90 \pm 0.09) \times 10^{-5}$	$(2.6 \pm 0.1) \times 10^{-4}$	$(2.06 \pm 0.02) \times 10^{-4}$
$[[13,1,3]]$	3	1	1,501	–	–	$(1.59 \pm 0.04) \times 10^{-4}$	$(0.69 \pm 0.03) \times 10^{-4}$
$[[13,1,3]]$	4	1	1,997	–	–	$(3.81 \pm 0.07) \times 10^{-4}$	$(1.95 \pm 0.04) \times 10^{-4}$
$[[13,1,3]]$	5	1	2,493	–	$(4.3 \pm 0.2) \times 10^{-5}$	$(4.9 \pm 0.2) \times 10^{-4}$	$(2.30 \pm 0.08) \times 10^{-4}$
$[[13,1,3]]$	10	1	4,973	$(1.8 \pm 0.1) \times 10^{-5}$	$(3.9 \pm 0.2) \times 10^{-5}$	$(5.1 \pm 0.2) \times 10^{-4}$	$(2.54 \pm 0.07) \times 10^{-4}$
$[[13,1,3]]$	15	1	7,453	$(1.9 \pm 0.1) \times 10^{-5}$	$(3.9 \pm 0.2) \times 10^{-5}$	$(4.9 \pm 0.1) \times 10^{-4}$	$(2.63 \pm 0.07) \times 10^{-4}$
$[[15,1,3]]$	3	1	2,127	$(1.3 \pm 0.2) \times 10^{-4}$	$(4.5 \pm 0.2) \times 10^{-4}$	$(0.86 \pm 0.03) \times 10^{-4}$	$(0.33 \pm 0.05) \times 10^{-4}$
$[[15,1,3]]$	4	1	2,831	$(4.9 \pm 0.7) \times 10^{-5}$	$(1.0 \pm 0.1) \times 10^{-4}$	$(1.5 \pm 0.6) \times 10^{-4}$	$(1.0 \pm 0.2) \times 10^{-4}$
$[[15,1,3]]$	5	1	3,535	$(5.8 \pm 0.8) \times 10^{-5}$	$(1.0 \pm 0.1) \times 10^{-4}$	$(1.8 \pm 0.2) \times 10^{-4}$	$(1.0 \pm 0.2) \times 10^{-4}$
$[[23,1,7]]$	10	1	16,023	$(1.1 \pm 0.3) \times 10^{-7}$	$(1.2 \pm 0.6) \times 10^{-7}$	$(1.14 \pm 0.05) \times 10^{-3}$	$(1.09 \pm 0.01) \times 10^{-3}$
$[[23,1,7]]$	20	1	32,023	$(1.2 \pm 0.4) \times 10^{-7}$	$(9 \pm 4) \times 10^{-8}$	$(2.33 \pm 0.02) \times 10^{-3}$	$(1.97 \pm 0.02) \times 10^{-3}$
$[[23,1,7]]$	30	1	48,023	–	–	$(2.98 \pm 0.04) \times 10^{-3}$	$(2.25 \pm 0.03) \times 10^{-3}$
$[[23,1,7]]$	40	1	64,023	–	–	$(3.33 \pm 0.02) \times 10^{-3}$	$(2.19 \pm 0.04) \times 10^{-3}$
$[[23,1,7]]$	10	2	28,023	$(4 \pm 1) \times 10^{-8}$	$(3 \pm 2) \times 10^{-8}$	$(5.76 \pm 0.09) \times 10^{-4}$	$(5.48 \pm 0.09) \times 10^{-4}$
$[[23,1,7]]$	20	2	56,023	$(5 \pm 1) \times 10^{-8}$	$\approx< 4 \times 10^{-8}$ [d]	$(1.23 \pm 0.01) \times 10^{-3}$	$(1.15 \pm 0.01) \times 10^{-3}$
$[[23,1,7]]$	30	2	84,023	–	–	$(1.628 \pm 0.006) \times 10^{-3}$	$(1.487 \pm 0.003) \times 10^{-3}$
$[[23,1,7]]$	40	2	112,023	–	–	$(1.95 \pm 0.01) \times 10^{-3}$	$(1.77 \pm 0.02) \times 10^{-3}$

Table B.1: Complete tabulation of code survey data, part 1

[a] for $[[5,1,3]]$ this parameter is NB.
[b] for $[[5,1,3]]$ this parameter is NC.
[c] for $[[5,1,3]]$ this parameter is the number of CNOT gates in a T_3 rectangle
[d] one failure in 5×10^7 samples

