Fault-Tolerance Testing on Small Quantum Error-Correcting Codes

Thesis

submitted in partial fulfilment of the
requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

Author:	BSc. Y.Yang
Student ID:	4738896
Supervisor:	Dr. C.G.Almudever
	MSc. L.Lao

December, 2019

Quantum and Computer Engineering Department,
Faculty of Electrical Engineering, Mathematics & Computer Science,
Delft University of Technology,
Delft, The Netherlands.

Abstract

Fault-Tolerance Testing on Small Quantum Error-Correcting Codes

Yaoling Yang
Master of Science
Department of EEMCS
TU Delft
2019

Quantum computers hold the promise to solve some hard problems that are intractable for even the most powerful current supercomputers. One of the most famous examples is Shor's algorithm for factorizing large numbers, which has exponential speedup compared to its best classical counterparts. However, running such an algorithm will require to build a large-scale quantum computer consisting of thousands or even millions of qubits that include quantum error correction (QEC) and fault-tolerant (FT) mechanisms.

Quantum computing is already a reality with the so-called Noisy Intermediate-Scale Quantum (NISQ) processors, some of them available in the cloud. Noisy refers to the imperfect control over the qubits and intermediate-scale to the relatively low number of quits (from fifty to a few hundred). Although current and near-term quantum devices will not have enough qubits for implementing large and fully corrected quantum computations, the use of small quantum error correction codes may extend the computation lifetime of NISQ devices. In this context and as a first step, it is important to test and demonstrate the fault-tolerance of these QEC codes.

In this thesis, we explore the fault-tolerance of two small quantum correction codes that are good candidates to be applied to NISQ processors, the [[4,2,2]] code and the [[7,1,3]] Steane code. To this purpose, by following the FT criterion prosed by Daniel Gottesman in 2016, we tested both codes using two simulators, the stabilizer formalism simulator that includes quite simple error models and a full density matrix simulator called quantumsim, which includes more realistic noise. The simulations are performed under reasonable noise parameter values. For the [[4,2,2]] code, 235 circuits are tested based on the two simulators. The results show that in the stabilizer formalism simulation, the FT criterion is satisfied for all circuits, while not fully satisfied in the full density matrix simulation. For the [[7,1,3]] Steane code, we use a parallel-flag error correction implementation which is tested using the full density matrix simulator. Our results show that without applying any QEC cycle, for all circuits (84 circuits for 1 logical qubit simulation and 452 circuits for 2 logical qubits simulation), the error rate of the encoded circuits is lower than the unencoded ones. Adding a quantum error correction (QEC) cycle will in general increase the error rate of the computation.

Acknowledgements

I would first like to thank my supervisor Dr.Carmen Garcia Almudever for offering me the chance to work on a quantum related thesis topic pursuing my interest as well as her guidance and patiently reviewing my thesis, and my daily supervisor Lingling Lao for her extremely patient guidance along half of my graduate study. I would also like to thank those people help me with my thesis project, especially Hans for helping me implement the OpenQL, Slava for answering my questions about quantumsim via email, Diogo for our discussions about quantumsim and OpenQL. I would also like to thank my friends in the QCA lab (Abid, Diogo, Hans, Peter, ...), and also my friends from the microelectronics track (Angqi, Daguang, Jiarui, Mengxin, Miao, Qiyou, Wencong, Xianglong) as well as all my friends encountered in TU Delft, for making my graduate life unforgettable.

Finally, I would like to express my gratitude to my family, especially to my parents for their unconditional support to my study and life.

Contents

1 Introduction **1**

1.1 A glimpse of quantum computing . . . 1

1.2 Fault-tolerant quantum computation . . . 2

1.2.1 The necessity for quantum error correction . . . 2

1.2.2 Quantum fault-tolerance and the threshold theorem . . . 3

1.2.3 The motivation for testing quantum fault-tolerance . . . 5

1.2.4 Thesis organization . . . 6

2 Quantum error correction **7**

2.1 Discretization of the errors . . . 7

2.2 The stabilizer formalism . . . 8

2.3 Syndrome extraction . . . 9

2.4 QEC codes used in the simulation . . . 11

2.4.1 The $[[4, 2, 2]]$ error-detecting code . . . 11

2.4.2 The [[7,1,3]] Steane code . . . 13

3 Simulation of QEC circuits **17**

3.1 The criterion for testing quantum fault-tolerance . . . 17

3.2 The stabilizer formalism simulation . . . 18

3.2.1 An overview of the stabilizer formalism simulation . . . 18

3.2.2 The depolarizing error model . . . 19

3.3 The full density-matrix simulation . . . 19

3.3.1 A brief introduction to quantumsim . . . 19

3.3.2 The noise model . . . 20

3.3.3 Scheduling . . . 21

4 Simulation results **23**

4.1 Simulation results using the stabilizer formalism simulator . . . 23

4.1.1 The results of simulating the [[4,2,2]] code . . . 23

4.2 Simulation results using the full density-matrix simulator . . . 26

4.2.1 The results of simulating the [[4,2,2]] code . . . 26

4.2.2 The results of simulating the [[7,1,3]] code . . . 29

5 Conclusion **34**
5.1 Conclusion 34
5.2 Future work 35

Bibliography **37**

A The test circuit families **39**

Chapter 1

Introduction

1.1 A glimpse of quantum computing

Quantum computing, the concept as originally proposed by Richard Feynman and Yuri Manin [1] in the 1980s, is a kind of computation model that harnesses more intrinsic principles of nature, such as superposition and entanglement. Unlike the classical bits which store information like 0 or 1, quantum bits can be 0 and 1 at the same time, which can be represented as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

Where $|0\rangle$ and $|1\rangle$ are states vectors representing the two states of a qubit, α and β are two complex numbers, where $|\alpha|^2$ and $|\beta|^2$ represents the probability of obtaining $|0\rangle$ and $|1\rangle$ respectively when measuring the state $|\psi\rangle$. For a single qubit, two states can be put in superposition, for two qubits, 4 states can be in superposition, the number of states can be put in superposition grows exponentially with the number of qubits. Entanglement is a kind of physical phenomenon that quantum systems can be closely correlated. In the standard interpretation of quantum mechanics, when two qubits get entangled with each other, the change of one qubit state will affect the other one instantaneously no matter how far they are from each other. An example of entangled states can be described by:

$$|\phi\rangle = \alpha|00\rangle + \beta|11\rangle \tag{1.2}$$

If a measurement is performed on one of the qubits, $|\phi\rangle$ will collapse to either $|00\rangle$ or $|11\rangle$. That is, though the measurement is only applied to one of the qubits, the state of the other qubit is affected.

The challenge of building a quantum computer is intriguing and meaningful, not only because a quantum computer is supposed to have a speedup over classical computers in some specific tasks, but also because quantum computers work directly and intrinsically based on quantum mechanics, which are fundamentally different from classical computers.

Quantum computing has many different hardware realizations. Superconducting realization and ion-trap based quantum computing are two popular ways that many companies are working on today. Scalability and mature fabrication technology make semiconductor quantum dots a serious candidate for large-scale quantum circuits. Majorana-based quantum computing may protect quantum information topologically thus has also attracted a lot of attention in recent years.

The community of quantum computing is getting bigger and bigger since the 1980s, not only physicists, electrical engineers, and computer scientists are joining this revolution. Many companies such as Google, IBM, Intel, Microsoft, Alibaba, etc. together with many universities become the main players in this field. However, the question that how far exactly we are from using quantum computers to solve practical problems remains unclear.

Quantum algorithms promise a speedup on solving some practical problems over known classical algorithms [2]. We can divide the algorithms into two categories. On the one aspect, simulating classical intractable complicate quantum systems using quantum computers, could help us get a deeper understanding of nature. With a practical quantum computer, many aspects of fundamental science such as condensed-matter physics, chemistry, etc. are likely to be driven forward. For example, solving the quantum many-body problem might help to find superconductors at room temperature. On the other aspect, the speed of many quantum algorithms may be able to surpass classical algorithms. The Shor's algorithm, which harnesses the periodicity finding ability brought by quantum mechanics, can factor big numbers with exponential speedup over all known classical algorithms, which is of vital importance for the safety of RSA encryption [3]. Many other known quantum algorithms like Grover's algorithm also provide polynomial speedup over classical algorithms. In recent years, new quantum algorithms are being proposed constantly. For example, AW Harrow et al. [4] proposed a quantum algorithm for solving linear systems of equations, which starts a trend for finding effective quantum algorithms for machine learning, namely quantum machine learning. In general, quantum algorithms have the potentials to efficiently simulate quantum systems and might provide a speedup over many known classical algorithms. However, it should be made clear that, for many problems, it often remains unknown that whether there exists a classical algorithm that could outperform or at least be comparable to its quantum counterpart.

For the problem mentioned above, there is a related term called quantum supremacy, as originally popularized by John Preskill [5]. Since executing quantum algorithms to solve practical problems is not feasible using the quantum hardware we have today, many skeptics might doubt the power of quantum computers. Demonstrating the quantum supremacy is a goal to convince skeptics that quantum computer is indeed powerful [6]. To fulfill this, researchers have been working on the construction of a specific algorithm in order to make it quantum easy and classically hard [‡]. Running such an algorithm using quantum hardware in the Noisy Intermediate-Scale Quantum (NISQ) era [8], can be regarded as demonstrating quantum supremacy.

In summary, quantum computing is a new type of computation model that is inherently based on quantum mechanics. Not only building a quantum computer is supposed to be meaningful, but also, on the road of creating such a sophisticated machine, nature may likely reveal some secrets to us.

1.2 Fault-tolerant quantum computation

1.2.1 The necessity for quantum error correction

Unlike classical computing devices, which are reliable with a typical failure rate of less than 10^{-17} [9], the typical failure rate of a quantum gate on different hardware realizations today, is only about 1%. Such a high failure rate makes it impossible for large-scale quantum computations. This is because various errors happening during computations are likely to accumulate and propagate, which would

[‡]Note that the hardness here is still conditional, it is very hard to eliminate all the possibilities that any efficient classical algorithms might exist, see [7] for more details.

finally turn the computing results into useless. Besides, quantum states are usually intrinsically fragile. Even without applying gates, being exposed to the environment, quantum states are constantly suffering from decoherence, that is, the information stored by quantum states would be easily dissipated. Due to these reasons, the quantum computer was once thought to be impossible by some physicists. It was not until Shor [10] and Steane [11] proposed the first quantum error-correcting (QEC) code that resurrected the wish to build a practical quantum computer.

Quantum information can be properly protected by error correction schemes. However, due to the no-cloning theorem [12], qubits cannot be copied, and any direct measurement tries to read out the states would disturb and lose track of the original quantum information. In spite of these limitations, classical error correction schemes can still be adapted to the quantum case, protecting information by adding redundant information.

In quantum error correction, we encode the quantum information using more data qubits, and we use ancilla qubits to couple to the encoded system. By measuring the ancilla qubits, we obtain the error syndromes. Based on these syndromes, the errors that happened to the encoded system can be inferred, thus the recovery process can be performed and quantum information is protected. The further details of quantum error correction will be introduced in chapter 2.

1.2.2 Quantum fault-tolerance and the threshold theorem

In general, only with the help of quantum error correction, it is not enough to support reliable quantum computations. On the one hand, the error correction procedure itself introduces errors. On the other hand, the error happens at one qubit may propagate to other qubits through multi-qubit gates e.g. CNOT gate, as shown in Figure A. It is possible the error propagation would seriously degenerate the output result. Therefore, the quantum fault-tolerance, which is a protocol that takes care of the error propagation, was proposed.

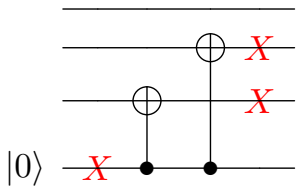


Figure 1.1: The error propagation of a CNOT gate. The X error happened to the qubit at the bottom propagates to the two qubits on the top through the CNOT gate.

The concept of fault-tolerance is not a new idea. In the early stage of classical computers, the concept of fault-tolerance was introduced. However, due to the high reliability of the classical hardware used in most cases today, the concept becomes less significant. For the quantum hardware we have today, it is indeed necessary to make the computation fault-tolerant (FT).

The quantum fault-tolerance can be defined as the property that, a single failure of one component in a procedure will cause at most one error in each encoded block of qubits output from the procedure [9]. The procedure here can refer to the fault-tolerant preparation procedure, the fault-tolerant syndrome extraction procedure, etc. An example of a quantum circuit diagram encoded with the FT protocol is shown in Figure 1.2.

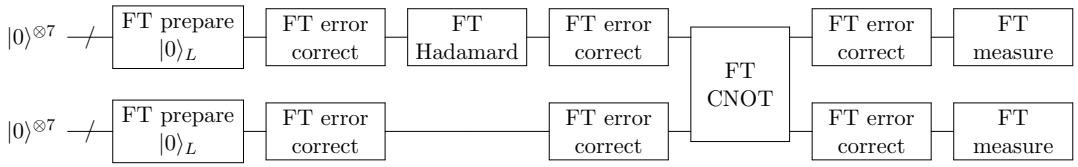
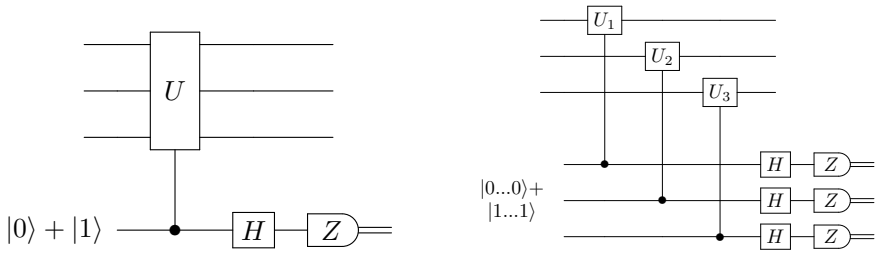


Figure 1.2: A block diagram of 2 logical qubits encoded in the Steane code with the FT protocol [9].