$[[n,k,d]]$	L	R[a]	CX/Rec	$p_1(p_{\text{mem}}=0, p_0=10^{-4})$	$p_1(p_{\text{mem}}=p_0=10^{-4})$	$p_{\text{th}}(p_{\text{mem}}=0)$	$p_{\text{th}}(p_{\text{mem}}=p_0)$
$[[23,1,7]]$	10	3	40,023	$(4 \pm 2) \times 10^{-8}$	$(3 \pm 1) \times 10^{-8}$	$(3.72 \pm 0.05) \times 10^{-4}$	$(3.45 \pm 0.05) \times 10^{-4}$
$[[23,1,7]]$	20	3	80,023	–	–	$(8.03 \pm 0.05) \times 10^{-4}$	$(7.67 \pm 0.05) \times 10^{-4}$
$[[23,1,7]]$	30	3	120,023	–	–	$(1.095 \pm 0.003) \times 10^{-3}$	$(1.036 \pm 0.008) \times 10^{-3}$
$[[23,1,7]]$	40	3	160,023	–	–	$(1.366 \pm 0.007) \times 10^{-3}$	$(1.280 \pm 0.009) \times 10^{-3}$
$[[25,1,5]]$	4	1	1,465	–	$(1.2 \pm 0.7) \times 10^{-6}$	$(8.6 \pm 0.2) \times 10^{-4}$	$(7.44 \pm 0.05) \times 10^{-4}$
$[[25,1,5]]$	5	1	1,825	–	$(1.0 \pm 0.1) \times 10^{-6}$	$(1.13 \pm 0.02) \times 10^{-3}$	$(9.74 \pm 0.07) \times 10^{-4}$
$[[25,1,5]]$	6	1	2,185	–	$(1.08 \pm 0.08) \times 10^{-6}$	$(1.16 \pm 0.02) \times 10^{-3}$	$(1.034 \pm 0.008) \times 10^{-3}$
$[[25,1,5]]$	7	1	2,545	–	$(1.1 \pm 0.2) \times 10^{-6}$	$(1.17 \pm 0.04) \times 10^{-3}$	$(1.01 \pm 0.04) \times 10^{-3}$
$[[41,1,5]]$	5	1	11,321	$(7.3 \pm 0.8) \times 10^{-6}$	$(2.39 \pm 0.05) \times 10^{-4}$	$(1.86 \pm 0.02) \times 10^{-4}$	$(7.9 \pm 0.1) \times 10^{-5}$
$[[41,1,5]]$	10	1	22,601	$[3 \times 10^{-7}]$[b]	$[7 \times 10^{-7}]$	$(7.44 \pm 0.03) \times 10^{-4}$	$(3.44 \pm 0.01) \times 10^{-4}$
$[[41,1,5]]$	15	1	33,881	–	–	$(1.224 \pm 0.003) \times 10^{-3}$	$(5.55 \pm 0.02) \times 10^{-4}$
$[[41,1,5]]$	20	1	45,161	–	–	$(1.577 \pm 0.004) \times 10^{-3}$	$(7.61 \pm 0.02) \times 10^{-4}$
$[[41,1,5]]$	30	1	67,721	–	–	$(2.06 \pm 0.01) \times 10^{-3}$	$(1.008 \pm 0.008) \times 10^{-3}$
$[[47,1,11]]$	10	1	52,527	–	–	$(3.25 \pm 0.04) \times 10^{-4}$	$(2.15 \pm 0.04) \times 10^{-4}$
$[[47,1,11]]$	20	1	105,007	–	–	$(6.89 \pm 0.05) \times 10^{-4}$	$(4.79 \pm 0.03) \times 10^{-4}$
$[[47,1,11]]$	30	1	157,487	$(1.6 \pm 0.9) \times 10^{-7}$	–	$(9.51 \pm 0.04) \times 10^{-4}$	$(6.45 \pm 0.03) \times 10^{-4}$
$[[49,1,9]]$	5	2	61,549	–	–	$(1.02 \pm 0.02) \times 10^{-4}$	$(5.4 \pm 0.1) \times 10^{-5}$
$[[49,1,9]]$	10	2	123,049	–	–	$(3.63 \pm 0.08) \times 10^{-4}$	$(2.23 \pm 0.04) \times 10^{-4}$
$[[49,1,9]]$	15	2	184,549	–	–	$(4.0 \pm 0.02) \times 10^{-4}$	$(3.20 \pm 0.08) \times 10^{-4}$
$[[49,1,9]]$	20	2	246,049	$(4 \pm 1) \times 10^{-6}$	–	$(4.2 \pm 0.3) \times 10^{-4}$	–
$[[49,1,7]]$	4	1	2,961	–	$(3.4 \pm 0.2) \times 10^{-6}$	$(4.73 \pm 0.09) \times 10^{-4}$	$(3.20 \pm 0.02) \times 10^{-4}$
$[[49,1,7]]$	6	1	4,417	–	$(2.8 \pm 0.2) \times 10^{-7}$	$(1.18 \pm 0.01) \times 10^{-3}$	$(8.7 \pm 0.2) \times 10^{-4}$
$[[49,1,7]]$	8	1	5,873	–	$(3.3 \pm 0.6) \times 10^{-7}$	$(1.41 \pm 0.02) \times 10^{-3}$	$(1.169 \pm 0.005) \times 10^{-3}$
$[[49,1,7]]$	9	1	6,601	–	$(2.2 \pm 0.7) \times 10^{-7}$	$(1.48 \pm 0.02) \times 10^{-3}$	$(1.224 \pm 0.005) \times 10^{-3}$
$[[49,1,7]]$	10	1	7,329	–	$(4.0 \pm 0.9) \times 10^{-7}$	$(1.42 \pm 0.03) \times 10^{-3}$	$(1.235 \pm 0.005) \times 10^{-3}$
$[[49,1,7]]$	11	1	8,057	–	$(2.5 \pm 0.2) \times 10^{-7}$	$(1.46 \pm 0.03) \times 10^{-3}$	$(1.241 \pm 0.006) \times 10^{-3}$
$[[49,1,7]]$	12	1	8,785	–	$(3 \pm 2) \times 10^{-7}$	$(1.46 \pm 0.02) \times 10^{-3}$	$(1.242 \pm 0.006) \times 10^{-3}$
$[[60,4,10]]$	10	1	86,460	–	–	$(1.129 \pm 0.004) \times 10^{-4}$	–
$[[60,4,10]]$	20	1	172,860	–	–	$(3.91 \pm 0.02) \times 10^{-4}$	$(2.20 \pm 0.04) \times 10^{-4}$

Table B.2: Complete tabulation of code survey data, part 2

[a] for $[[49,1,9]]$ this parameter is the number of preparation attempts for a 7-qubit encoded ancilla used in error detection

[b] The values in square brackets are extrapolated from a linear least-squares fit to the logarithm of $p_1(p_0)$

$[[n,k,d]]$	L	R	CX/Rec	$p_1(p_{\text{mem}}=0,p_0=10^{-4})$	$p_1(p_{\text{mem}}=p_0=10^{-4})$	$p_{\text{th}}(p_{\text{mem}}=0)$	$p_{\text{th}}(p_{\text{mem}}=p_0)$
$[[81,1,9]]$	4	1	4,977	–	$(4.4\pm0.7)\times10^{-5}$	$(2.1\pm0.2)\times10^{-4}$	$(1.407\pm0.005)\times10^{-4}$
$[[81,1,9]]$	6	1	7,425	–	$[1\times10^{-6}]$[a]	$(7.1\pm0.1)\times10^{-4}$	$(4.47\pm0.03)\times10^{-4}$
$[[81,1,9]]$	10	1	12,321	–	$[7\times10^{-7}]$	$(1.25\pm0.02)\times10^{-3}$	$(9.57\pm0.03)\times10^{-4}$
$[[81,1,9]]$	11	1	13,545	–	–	$(1.32\pm0.02)\times10^{-3}$	$(1.029\pm0.004)\times10^{-3}$
$[[81,1,9]]$	12	1	14,769	–	–	$(1.29\pm0.03)\times10^{-3}$	$(1.069\pm0.006)\times10^{-3}$
$[[81,1,9]]$	18	1	22,113	–	–	$(1.30\pm0.03)\times10^{-3}$	$(1.113\pm0.006)\times10^{-3}$
$[[81,1,9]]$	19	1	23,337	–	–	$(1.34\pm0.03)\times10^{-3}$	$(1.098\pm0.006)\times10^{-3}$
$[[81,1,9]]$	20	1	24,561	–	–	$(1.34\pm0.02)\times10^{-3}$	$(1.112\pm0.006)\times10^{-3}$
$[[85,1,7]]$	5	1	30,405	–	–	$(5.7\pm0.1)\times10^{-5}$	$(2.03\pm0.07)\times10^{-5}$
$[[85,1,7]]$	10	1	60,725	–	–	$(2.48\pm0.01)\times10^{-4}$	$(1.03\pm0.04)\times10^{-4}$
$[[85,1,7]]$	15	1	91,045	–	–	$(4.18\pm0.05)\times10^{-4}$	$(1.76\pm0.02)\times10^{-4}$
$[[85,1,7]]$	20	1	121,365	–	$[2\times10^{-7}]$	$(5.59\pm0.04)\times10^{-4}$	$(2.32\pm0.02)\times10^{-4}$

Table B.3: Complete tabulation of code survey data, part 3

[a]The values in square brackets are extrapolated from a linear least-squares fit to the logarithm of $p_1(p_0)$

$[[n,k,d=\ell]]$	CX/Rec	$p_1(p_{\rm mem}=p_0=10^{-4})$	$p_{\rm th}(p_{\rm mem}=p_0)$
$[[41,1,5]]$	1,481	$(1.7\pm0.1)\times10^{-4}$	$(6.8\pm0.6)\times10^{-5}$
$[[85,1,7]]$	4,453	$(5\pm2)\times10^{-5}$	$(2.3\pm0.2)\times10^{-4}$
$[[145,1,9]]$	9,937	$(2\pm1)\times10^{-5}$	$(4.5\pm0.2)\times10^{-4}$
$[[221,1,11]]$	18,701	$[8\times10^{-6}]$[a]	$(6.6\pm0.2)\times10^{-4}$
$[[313,1,13]]$	31,513	$[8\times10^{-6}]$	$(9.0\pm0.4)\times10^{-4}$

Table B.4: Surface code data using Shor-EC and a transversal CNOT as in [DKLP02], taking ℓ syndromes for an $\ell\times\ell$ code EC.

[a]The values in square brackets are extrapolated from a linear least-squares fit to the logarithm of $p_1(p_0)$

$[[n,k,d]]$	family	p_{th}(perfect ancilla)
$[[5,1,3]]$		$(2.0 \pm 0.1) \times 10^{-4}$
$[[7,1,3]]$	doubly-even dual-containing	$(9.1 \pm 0.2) \times 10^{-4}$
$[[9,1,3]]$	Bacon-Shor	$(6.0 \pm 0.9) \times 10^{-4}$
$[[13,1,3]]$	surface	$(8.8 \pm 0.1) \times 10^{-4}$
$[[21,3,5]]$	polynomial	$< 10^{-5}$
$[[23,1,7]]$	dual-containing	$(5.34 \pm 0.04) \times 10^{-3}$
$[[25,1,5]]$	Bacon-Shor	$(1.88 \pm 0.04) \times 10^{-3}$
$[[41,1,5]]$	surface	$(3.8 \pm 0.3) \times 10^{-3}$
$[[47,1,11]$	doubly-even dual-containing	$(7.67 \pm 0.03) \times 10^{-3}$
$[[49,1,7]]$	Bacon-Shor	$(2.56 \pm 0.05) \times 10^{-3}$
$[[49,1,9]]$	doubly-even dual-containing	$(4.8 \pm 0.2) \times 10^{-3}$
$[[60,4,10]]$	polynomial	$(1.88 \pm 0.04) \times 10^{-3}$
$[[81,1,9]]$	Bacon-Shor	$(2.88 \pm 0.04) \times 10^{-3}$
$[[85,1,7]]$	surface	$(7.5 \pm 0.3) \times 10^{-3}$
$[[121,1,11]]$	Bacon-Shor	$(2.83 \pm 0.07) \times 10^{-3}$
$[[145,1,9]]$	surface	$(1.01 \pm 0.02) \times 10^{-2}$
$[[169,1,13]]$	Bacon-Shor	$(2.97 \pm 0.09) \times 10^{-3}$

Table B.5: Level-1 pseudo-thresholds for rectangles using Steane-EC with perfect (noiseless) ancilla, $n < 200$.

Bibliography

[Aar04] S. Aaronson. *Limits on efficient computation in the physical world.* Ph.D. thesis, University of California at Berkeley, Berkeley, CA, USA, 2004. `http://arxiv.org/abs/quant-ph/0412143`

[ABO97] D. Aharonov and M. Ben-Or. Fault tolerant quantum computation with constant error. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pp. 176–188, 1997. `http://arxiv.org/abs/quant-ph/9611025`

[ABO99] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error rate. *unpublished*, 1999. `http://arxiv.org/abs/quant-ph/9906129`

[AC07] P. Aliferis and A. W. Cross. Subsystem fault tolerance with the Bacon-Shor code. *Physical Review Letters*, **98**, 220502, 2007. `http://arxiv.org/abs/quant-ph/0610063`

[AC08] V. Aggarwal and A. R. Calderbank. Boolean functions, projection operators, and quantum error correcting codes. *IEEE Transactions on Information Theory*, **54**(4), 2008. `http://arxiv.org/abs/cs/0610159`

[AG03] D. Aharonov and D. Gottesman. unpublished, 2003.

[AGP06] P. Aliferis, D. Gottesman, and J. Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Information & Computation*, **6**, 97, 2006. `http://arxiv.org/abs/quant-ph/0504218`

[AGP08] P. Aliferis, D. Gottesman, and J. Preskill. Accuracy threshold for postselected quantum computation. *Quantum Information & Computation*, **8**, 181, 2008. `http://arxiv.org/abs/quant-ph/0703264`

[Ahn04] C. Ahn. *Extending quantum error correction: new continuous measurement protocols and improved fault-tolerant overhead.* Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2004.

[AKP04] V. Arvind, P. P. Kurur, and K. R. Parthasarathy. Nonstabilizer quantum codes from abelian subgroups of the error group. *Quantum Information & Computation*, **4**(6-7), 411–436, 2004. `http://arxiv.org/abs/quant-ph/0210097`

[AKP06] D. Aharonov, A. Kitaev, and J. Preskill. Fault-tolerant quantum computation with long-range correlated noise. *Physical Review Letters*, **96**(050504), 2006. `http://arxiv.org/abs/quant-ph/0510231`

[Ali07] P. Aliferis. *Level Reduction and the Quantum Threshold Theorem.* Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2007. `http://arxiv.org/abs/quant-ph/0703230`

[Ali08] P. Aliferis. Fibonacci schemes of fault-tolerant quantum computation. *to be published*, 2008. `http://arxiv.org/abs/0709.3603`

[AP07] P. Aliferis and J. Preskill. Fault-tolerant quantum computation against biased noise. *to be published*, 2007. `http://arxiv.org/abs/0710.1301`

[Art91] M. Artin. *Algebra.* Prentice Hall, Englewood Cliffs, N.J., 1991.

[Asc00] M. Aschbacher. *Finite group theory.* Cambridge University Press, Cambridge, UK, 2000.