Each procedure in the diagram is necessary to be made fault-tolerant. For example, the syndrome extraction process in the error correction procedure should be fault-tolerant. The schematics of a Non-FT and a kind of FT implementation of the syndrome extraction process are shown in Figure 1.3. While Figure 1.3 (a) is a Non-FT implementation for the syndrome extraction process, in which the errors can propagate to the data qubits through the multi-qubit gate. Figure 1.3 (b) gives a way to fault-tolerantly implement the syndrome extraction process. The encoded system and the ancilla qubits are coupled individually. Therefore, one error that happened in the ancilla qubit can at most propagate to one qubit, which guarantees the fault-tolerance.



(a) A non-fault-tolerant implementation of the syndrome extraction process of U.

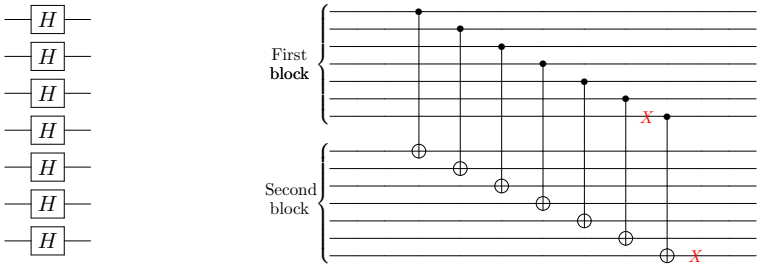
(b) A fault-tolerant implementation of the syndrome extraction process of U.

Figure 1.3: The schematics of a Non-FT and a FT implementation of a syndrome extraction process [13].

There are many ways for fault-tolerantly implementing the syndrome extraction process, for example, the Shor's style [14], the Steane's style [15], and the Knill's style [16]. A recent method called the flag-based quantum error correction will be used in this thesis, which requires fewer ancilla qubits than the other three methods mentioned above. The flag-based quantum error correction will be introduced in the next section.

The FT circuits also have to compute fault-tolerantly. In general, this requires all the logical operations to be fault-tolerant. One way for implementing the FT logical operations is by using the transversal gates. For steane code, all the basic logical gates are transversal. The transversal CNOT gate and Hadamard gate of the Steane code are shown in Figure 1.4. Note that with the transversal logical gates, during each logical operation procedure, one error can at most result in one fault at the output of each block.

One of the most remarkable results of fault-tolerant quantum computation and quantum error correction is the threshold theorem. The main idea of the threshold theorem is, there exists a threshold p_{th}, that when the failure probability p of the component in a circuit is below p_{th}, one can concatenate a quantum error-correcting code with another layer of quantum error-correcting code, namely increasing the code distance by the concatenation, to further increase the accuracy of components. By doing this



(a) The transversal Hadamard gate.

(b) The transversal CNOT gate.

Figure 1.4: The transversal Hadamard gate and CNOT gate of the Steane code.

concatenation so forth *ad infinitum*, the arbitrary low failure rate can be achieved.

Though the number of required components of circuits will increase as more concatenations go on, there is a proof [9] suggesting that the size of required gates in the encoded circuit is

$$O(\text{poly}(\log p(n)/\epsilon)p(n)) \tag{1.3}$$

while ϵ is the target accuracy of the quantum algorithm, $p(n)$ is a polynomial function of the size n of the circuit. It suggests the size of the encoded circuit is only polylogarithmically larger than the size of the original circuit. This result gives a stronger hope to large-scale quantum computation.

In summary, the threshold theorem is a remarkable result of fault-tolerant quantum computation and quantum error correction, it reveals that the possibility of the arbitrary low failure rate of components. However, in real hardware, and the threshold theorem is always accompanied by some assumptions.

1.2.3 The motivation for testing quantum fault-tolerance

The threshold theorem is a theoretic proof based on physically reasonable assumptions [9], such as high parallel operations, the low overhead of classical computation and communication, and a constant supply of fresh ancilla qubits, etc. Since things would naturally be far more complicated on real hardware, skeptics might doubt the effectiveness of the proof. Besides, the fault-tolerant protocol also remains unknown whether it can deal with realistic errors such as coherent error [17]. Therefore, it is important to see how the fault-tolerant protocol performs in a real experiment.

Though many experiments of testing quantum error-correcting codes have been carried out over the last few years, quantum fault-tolerance has not been widely tested. Daniel Gottesman suggested the necessity for demonstrating quantum fault-tolerance [17] in 2016. An explicit criterion for testing quantum fault-tolerance was proposed. After that, a few experiments were conducted to test fault-tolerance based on the $[[4, 2, 2]]$ code on IBM Q Experience devices (mainly on ibmqx5) [18, 19, 20].

Since we are in the NISQ era, it means the noise of the quantum systems is relatively high and the available number of qubits is limited. Therefore, in this thesis, we will focus on the small quantum error-correcting codes. We will simulate the fault-tolerance testing experiment for the $[[7, 1, 3]]$ code and the $[[4, 2, 2]]$ code in two simulation environments, one is based on the stabilizer formalism, the other one is called quantumsim [21, 22], which is a full density-matrix simulation environment. By using the stabilizer formalism, we can only simulate simple error models such as the depolarizing error model.

In contrast, quantumsim simulates more complicated errors such as relaxation and dephasing errors, two-qubit phase errors and measurement errors, etc. The parameters for the noise models used in our simulation are based on real experiments indicated in [21]. For the $[[7, 1, 3]]$ code, a flag-based quantum error correction [23] is adopted. Our simulation could offer insights for experiments to be conducted in the nearest future.

1.2.4 Thesis organization

This thesis is organized as follows: the second chapter introduces some basics about quantum error correction and quantum fault tolerance. The flag-based error correction method, as well as the QEC codes used in this thesis, are also introduced. The third chapter introduces the criterion to test quantum fault-tolerance, followed by an overview of the two simulation environments: the stabilizer circuits simulation environment and the full density-matrix simulation environment. The fourth chapter presents the simulation results. The last chapter gives the conclusion and future perspectives.