[ATL01] A. Ashikhmin, M. A. Tsfasman, and S. Litsyn. Asymptotically good quantum codes. *Physical Review A*, **63**, 32311-1–032311-5, 2001. `http://arxiv.org/abs/quant-ph/0006061`

[Bac06] D. Bacon. Operator quantum error-correcting subsystems for self-correcting quantum memories. *Physical Review A*, **73**(012340), 2006. `http://arxiv.org/abs/quant-ph/0506023`

[BBBV97] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *Society of Industrial and Applied Mathematics Journal of Computing*, **26**(5), 1510–1523, 1997. `http://arxiv.org/abs/quant-ph/9701001`

[BCDP96] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill. Efficient networks for quantum factoring. *Physical Review A*, **54**, 1034, 1996. `http://arxiv.org/abs/quant-ph/9602016`

[BCL+06] H. Buhrman, R. Cleve, M. Laurent, N. Linden, A. Schrijver, and F. Unger. New limits on fault-tolerant quantum computation. *47th Annual IEEE Symposium on Foundations of Computer Science, 2006.*, pp. 411–419, 2006. `http://arxiv.org/abs/quant-ph/0604141`

[BDL07] S. Bravyi, D. DiVincenzo, and D. Loss. Polynomial-time algorithm for simulation of weakly interacting quantum spin systems. *to be published*, 2007. `http://arxiv.org/abs/0707.1894`

[BDS+03] M. D. Barrett, B. DeMarco, T. Shaetz, D. Leibfried, J. Britton, J. Chiaverini, W. M. Itano, B. Jelenkovic, J. D. Jost, C. Langer, T. Rosenband, and D. J. Wineland. Sympathetic cooling of $^9Be^+$ and $^{24}Mg^+$ for quantum logic. *Physical Review A*, **68**(4), 042302, 2003. `http://arxiv.org/abs/quant-ph/0307088`

[BDSW96] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters. Mixed state entanglement and quantum error correction. *Physical Review A*, **54**, 3824, 1996. `http://arxiv.org/abs/quant-ph/9604024`

[BE97] J. Bierbrauer and Y. Edel. New code parameters from Reed-Solomon subfield codes. *IEEE Transactions on Information Theory*, **43**(3), 1997. `http://citeseer.ist.psu.edu/bierbrauer96new.html`

[BK98] S. B. Bravyi and A. Y. Kitaev. Quantum codes on a lattice with boundary. *unpublished*, 1998. `http://arxiv.org/abs/quant-ph/9811052`

[BK05] S. Bravyi and A. Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, **71**(2), 022316, 2005. `http://link.aps.org/abstract/PRA/v71/e022316`

[BKRB08] J. Benhelm, G. Kirchmair, C. F. Roos, and R. Blatt. Towards fault-tolerant quantum computing with trapped ions. *submitted to the New Journal of Physics*, 2008. `http://arxiv.org/abs/0803.2798`

[BMD07] H. Bombin and M. A. Martin-Delgado. Quantum measurements and gates by code deformation. *unpublished*, 2007. `http://arxiv.org/abs/0704.2540`

[Bol61] B. Bolt. On the Clifford collineation, transform, and similarity groups III: generators and involutions. *Journal of the Australian Mathematical Society*, **2**, 334–344, 1961.

[BR05] P. O. Boykin and V. P. Roychowdhury. Reversible fault-tolerant logic. In *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 444–453, 2005.

[BRW61] B. Bolt, T. G. Room, and G. E. Wall. On the Clifford collineation, transform, and similarity groups I,II. *Journal of the Australian Mathematical Society*, **2**, 60–96, 1961.

[BV93] E. Bernstein and U. Vazirani. Quantum complexity theory. *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pp. 11–20, 1993. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.4771`

[CCC$^+$08] X. Chen, H. Chung, A. W. Cross, B. Zeng, and I. L. Chuang. Subsystem stabilizer codes cannot have a universal set of transversal gates for even one encoded qudit. *submitted to Physical Review A*, 2008. `http://arxiv.org/abs/0801.2360`

[CCS$^+$08] I. L. Chuang, A. W. Cross, G. Smith, J. A. Smolin, and B. Zeng. Codeword stabilized quantum codes: algorithm and structure. *submitted to Physical Review A*, 2008. `http://arxiv.org/abs/0803.3232`

[CDT07] A. W. Cross, D. P. DiVincenzo, and B. M. Terhal. A comparative code study for quantum fault-tolerance. *Submitted to Quantum Information & Computation*, 2007. `http://arxiv.org/abs/0711.1556`

[CFP02] A. M. Childs, E. Farhi, and J. Preskill. Robustness of adiabatic quantum computation. *Physical Review A*, **65**(012322), 2002. `http://arxiv.org/abs/quant-ph/0108048`

[CGS02] C. Crepeau, D. Gottesman, and A. Smith. Secure multi-party quantum computation. *Proceedings of 34th Annual ACM Symposium on Theory of Computing*, 2002. `citeseer.ist.psu.edu/article/crepeau02secure.html`

[COI+03] D. Copsey, M. Oskin, F. Impens, T. Metodiev, A. Cross, F. T. Chong, I. L. Chuang, and J. Kubiatowicz. Toward a scalable, silicon-based quantum computing architecture. *Journal of Selected Topics in Quantum Electronics*, **9**(6), 1552–1569, 2003.

[CP01] R. Crandall and C. Pomerance. *Prime Numbers: A Computational Perspective.* Springer, New York, NY, 2001.

[CRB+] J. Cramwinckel, E. Roijackers, R. Baart, E. Minkes, L. Ruscio, R. Miller, T. Boothby, C. Tjhai, and D. Joyner. GAP package GUAVA. `http://sage.math.washington.edu/home/wdj/guava/`

[CRSS98] A. Calderbank, E. Rains, P. Shor, and N. Sloane. Quantum error correction via codes over $GF(4)$, 1998. `http://arxiv.org/abs/quant-ph/9608006`

[CS96] A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, **54**, 1098, 1996. `http://arxiv.org/abs/quant-ph/9512032`

[CSSZ08] A. Cross, G. Smith, J. A. Smolin, and B. Zeng. Codeword stabilized quantum codes. *to appear in Proceedings of the IEEE International Symposium on Information Theory*, 2008. `http://arxiv.org/abs/0708.1021`

[CTC+07] Y. H. Chen, T. K. Truong, Y. Chang, C. D. Lee, and S. H. Chen. Algebraic decoding of quadratic residue codes using Berlekamp-Massey algorithm. *Journal of Information Science and Engineering*, **23**(1), 127–145, 2007.

[CZ95] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, **74**(20), 1995.

[Dan05] L. E. Danielsen. On self-dual quantum codes, graphs, and Boolean functions. Master's thesis, University of Bergen, 2005. `http://arxiv.org/abs/quant-ph/0503236`

[Deu89] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, **425**(1868), 73–90, 1989.