Chapter 2

Quantum error correction

2.1 Discretization of the errors

The errors we consider in quantum information are any unwanted transformations of density matrices living in Hilbert space. Quantum error correction is a process that bring a contaminated state $E_i|\psi\rangle$ back to $|\psi\rangle$, while E_i here refers to a so-called error operator. For a quantum state, errors are living in a continuous space, for example, a qubit living in Bloch sphere might undergo a deformation like rotating $\pi/233$ rad around y axis. To study the effect of noise, we can consider the interaction between the environment and the qubit system, a typical observation is the following [24]:

$$|\phi\rangle |\psi_0\rangle_e \rightarrow \sum_i (E_i|\phi\rangle) |\psi_i\rangle_e \tag{2.1}$$

while $|\phi\rangle$ is a set of qubits we care about, $|\psi_0\rangle_e$ represents the state of the environment (exclude $|\phi\rangle$) before the interaction, the arrow stands for "after interaction between the qubits and the environment", E_i is again the error operator applied on the set of qubits, and $|\psi_i\rangle_e$ represents the state of environment after the interaction. A key observation here is each E_i in formula 2.1 can be represented by a tensor product of Pauli matrices (together with an identity matrix for the error-free case):

$$\sigma_0 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.2}$$

$$\sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.3}$$

$$\sigma_2 = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2.4}$$

$$\sigma_3 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.5}$$

This is true because Pauli matrices are a complete set, it can represent any transformation that happened to qubits. For simplicity, we consider a single-qubit error as an example. For a single-qubit error happened to a set of qubits, the error operators can always be written in a linear combination of

Pauli matrices as [9]:

$$\sum_i E_i = e_{i0}I + e_{i1}X_1 + e_{i2}Z_1 + e_{i3}X_1Z_1 \tag{2.6}$$

while the I,X_1,Z_1 apply on the single qubit. Note that we can use the notion X_uZ_v to represent the tensor products of Pauli matrices, with u,v are both a set of the binary array representing if an X or Z error happened to qubits at certain positions. For example, $X_{10110}Z_{11010}$ represents applying $Y \otimes Z \otimes X \otimes Y \otimes I$ on the five qubits respectively.

For an error operator applying on a multi-qubit system, we can write it as:

$$E_i|\psi\rangle = X_uZ_v|\psi\rangle \tag{2.7}$$

that is, to correct this error, we can first correct X_u, then correct Z_v, it will bring us back to $|\psi\rangle$. Note that this is a remarkable result of quantum error correction, it suggests that being able to correct a discrete error set $\{X, Z\}$ is enough for fighting against any errors living in the continuous space. This is also the reason that why the depolarizing error model[‡] is favored by many authors in quantum error correction, a QEC code being good at correcting the discrete error set is likely to perform well in the presence of real hardware noise.

2.2 The stabilizer formalism

The stabilizer formalism, as originated from group theory, offers a convenient and significant way to study quantum states and quantum operations. With the stabilizer formalism, a large family of quantum error-correcting codes can be easily described, and the syndrome extraction procedure which will be discussed later can be more easily carried out. In this section, we will mainly focus on the basic ideas, for further details, please refer to [25].

The key idea of the stabilizer formalism can be illustrated through a simple example. Considering a bell state:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.8}$$

It is easy to see that such state satisfies:

$$X_1X_2|\psi\rangle = |\psi\rangle \tag{2.9}$$

$$Z_1Z_2|\psi\rangle = |\psi\rangle \tag{2.10}$$

This shows that applying X_1X_2 and Z_1Z_2 on $|\psi\rangle$ does not change the state $|\psi\rangle$. Besides, it can be verified that $|\psi\rangle$ is the only state (up to a global phase) that satifies the conditions. In other words, If we select X_1X_2 and Z_1Z_2, we already define a state that would not be affected when we apply the two operations. Formally, we can say X_1X_2 and Z_1Z_2 are the stabilizers of the state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, and $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is stabilized by X_1X_2 and Z_1Z_2,.

The above example gives a first impression of the stabilizer formalism, now let us consider a more general example [24]. For a n-qubit system, suppose $\mathcal{S} = \{M\}$ is a set containing all the error operators

[‡]In general, depolarizing error models refer to an error model that each component in a circuit undergoes X,Y,Z errors with certain probability p.

that commute with each other. Let $\mathcal{C} = \{|u\rangle\}$ be the vector space where all the elements are orthogonal to each other and satisfy the following equation:

$$M|u\rangle = |u\rangle \quad \forall u \in \mathcal{C}, \forall M \in \mathcal{S} \tag{2.11}$$

Then the set $\mathcal{C}$ is a quantum error-correcting code, M is the stabilizer of any $|u\rangle$ in $\mathcal{C}$, and $|u\rangle$ is called codewords or code vectors. Suppose we are encoding k logical qubits with n physical qubits, then the number of elements in $\mathcal{C}$ is 2^k, the dimension of the space spanned by the vectors in $\mathcal{C}$ is 2^k as well, which is different from the classical coding theory since superposition is allowed in quantum cases. The 2^k dimension space is a subspace of the 2^n dimensions Hilbert space of the whole quantum system. A logical state or encoded state $|\phi\rangle_L$ of the quantum error-correcting code is a general vector in the subspace, which can be expressed by the superposition of the codewords:

$$|\phi\rangle_L = \sum_{u \in \mathcal{C}} a_u |u\rangle \tag{2.12}$$

while the a_u is the normalization factor. The stabilizer formalism can also be used to describe some errors happen to the quantum system. In general, a error operator E is detectable if it anti-commutes with any stabilizers. Suppose E anti-commute with M_1, we have:

$$M_1(E|\phi\rangle) = -EM_1|u\rangle = -(E|u\rangle) \tag{2.13}$$

when the state $|\phi\rangle$ is transformed to $E|\phi\rangle$ by any noise, measuring M_1 will give an eigenvalue -1, which suggests the quantum state is no longer stabilized by $S = \{M\}$, errors must have happened, thus we detect the errors.

In the group S, there are $(n-k)$ elements $\{g_1, g_2, ..., g_{n-k}\}$ that can generate all the stabilizers in the group, such elements are called stabilizer generators, the generated group S can thus be written as $S =< g_1, g_2, ..., g_{n-k} >$. It can be proved [9] that to detect errors, it is enough to only measure the $(n-k)$ generators, such measurement is called syndrome measurement, which will be discussed in the next section.

At last, a natural question would be how to describe the logical operations under the stabilizer formalism. In general, the logical operations such as X_L or Z_L should be detected when we measure the stabilizer generators. Therefore, the logical operations are supposed to commute with all the stabilizer generators. In other words, if any error changes the logical state in a way that a logical operation can do, which cannot be detected or corrected by the QEC code, then it is a logical error.

2.3 Syndrome extraction

The stabilizer formalism can also be used to describe the process that how we extract the errors in a qubit system, such procedure is called error syndrome extraction or error syndrome measurement.

Suppose an encoded system $|u\rangle_L$ is contaminated by some error operators E_i, denoted as $E_i|u\rangle_L$, In order to determine E_i, we use $(n-k)$ ancilla qubits[‡] to couple with the whole system. Before any interaction happens between the ancilla qubits and the system, the state of the system can be represented

[‡]The number of required ancilla qubits $(n-k)$ here is for convenient illustration purpose, the exact number depends on how the error correction scheme is implemented.

as:

$$|0\rangle_a \sum_i (E_i|\phi\rangle)\,|\psi_i\rangle_e \tag{2.14}$$

where the ancilla qubits are coupled to the system through a tensor product. Then let's make the $(n-k)$ ancilla qubits interact with the system in a specific way. The interaction is done by applying some CNOT gates and Hadamard gates between the system and the ancilla qubits, a simple example is shown in Figure 2.1, where the ancilla qubits are coupled to measure the six stabilizer generators of the Steane code.

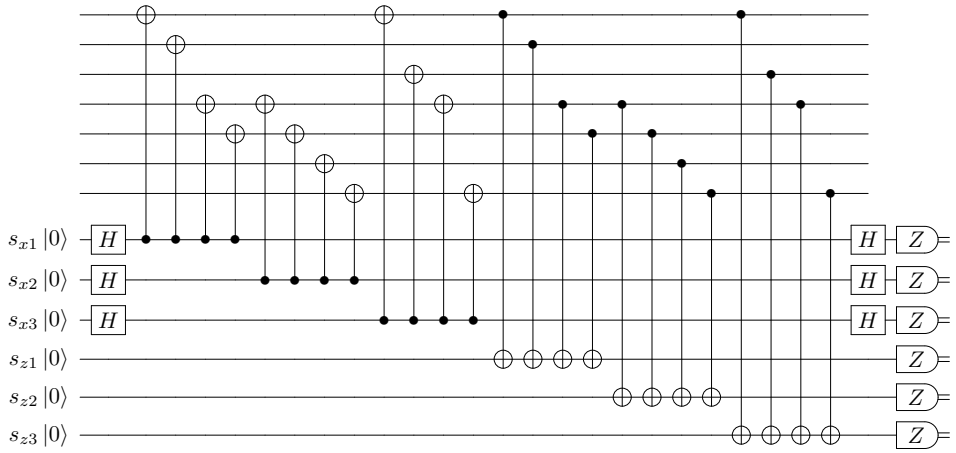


Figure 2.1: The (Non-FT) syndrome extraction circuit for the Steane code. The s_{x1},s_{x2},s_{x3} measure $XXIXXII$, $IIIXXXX$, $XIXXIIX$ respectively, the s_{z1},s_{z2},s_{z3} measure $ZZIZZII$, $IIIZZZZ$, $ZIZZIIZ$ respectively.

After the interaction, equation 2.14 becomes:

$$\sum_i |s_i\rangle_a\,(E_i|\phi\rangle_L)\,|\psi_i\rangle_e \tag{2.15}$$

During the interaction, the information of the error operators is passing to the ancilla qubits. Since the whole system is still in superposition, if we measure the ancilla qubits, due to the collapse only one term in the sum operator $\sum_i$ of equation 2.15 will survive. Suppose the i^{th} term is the survived term, the state of the whole system becomes:

$$|s_i\rangle_a\,(E_i|\phi\rangle_L)\,|\psi_i\rangle_e \tag{2.16}$$

where s_i is the final state of ancilla qubits, and we also obtain the information about s_i through the measurement. The s_i is called the error syndrome, which gives the information of E_i. The whole procedure is called syndrome extraction. Through syndrome extraction, the original errors in the system are digitized and collapsed to the measurement basis that we select. Based on the syndrome, the decoding process can be performed and we may deduce E_i and apply E_i to recover the state.

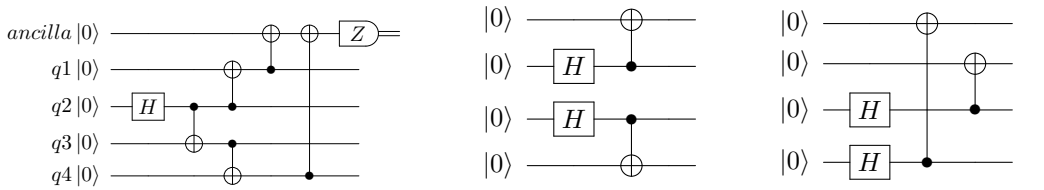
2.4 QEC codes used in the simulation

2.4.1 The $[[4, 2, 2]]$ error-detecting code

The $[[4, 2, 2]]$ code[†] encodes 2 logical qubits with 4 physical qubits, since its code distance is 2, it can only detect one error. The stabilizers of the [[4,2,2]] code is $XXXX$ and $ZZZZ$, the codewords are shown as follows:

$$\begin{aligned} |00\rangle_L &= \tfrac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \\ |01\rangle_L &= \tfrac{1}{\sqrt{2}}(|1100\rangle + |0011\rangle) \\ |10\rangle_L &= \tfrac{1}{\sqrt{2}}(|1010\rangle + |0101\rangle) \\ |11\rangle_L &= \tfrac{1}{\sqrt{2}}(|0110\rangle + |1001\rangle) \end{aligned} \tag{2.17}$$

The $[[4, 2, 2]]$ code can be encoded in different ways. Based on different encoding circuits, we can prepare different initial states. In this thesis, by following the suggestion of [17], three initial states are prepared, they are $|00\rangle_L$, $|0+\rangle_L$, and $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$. The encoding process is shown in Figure 2.2.



(a) The fault-tolerant encoding of $|00\rangle_L$.

(b) The fault-tolerant encoding of $|0+\rangle_L$.

(c) The fault-tolerant encoding of $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$

Figure 2.2: The fault-tolerant state preparation of $|00\rangle_L$, $|0+\rangle_L$, and $\frac{|00\rangle_L+|11\rangle_L}{2}$ for the $[[4, 2, 2]]$ code.

For the state $|00\rangle_L$, an ancilla qubit is needed to help to check whether the circuits are run in a fault-tolerant way. If a gate error occurs and results in a two-qubit error[‡] at the end of the procedure, measuring the ancilla qubit will result in -1, which suggests the prepared state is no longer $|00\rangle_L$, this run should be rejected. Otherwise, we say the state is successfully prepared, we can move to the next step. This ancilla qubit check guarantees the state preparation procedure is fault-tolerant, a single gate error can only result in a one-qubit error at the beginning of the next procedure.

For the other two initial states, the codewords can be written as a tensor product of two bell states with different numbering method of the qubits: $(|00\rangle + |11\rangle) \otimes (|00\rangle + |11\rangle)$ up to a global phase. It can be verified the encoding procedure is also fault-tolerant. This is because a two-qubit fault from a CNOT gate only affects one of the bell states. The error can always be attributed to the case one-qubit error happens or no error happens. For example, if the CNOT gate is followed by a two-qubit fault XX, it will not affect the bell state, since applying XX on a bell state remains in bell state. If the CNOT gate is followed by XZ, the bell state changes to $(|10\rangle - |01\rangle)$, which can be attributed to a Y error that happens to a single qubit (up to a global phase).