[DHU+08] M. Dolev, M. Heiblum, V. Umansky, A. Stern, and D. Madalu. Observation of a quarter of an electron charge at the $\nu = 5/2$ quantum Hall state. *Nature*, **452**, 2008.

[DiV00] D. P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, **48**(9–11), 771–783, 2000. `http://arxiv.org/abs/quant-ph/0002077`

[DKLP02] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, **43**(9), 4452–4505, 2002. `http://arxiv.org/abs/quant-ph/0110143`

[dNDD04] M. Van den Nest, J. Dehaene, and B. De Moor. Graphical description of the action of local Clifford transformations on graph states. *Physical Review A*, **69**, 022316, 2004. `http://arxiv.org/abs/quant-ph/0308151`

[dNDM05] M. Van den Nest, J. Dehaene, and B. De Moor. Local unitary versus local Clifford equivalence of stabilizer states. *Physical Review A*, **71**(6), 062323, 2005.

[DO77] R. Dobrushin and E. Ortyukov. Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements. *Problems of Information Transmission*, **23**(2), 203–218, 1977.

[Eas07] B. Eastin. Fault-tolerant thresholds for encoded ancillae with homogeneous errors. *Physical Review A*, **75**(2), 022301, 2007.

[Eli58] P. Elias. Computation in the presence of noise. *IBM Journal of Research and Development*, **2**(4), 346–353, 1958.

[EP98] W. Evans and N. Pippinger. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, **44**(3), 1998.

[ES03] W. Evans and L. Schulman. On the maximum tolerable noise of k-input gates for reliable computation by formulas. *IEEE Transactions on Information Theory*, **49**(11), 2003.

[Eva94] W. S. Evans. *Information Theory and Noisy Computation.* Ph.D. thesis, University of California at Berkeley, Berkeley, CA, 1994.

[Fed89] T. Feder. Reliable computation by networks in the presence of noise. *IEEE Transactions on Information Theory*, **35**(3), 1989.

[Fle07] A. S. Fletcher. *Channel-adapted quantum error correction.* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 2007. `http://arxiv.org/abs/0706.3400`

[FM01] M. Freedman and D. Meyer. Projective plane and planar quantum codes, 2001. http://arxiv.org/abs/quant-ph/9810055

[For66] G. D. Forney. *Concatenated codes.* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1966.

[Fow05] A. G. Fowler. *Towards Large-Scale Quantum Computation.* Ph.D. thesis, University of Melbourne, Melbourne, Australia, 2005. http://arxiv.org/abs/quant-ph/0506126

[FSG08] A. G. Fowler, A. M. Stephens, and P. Groszkowski. High threshold universal quantum computation on the surface code. *to be published*, 2008. http://arxiv.org/abs/0803.0272

[FX08] K. Feng and C. Xing. A new construction of quantum error-correcting codes. *Transactions of the American Mathematical Society*, **360**(4), 2007–2019, 2008.

[Gab74] H. N. Gabow. *Implementation of algorithms for maximum matching on nonbipartite graphs.* Ph.D. thesis, Stanford University, Stanford, CA, USA, 1974.

[GAP07] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007. http://www.gap-system.org

[Gar04] P. B. Garrett. *The mathematics of coding theory : information, compression, error correction, and finite fields.* Pearson/Prentice Hall, Upper Saddle River, NJ, 2004.

[GB97] M. Grassl and T. Beth. A note on non-additive quantum codes. *unpublished*, 1997. http://arxiv.org/abs/quant-ph/9703016

[GB99] M. Grassl and T. Beth. Quantum BCH codes. *unpublished*, 1999. http://arxiv.org/abs/quant-ph/9910060

[GC99] D. Gottesman and I. L. Chuang. Quantum teleportation is a universal computational primitive. *Nature*, **402**, 390, 1999. http://arxiv.org/abs/quant-ph/9908010

[GdN08] D. Gross and M. Van den Nest. The LU-LC conjecture, diagonal local operations and quadratic forms over $GF(2)$. *Quantum Information & Computation*, **8**(263), 2008. http://arxiv.org/abs/0707.4000

[GG94] P. Gacs and A. Gal. Lower bounds for the complexity of reliable Boolean circuits with noisy gates. *IEEE Transactions on Information Theory*, **40**(2), 1994. `http://citeseer.ist.psu.edu/gacs94lower.html`

[GGB99] M. Grassl, W. Geiselmann, and T. Beth. Quantum Reed-Solomon codes. In *Proceedings of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pp. 231–244, 1999. `http://arxiv.org/abs/quant-ph/9910059`

[GKR02] M. Grassl, A. Klappenecker, and M. Roetteler. Graphs, quadratic forms, and quantum codes. *Proceedings of the 2002 IEEE International Symposium on Information Theory*, 2002. `http://arxiv.org/abs/quant-ph/0703112`

[Got97] D. Gottesman. *Stabilizer Codes and Quantum Error Correction.* Ph.D. thesis, California Institute of Technology, Pasadena, CA, USA, 1997. `http://arxiv.org/abs/quant-ph/9705052`

[Got98a] D. Gottesman. The Heisenberg representation of quantum computers. *unpublished*, 1998. `http://arxiv.org/abs/quant-ph/9807006`

[Got98b] D. Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, **57**(1), 127–137, 1998. `http://arxiv.org/abs/quant-ph/9702029`

[Got00] D. Gottesman. Fault-tolerant quantum computation with local gates. *Journal of Modern Optics*, **47**, 333–345, 2000. `http://arxiv.org/abs/quant-ph/9903099`

[Got07] D. Gottesman. Quantum error-correction/CO781 class taught at the Perimeter Institute, 2007. `http://perimeterinstitute.ca/personal/dgottesman/QECC2007`

[GR08] M. Grassl and M. Roetteler. Quantum Goethals-Preparata codes. *to appear in Proceedings of the IEEE International Symposium on Information Theory*, 2008. `http://arxiv.org/abs/0801.2150`

[GRB03] M. Grassl, M. Roetteler, and T. Beth. Efficient quantum circuits for non-qubit quantum error-correcting codes. *International Journal of Foundations of Computer Science*, **14**, 757, 2003. `http://arxiv.org/abs/quant-ph/0211014`

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the 28th annual ACM symposium on the theory of computing*, pp. 212–219, 1996. `http://arxiv.org/abs/quant-ph/9605043`

[Gro05] D. Gross. *Finite phase space methods in quantum information.* Ph.D. thesis, University of Potsdam, 2005.

[HJ85] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, Cambridge [Cambridgeshire]; New York, 1985.

[HOS$^+$06] W. K. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, J. Rabchuk, and C. Monroe. T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation. *Applied Physics Letters*, **88**(034101), 2006. `http://arxiv.org/abs/quant-ph/0508097`

[HP03] W. C. Huffman and V. Pless. *Fundamentals of error-correcting codes.* Cambridge University Press, Cambridge, U.K.; New York, 2003.