[†]QEC codes usually writes as [[n,k,d]] form, where the n is of physical qubits, k is of logical qubits and d is the code distance. A code with distance d is supposed to correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$, and detect up to $\left\lfloor \frac{d+1}{2} \right\rfloor$.

[‡]A single gate fault might cause a one-qubit error, two-qubit error, or four-qubit error at the end of the encoding circuit. However, the case that the one-qubit error left is fault-tolerant, the case that the two-qubit error left will flip the ancilla qubit, the case four-qubit error left changes the state of the four qubits, this is equivalent to leave the original state unchanged.

The [[4, 2, 2]] code allows FT logical operations such as $X1$, $X2$, $Z1$, $Z2$, H_1H_2SWAP, CZ, in this thesis we also use these operations. The methods for applying those gates on the state encoded in the [[4, 2, 2]] code and applying those gates on an unencoded state (2 physical qubits) are shown in Table 2.1.

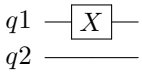
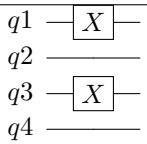
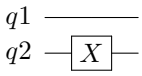
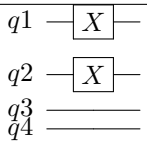
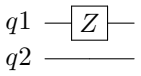
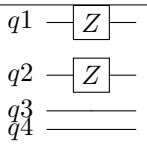
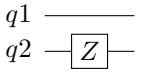
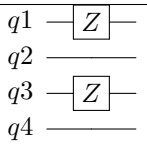
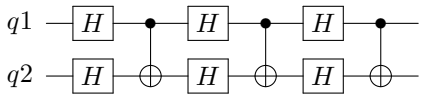
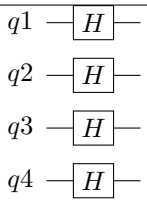
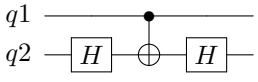
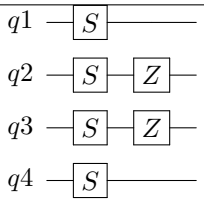
Gate	**Unencoded version**	**FT encoded version**
X1		
X2		
Z1		
Z2		
$H_1 \otimes H_2 \cdot \text{SWAP}$		
CZ		

Table 2.1: The fault-tolerant logical operations used in [[4,2,2]] code.

Since the [[4, 2, 2,]] code is an error-detecting code, it does not have the syndrome extraction procedure. Instead, after applying all the logical operations, we measure the 4 qubits. The state of the logical qubits is decided by comparing the measurement outcomes of the four data qubits to the codewords. If the measured state is inside the codeword, we obtain a valid measurement outcome. If the measured state is not a valid codeword, we know any error is detected, and we reject the result. This is called the post-selection process.

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

The Steane code encodes 1 logical qubit by using 7 physical qubits. The 6 staibilizer generators of Steane code are shown as follows:

Name	Operator
g_1	$X_1X_2X_4X_5$
g_2	$X_4X_5X_6X_7$
g_3	$X_1X_3X_4X_7$
g_4	$Z_1Z_2Z_4Z_5$
g_5	$Z_4Z_5Z_6Z_7$
g_6	$Z_1Z_3Z_4Z_7$

Table 2.2: The stabilizer generators of the Steane code $[[7, 1, 3]]$.

The logical states of the Steane code are shown as below[‡]:

$$\begin{aligned}|0\rangle_L = \frac{1}{\sqrt{8}}(&|0000000\rangle + |0111010\rangle + |0001111\rangle + |0110101\rangle \\ &+|1011001\rangle + |1100011\rangle + |1010110\rangle + |1101100\rangle)\end{aligned} \tag{2.18}$$

$$\begin{aligned}|1\rangle_L = \frac{1}{\sqrt{8}}(&|1111111\rangle + |1000101\rangle + |1110000\rangle + |1001010\rangle \\ &+|0100110\rangle + |0011100\rangle + |0101001\rangle + |0010011\rangle)\end{aligned} \tag{2.19}$$

For the Steane code, all the procedures including encoding, syndrome extraction, and logical operations can be implemented in a fault-tolerant way. In this thesis, the encoding procedure is done by performing a Non-FT encoding with 8 CNOT gates and 3 Hadamard gates shown in Figure 2.3, followed by a complete syndrome extraction and recovery procedure [26], which will be introduced later. The Non-FT encoding process is shown in Figure 2.3.

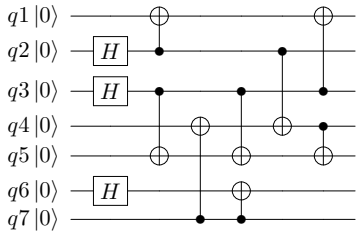
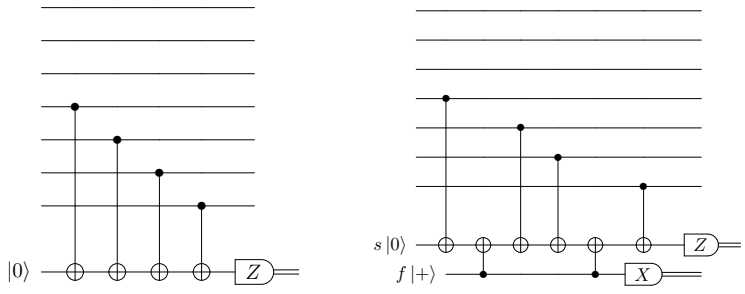


Figure 2.3: The Non-FT encoding of $|00\rangle_L$ of the Steane code.

The flag-based quantum error correction

As previously mentioned, there are different syndrome extraction schemes for quantum error correction, such as Shor's style [14], Steane's style [15], and Knill's style [16]. In this thesis, we do not use the methods mentioned above, since they usually have a large overhead of qubits. Instead, a recent fault-tolerant method called the flag-based quantum error correction, which was initially introduced by [27]

[‡]Note that in this thesis, in the state vector representation, the rightest qubit is numbered as q_1, the index of qubit increases from the right to the left side, while in the circuit, from the top to the bottom, the qubits are numbering from q_1 to q_n.



(a) The Non-FT version.

(b) The fault-tolerant version based on the flag technique.

Figure 2.4: Syndrome extraction circuits of $ZZZZ$ for the Steane code: (a) The Non-FT version and (b) the flag-based method.

is selected. This method requires fewer qubits than the mentioned methods, it uses some ancilla qubits acting as flags to detect correctable error to make the procedure fault-tolerant. The basic principle of the flag-based method can be illustrated in Figure 2.4.

In Figure 2.4 (a), without the flag qubit, it is easy to see that a Z error on the ancilla qubit can propagate to multiple qubits through the CNOT gates without detection. This problem can be fixed by adding an extra ancilla qubit working as the flag qubit shown in Figure 2.4 (b). Excluding the case that the error propagation does not change the stabilizers of the state, e.g. a Z error occurred at the beginning of the syndrome extraction qubit s can propagate to 5 qubits but will not flip the flag qubit. It can be verified that for all the other cases the error propagation can be detected by the flag qubit. In general, when the measurement results are non-trivial, some actions may need to be taken. The flag-based quantum error correction also measures the six stabilizers. Figure 2.4 (b) is an example measuring $Z_1Z_3Z_4Z_7$, while the circuits measuring the other stabilizers are similar. The flag-based error correction algorithm for the Steane code used in this thesis is summarized as follows:

Algorithm 1 The flag-based quantum error correction algorithm for the Steane code

Suppose C_1, C_2,..., C_6 are the six stabilizer measurement circuits, $\{M_i\}$ is the measurement results for the syndrome qubit(s) and the flag qubit of each circuit.

Step 1: Run C_1, C_2,..., C_6 sequentially, measure the syndrome measurement qubit and the flag qubit after each cycle, if $\{M_i\}$ is not all $+1$, stop Step 1 and jump to Step 2, otherwise no action needs to be taken. If all the six circuits are all measuring $+1$, no decoding is needed.

Step 2: Run C_1, C_2,..., C_6 sequentially, record all the $\{M_i\}$. Use $\{M_i\}$ in Step 1 and Step 2 for decoding.

For the Steane code, there are several ways to implement the flag error correction scheme [23, 28]. For example, we can measure the syndrome using two ancilla qubits, one acting for syndrome extraction, one acting as the flag qubit, like what we have seen in Figure 2.4 (b). We can use six ancilla qubits in total to sequentially measure all the stabilizer generators. The circuits for measuring the Z and X stabilizer generators sequentially are shown in Figure 2.5 and Figure 2.6 respectively. That is, we need to first use the six ancilla qubits to measure the X (or Z) syndromes, then reuse them to measure the Z (or X) syndromes. In the real case, if we need to reuse any ancilla qubit, one way to doing that is resetting it digitally, not physically. However, in the full density-matrix simulation, for simplicity, we

directly use *ResetGate* to reset the qubits. Since after a qubit is reset to zero states, unlike the case that resetting it digitally, it no longer suffers from the decoherence, the final measurement results might have a lower error rate. A similar problem was discussed in [21], and the result shows that the logical fidelity is slightly improved when apply a conditional gate to flip the nontrivial measurement state, which is negligible. Therefore, we think it is safe to reset the state of the ancilla qubits by using *ResetGate* after the measurement.

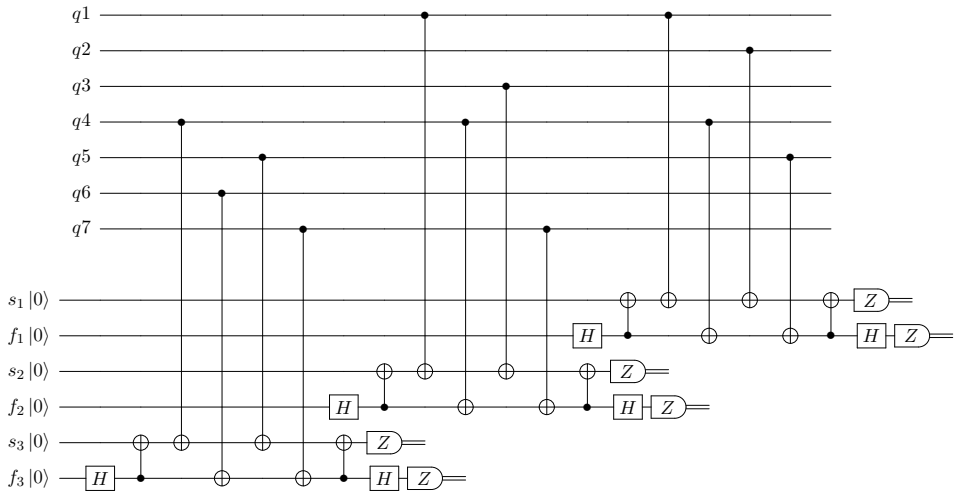


Figure 2.5: The sequential flag syndrome extraction for measuring the Z stabilizer generators. In the figure, s_1,s_2,s_3 measure $Z_1Z_2Z_4Z_5$, $Z_1Z_3Z_4Z_7$, $Z_4Z_5Z_6Z_7$ respectively.

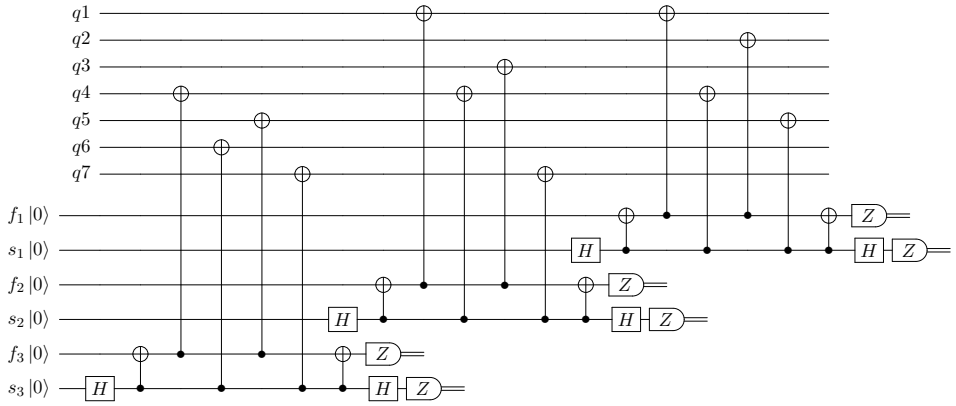


Figure 2.6: The sequential flag syndrome extraction for measuring the X stabilizer generators. In the figure, s_1,s_2,s_3 measure $X_1X_2X_4X_5$, $X_1X_3X_4X_7$, $X_4X_5X_6X_7$ respectively.

An alternative way of the flag-based error correction is using 4 extra qubits: three qubits act for syndrome extraction, one qubit acts as the flag qubit. All the X or Z stabilizer generators can be measured in parallel fault-tolerantly. The flag-based method for measuring Z and X stabilizer generators in parallel are shown in Figure 2.8 and Figure 2.7 respectively.