[HW91] B. Hajek and T. Weller. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, **37**(2), 1991.

[Imp04] Francóis Impens. Fine-grained fault-tolerance : reliability as a fungible resource. Master's thesis, Massachusetts Institute of Technology, 2004.

[IY03] K. Iwama and S. Yamashita. Transformation rules for CNOT-based quantum circuits and their applications. *New Generation Computing*, **21**(4), 297–317, 2003.

[Jam98] D. James. Quantum dynamics of cold trapped ions, with application to quantum computation. *Applied Physics B*, **66**(2), 181–190, 1998. `http://arxiv.org/abs/quant-ph/9702053`

[JCWY07] Z. Ji, J. Chen, Z. Wei, and M. Ying. The LU-LC conjecture is false. *unpublished*, 2007. `http://arxiv.org/abs/0709.1266`

[JMV90] D. Jungnickel, A. J. Menezes, and S. A. Vanstone. On the number of self-dual bases of $GF(q^m)$ over $GF(q)$. *Proceedings of the American Mathematical Society*, **109**(1), 1990.

[Kit03] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, **303**, 2–30, 2003. `http://arxiv.org/abs/quant-ph/9707021`

[KKKS06] A. Ketkar, A. Klappenecker, S. Kumar, and P. K. Sarvepalli. Nonbinary stabilizer codes over finite fields. *IEEE Transactions on Information Theory*, **52**(11), 4892–4914, 2006. `http://arxiv.org/abs/quant-ph/0508070`

[KKM$^+$00] D. Kielpinski, B. E. King, C. J. Myatt, C. A. Sackett, Q. A. Turchette, W. M. Itano, C. Monroe, D. J. Wineland, and W. H. Zurek. Sympathetic cooling of trapped ions for quantum logic. *Physical Review A*, **61**(3), 032310, 2000. `http://arxiv.org/abs/quant-ph/9909035`

[KL00] E. Knill and R. Laflamme. A theory of quantum error-correcting codes. *Physical Review Letters*, **84**, 2525, 2000. `http://arxiv.org/abs/quant-ph/9604034`

[KLMN01] E. Knill, R. Laflamme, R. Martinez, and C. Negrevergne. Benchmarking quantum computers: the five-qubit error correcting code. *Physical Review Letters*, **86**(25), 5811–5814, 2001.

[KLZ96] E. Knill, R. Laflamme, and W. Zurek. Threshold accuracy for quantum computation. *unpublished*, 1996. `http://arxiv.org/abs/quant-ph/9610011`

[KLZ98] E. Knill, R. Laflamme, and W. H. Zurek. Resilient Quantum Computation. *Science*, **279**(5349), 342–345, 1998. `http://arxiv.org/abs/quant-ph/9702058`

[KMW02] D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, **417**, 2002.

[Kni04a] E. Knill. Fault-tolerant postselected quantum computation: schemes. *unpublished*, 2004. `http://arxiv.org/abs/quant-ph/0402171`

[Kni04b] E. Knill. Fault-tolerant postselected quantum computation: threshold analysis. *unpublished*, 2004. `http://arxiv.org/abs/quant-ph/0404104`

[Kni05a] E. Knill. Quantum computing with realistically noisy devices. *Nature*, **434**(7029), 39–44, 2005. `http://arxiv.org/abs/quant-ph/0410199`

[Kni05b] E. Knill. Scalable quantum computation in the presence of large detected-error rates. *Physical Review A*, **71**(4), 042322, 2005. `http://arxiv.org/abs/quant-ph/0312190`

[KR00] A. Kapoor and R. Rizzi. Edge-coloring bipartite graphs. *Journal of Algorithms*, **34**(2), 390–396, 2 2000. `http://citeseer.ist.psu.edu/kapoor99edgecoloring.html`

[KRUdW08] J. Kempe, O. Regev, F. Unger, and R. de Wolf. Upper bounds on the noise threshold for fault-tolerant quantum computing. *to be published*, 2008. `http://arxiv.org/abs/0802.1464`

[KSV02] A. Yu Kitaev, A. Shen, and M. N. Vyalyi. *Classical and quantum computation.* American Mathematical Society, Providence, R.I., 2002.

[LDM$^+$03] D. Leibfried, B. DeMarco, V. Meyer, D. Lucas, M. Barrett, J. Britton, W. M. Itano, B. Jelenkovic, C. Langer, T. Rosenband, and D. J. Wineland. Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate. *Nature*, **422**, 412–415, 2003.

[LGA$^+$08] J. Labaziewicz, Y. Ge, P. Antohi, D. Leibrandt, K. R. Brown, and I. L. Chuang. Suppression of heating rates in cryogenic surface-electrode ion traps. *Physical Review Letters*, **100**(1), 013001, 2008. `http://arxiv.org/abs/0706.3763`

[LKOW07] D. Leibfried, E. Knill, C. Ospelkaus, and D. J. Wineland. Transport quantum logic gates for trapped ions. *Physical Review A*, **76**(032324), 2007. `http://arxiv.org/abs/0707.3646`

[LMPZ96] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek. Perfect quantum error correcting code. *Physical Review Letters*, **77**(1), 198–201, 1996. `http://arxiv.org/abs/quant-ph/9602019`

[LNCY97] D. W. Leung, M. A. Nielsen, I. L. Chuang, and Y. Yamamoto. Approximate quantum error correction can lead to better codes. *Physical Review A*, **56**, 2567–2573, 1997. `http://arxiv.org/abs/quant-ph/9704002`

[LS07] R. Lang and P. Shor. Nonadditive quantum error correcting codes adapted to the amplitude damping channel. *to be published*, 2007. `http://arxiv.org/abs/0712.2586`

[Met06] R. D. Van Meter III. *Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm.* Ph.D. thesis, Keio University, Fujisawa-shi, Kanagawa, Japan, 2006. `http://arxiv.org/abs/quant-ph/0607065`

[MK97] M. M. Mano and C. R. Kime. *Logic and computer design fundamentals.* Prentice Hall, New Jersey, 1997.

[MMM04] D. J. C. MacKay, G. Mitchison, and P. L. McFadden. Sparse-graph codes for quantum error correction. *IEEE Transactions on Information Theory*, **50**(10), 2315–2330, 2004. `http://aps.arxiv.org/abs/quant-ph/0304161`

[Mon08] C. Monroe. Periodic table of ions. World Wide Web electronic publication, 2008. `http://www.iontrap.umd.edu/research_info/ioncatalog/`

[MTC$^+$05a] T. Metodi, D. Thaker, A. Cross, I. Chuang, and F. Chong. A general purpose architectural layout for arbitrary quantum computations. *SPIE International Society for Optical Engineering, Defense and Security Symposium*, 2005.

[MTC$^+$05b] T. Metodi, D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang. A quantum logic array microarchitecture: scalable quantum data movement and computation. *The 38th Annual IEEE/ACM International Symposium on Microarchitecture*, 2005. `http://www.arxiv.org/abs/quant-ph/0509051`

[MTC$^+$06] T. Metodiev, D. Thaker, A. Cross, F. Chong, and I. Chuang. Scheduling physical operations in a quantum information processor. *SPIE International Society for Optical Engineering, Defense and Security Symposium*, 2006.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, Cambridge; New York, 2000.

[NRS01] G. Nebe, E. M. Rains, and N. J. A. Sloane. The invariants of the Clifford groups. *Designs, Codes and Cryptography*, **24**, 99–122, 2001. `http://citeseer.ist.psu.edu/270928.html`

[OBK99] P. R. J. Ostergard, T. Baicheva, and E. Kolev. Optimal binary one-error-correcting codes of length 10 have 72 codewords. *IEEE Transactions on Information Theory*, **45**(4), 1229–1231, 1999. `http://citeseer.ist.psu.edu/osterg99optimal.html`

[Pip85] N. Pippenger. On networks of noisy gates. In *26th Annual Symposium on Foundations of Computer Science*, Volume 26, pp. 30–38, IEEE Press, New York, NY, 1985.

[Pip89] N. Pippenger. Invariance of complexity measures for networks with unreliable gates. *Journal of the Association for Computing Machinery*, **36**(3), 531–539, 1989.

[Pou05] D. Poulin. Stabilizer formalism for operator quantum error correction. *Physical Review Letters*, **95**, 230504, 2005. `http://arxiv.org/abs/quant-ph/0508131`

[Pre98] J. Preskill. Fault-tolerant quantum computation. In H. Lo, T. Spiller, and S. Popescu, editors, *Introduction to quantum computation and information*, World Scientific, Singapore; [River Edge], NJ, 1998. `http://arxiv.org/abs/quant-ph/9712048`

[Pre99] J. Preskill. Plug-in quantum software. *Nature*, **402**, 357–358, 1999.

[PST91] N. Pippenger, G.D. Stamoulis, and J.N. Tsitsiklis. On a lower bound for the redundancy of reliable networks with noisy gates. *IEEE Transactions on Information Theory*, **37**(3), 639–643, 1991.

[Rai99a] E. M. Rains. Nonbinary quantum codes. *IEEE Transactions on Information Theory*, **45**(6), 1827–1832, 1999. `http://arxiv.org/abs/quant-ph/9703048`

[Rai99b] E. M. Rains. Quantum codes of minimum distance two. *IEEE Transactions on Information Theory*, **45**(1), 266–271, 1999. `http://arxiv.org/abs/quant-ph/9704043`

[Rai99c] E. M. Rains. Quantum shadow enumerators. *IEEE Transactions on Information Theory*, **45**, 2361, 1999. `http://arxiv.org/abs/quant-ph/9611001`

[Raz04] A. Razborov. An upper bound on the threshold quantum decoherence rate. *Quantum Information & Computation*, **4**(3), 222–228, 2004. `http://arxiv.org/abs/quant-ph/0310136`

[RBB03] R. Raussendorf, D. E. Browne, and H. J. Briegel. Measurement-based quantum computation with cluster states. *Physical Review A*, **68**(022312), 2003. `http://arxiv.org/abs/quant-ph/0301052`

[RBKD$^+$02] M. A. Rowe, A. Ben-Kish, B. DeMarco, D. Leibfried, V. Meyer, J. Beall, J. Britton, J. Hughes, W. M. Itano, B. Jelenkovic, C. Langer, T. Rosenband, and D. J. Wineland. Transport of quantum states and separation of ions in a dual RF ion trap. *Quantum Information & Computation*, **2**(4), 257–271, 2002. `http://arxiv.org/abs/quant-ph/0205094`

[Rei04] B. W. Reichardt. Improved ancilla preparation scheme increases fault-tolerant threshold. *unpublished*, 2004. `http://arxiv.org/abs/quant-ph/0406025`

[Rei06a] B. W. Reichardt. *Error-detection-based quantum fault tolerance against discrete Pauli noise.* Ph.D. thesis, University of California at Berkeley, Berkeley, CA, 2006. `http://arxiv.org/abs/quant-ph/0612004`

[Rei06b] B. W. Reichardt. Quantum universality by distilling certain one- and two-qubit states with stabilizer operations. *unpublished*, 2006. `http://arxiv.org/abs/quant-ph/0608085`

[RHG07] R. Raussendorf, J. Harrington, and K. Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, **9**(6), 199, 2007. `http://arxiv.org/abs/quant-ph/0703143`

[RHSS97] E. M. Rains, R. H. Hardin, P. W. Shor, and N. J. A. Sloane. A nonadditive quantum code. *Physical Review Letters*, **79**(5), 953–954, 1997. `http://arxiv.org/abs/quant-ph/9703002`

[RLB$^+$07] R. Reichle, D. Leibfried, R. B. Blakestad, J. Britton, J. D. Jost, E. Knill, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Transport dynamics of single ions in segmented microstructured Paul trap arrays. *Elements of Quantum Information*, pp. 69–97, 2007. `http://arxiv.org/abs/quant-ph/0606237`

[RS89] R. Reischuk and B. Schmeltz. Area efficient methods to increase the reliability of combinatorial circuits. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 314–326, Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[RS91] R. Reischuk and B. Schmeltz. Reliable computation with noisy circuits and decision treesa general $n \log\, n$ lower bound. In *Proceedings of the 32nd annual symposium on Foundations of computer science*, pp. 602–611, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991.

[RV99] V. P. Roychowdhury and F. Vatan. On the existence of nonadditive quantum codes. *Quantum Computing and Quantum Communications*, **1509**, 325–336, 1999. `http://arxiv.org/abs/quant-ph/9710031`

[SBBK08] R. D. Somma, S. Boixo, H. Barnum, and E. Knill. Quantum simulations of classical annealing processes. *to be published*, 2008. `http://arxiv.org/abs/0804.1571`

[SCA$^+$06] K. Svore, A. Cross, A. Aho, I. Chuang, and I. Markov. Toward a software architecture for quantum computing design tools. *IEEE Computer*, **January**, 2006.

[SCCA06] K. M. Svore, A. W. Cross, I. L. Chuang, and A. V. Aho. A flow-map model for analyzing pseudothresholds in fault-tolerant quantum computing. *Quantum Information & Computation*, **6**(3), 2006. `http://arxiv.org/abs/quant-ph/0508176`

[Sch02] D. Schlingemann. Stabilizer codes can be realized as graph codes. *Quantum Information & Computation*, **2**(4), 307–323, 2002. `http://arxiv.org/abs/quant-ph/0111080`

[SCR$^+$06] S. Seidelin, J. Chiaverini, R. Reichle, J. J. Bollinger, D. Leibfried, J. Britton, J. H. Wesenberg, R. B. Blakestad, R. J. Epstein, D. B. Hume, J. D. Jost, C. Langer, R. Ozeri, N. Shiga, and D. J. Wineland. A microfabricated surface-electrode ion trap for scalable quantum information processing. *Physical Review Letters*, **96**(253003), 2006. `http://arxiv.org/abs/quant-ph/0601173`

[SDT07] K. M. Svore, D. P. DiVincenzo, and B. M. Terhal. Noise threshold for a fault-tolerant two-dimensional lattice architecture. *Quantum Information & Computation*, **7**(4), 2007. `http://arxiv.org/abs/quant-ph/0604090`

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, **27**, 379–423, 1948.

[Sho94] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Proceedings of the 35th annual symposium on foundations of computer science*, pp. 124–134, 1994. `http://arxiv.org/abs/quant-ph/9508027`

[Sho95] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, **52**(4), 2493–2496, 1995.

[Sho96] P. W. Shor. Fault-tolerant quantum computation. *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 56–65, 1996. `http://arxiv.org/abs/quant-ph/9605011`

[SHO$^+$06] D. Stick, W. K. Hensinger, S. Olmschenk, M. J. Madsen, K. Schwab, and C. Monroe. Ion trap in a semiconductor chip. *Nature Physics*, **2**, 36–39, 2006. http://arxiv.org/abs/quant-ph/0601052

[SI05] A. M. Steane and B. Ibinson. Fault-tolerant logical gate networks for Calderbank-Shor-Steane codes. *Physical Review A*, **72**(5), 052335, 2005. http://arxiv.org/abs/quant-ph/0311014

[SM99] A. Sorensen and K. Molmer. Quantum computation with ions in thermal motion. *Physical Review Letters*, **82**, 1971–1974, 1999. http://arxiv.org/abs/quant-ph/9810039

[SSW07] J. A. Smolin, G. Smith, and S. Wehner. A simple family of non-additive quantum codes. *Physical Review Letters*, **99**, 130505, 2007. http://arxiv.org/abs/quant-ph/0701065

[Ste96] A. Steane. Multiple-Particle Interference and Quantum Error Correction. *Royal Society of London Proceedings Series A*, **452**, 2551–2577, 1996. http://arxiv.org/abs/quant-ph/9601029

[Ste97] A. M. Steane. Active stabilization, quantum computation, and quantum state synthesis. *Physical Review Letters*, **78**(11), 2252–2255, 1997. http://arxiv.org/abs/quant-ph/9611027

[Ste99a] A. M. Steane. Efficient fault-tolerant quantum computing. *Nature*, **399**(6732), 124–126, 1999. http://arxiv.org/abs/quant-ph/9809054

[Ste99b] A. M. Steane. Quantum Reed-Muller codes. *IEEE Transactions on Information Theory*, **45**(5), 1701–1703, 1999. http://arxiv.org/abs/quant-ph/9608026

[Ste02] A. M. Steane. Fast fault-tolerant filtering of quantum codewords. *unpublished*, 2002. http://arxiv.org/abs/quant-ph/0202036

[Ste03] A. M. Steane. Overhead and noise threshold of fault-tolerant quantum error correction. *Physical Review A*, **68**(4), 042322, 2003. http://arxiv.org/abs/quant-ph/0207119

[TB05] B. M. Terhal and G. Burkard. Fault-tolerant quantum computation for local non-Markovian noise. *Physical Review A*, **71**, 012336, 2005. `http://arxiv.org/abs/quant-ph/0402104`

[TIC$^+$05] D. D. Thaker, F. Impens, I. L. Chuang, R. Amirtharajah, and F. T. Chong. Recursive TMR: Scaling fault tolerance in the nanoscale era. *IEEE Design and Test of Computers*, **22**(4), 298–305, 2005.

[TMC$^+$06] D. Thaker, T. Metodiev, A. Cross, I. Chuang, and F. Chong. Quantum memory hierarchies: efficient designs to match available parallelism in quantum computing. *The 33rd Annual International Symposium on Computer Architecture*, 2006. `http://www.arxiv.org/abs/quant-ph/0604070`

[Tof80] T. Toffoli. Reversible computing. *Automata, Languages, and Programming*, pp. 632–644, 1980.

[Tur37] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **s2-42**(1), 230–265, 1937.

[vN56] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pp. 43–98. Princeton University Press, 1956.

[vS08] W. van Dam and I. Shparlinski. Classical and quantum algorithms for exponential congruences. *to be published*, 2008. `http://arxiv.org/abs/0804.1109`

[War99] H. N. Ward. An introduction to divisible codes. *Designs, Codes, and Cryptography*, **17**, 73–79, 1999.

[WC63] S. Winograd and J. D. Cowan. *Reliable computation in the presence of noise.* M.I.T. Press, Cambridge, Mass., 1963.

[WCP97] W.Bosma, J. Cannon, and C. Playoust. The MAGMA algebra system I: the user language. *Journal of Symbolic Computation*, **24**(3-4), 235–265, 1997.

[Wor94] T. Worsch. Lower and upper bounds for (sums of) binomial coefficients. Technical Report iratr-1994-31, Universität Karlsruhe, 1994. `citeseer.ist.psu.edu/worsch94lower.html`

[Yao93] A. C. Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pp. 352–361, 1993.

[YCLO07] S. Yu, Q. Chen, C. H. Lai, and C. H. Oh. Nonadditive quantum error-correcting code. *to be published*, 2007. `http://arxiv.org/abs/0704.2122`

[YCO07] S. Yu, Q. Chen, and C. H. Oh. Graphical quantum error-correcting codes. *to be published*, 2007. `http://arxiv.org/abs/0709.1780`

[ZCC07] B. Zeng, X. Chen, and I. L. Chuang. Semi-Clifford operations, structure of $\mathcal{C}_k$ hierarchy, and gate complexity for fault-tolerant quantum computation. *to be published*, 2007. `http://arxiv.org/abs/0712.2084`

[ZCCC07] B. Zeng, H. Chung, A. W. Cross, and I. L. Chuang. Local unitary versus local Clifford equivalence of stabilizer and graph states. *Physical Review A*, **75**(3), 2007. `http://arxiv.org/abs/quant-ph/0611214`

[Zur81] W. H. Zurek. Pointer basis of quantum apparatus: Into what mixture does the wave packet collapse? *Physical Review D*, **24**(6), 1516–1525, 1981.

[Zur82] W. H. Zurek. Environment-induced superselection rules. *Physical Review D*, **26**(8), 1862–1880, 1982.