The method using 6 ancilla qubits and the method using 4 ancilla qubits are both correct for syndrome extraction. However, they are different in the number of ancilla qubits and the duration time. When running the syndrome extraction circuits in simulation or on real hardware, which method is better

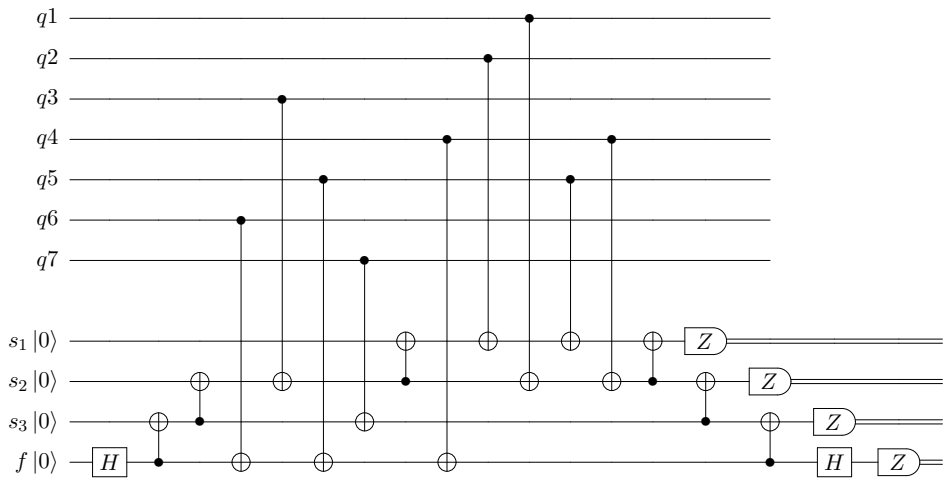


Figure 2.7: The parallel flag syndrome extraction for measuring the Z stabilizer generators. In the figure, s_1,s_2,s_3 measure $Z_1Z_2Z_4Z_5$, $Z_1Z_3Z_4Z_7$, $Z_4Z_5Z_6Z_7$ respectively.

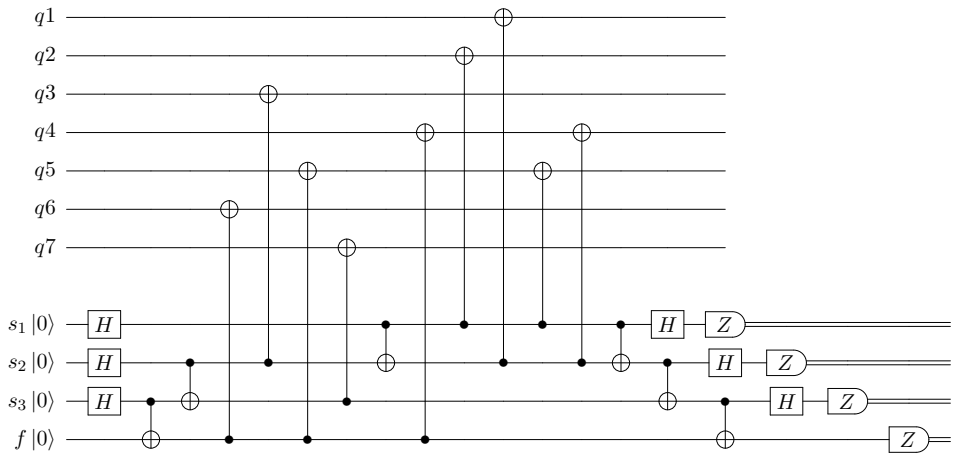


Figure 2.8: The parallel flag syndrome extraction for measuring the X stabilizer generators. In the figure, s_1,s_2,s_3 measure $X_1X_2X_4X_5$, $X_1X_3X_4X_7$, $X_4X_5X_6X_7$ respectively.

depends on many factors. For example, a typical coherent time for superconducting transmon qubits can be $30\mu s$, while the value is much bigger for the ion-trap qubits. Thus, for superconducting systems, parallelism is important, scheduling these circuits to increase parallelism in order to execute all operations in a shorter time is likely to be considered. In this thesis, to decide which method works better, we will evaluate the two methods using a metric called "logical fidelity" suggested in [21], further details will be introduced in Chapter 4.

After the syndrome extraction procedure, we simply use a look-up table for decoding, which is obtained by enumerating all the possible syndromes caused by one Pauli error that happened in any place in the circuit. After the decoding process, for convenience, we directly apply error-free corrections to recover the state.

Chapter 3

Simulation of QEC circuits

3.1 The criterion for testing quantum fault-tolerance

The criterion for testing quantum fault-tolerance used in this thesis is proposed by Daniel Gottesman in 2016 [17]. To introduce the criterion, we consider an original circuit (unencoded) and its FT encoded version. To test the quantum fault-tolerance, we need to run both the original circuit and its encoded version. For a single circuit, after applying all the quantum operations, we measure all the qubits resulting in a measurement output i. Under certain noise condition, if we repeatedly run the circuit for many times, we can obtain an output distribution of a circuit (Let $\{r_i\}$ and $\{q_i\}$ denote the output distribution of an original circuit and its encoded version respectively). We can also directly calculate the ideal output distribution in error-free case if the circuit is not too large, denote as $\{p_i\}$. To show the effectiveness of the FT protocol, we need to calculate the statistical distance (also called the error rate) between $\{q_i\}$ and $\{p_i\}$, as well as $\{r_i\}$ and $\{p_i\}$, which can be done in a simple way as follows:

$$P_u(C) = \frac{1}{2}\sum_i |p_i - q_i| \tag{3.1}$$

$$P_e(C) = \frac{1}{2}\sum_i |p_i - r_i| \tag{3.2}$$

while $P_u(C)$ and $P_e(C)$ refer to the statistical distance of an unencoded circuit and its encoded version respectively.

The main idea of the criterion to test fault-tolerance is comparing $P_u(C)$ and $P_e(C)$ for many different test circuits C. If for all the test circuits in a circuit family $\{C\}$, the encoded versions have lower statistical distances than the unencoded versions, we say the quantum fault-tolerance criterion is successfully demonstrated. In contrast, if there exists any unencoded circuit in $\{C\}$ that has a lower error rate than the encoded version, the demonstration is failed.

To make the demonstration convincing, there are a few points worth mentioning:

1. All the procedures in the encoded circuits are necessary to be fault-tolerant.
2. In the error-free case, the encoded circuits should have the same output distribution as their unencoded versions.
3. All the circuits are necessary to be run in the same system, that is, under the same noise condition.

4. The test circuits family $\{C\}$ needs to be representative and not prohibitively large for practical experiments.

A method for choosing the circuit family was given in [17]. Firstly, all the circuits in the family are constructed from a gate set containing several gates[‡]. The circuit family generated from the gate set should be representative, both long circuit and the short circuit should be considered. In general, there are three parameters need to be specified beforehand, the maximum circuit depth T in the family, the number of different circuit types r, and the maximum periodicity to test p. Note that for some small number t, we can enumerate all the possible circuits. The details of the method can be illustrated as follows:

1. For a circuit containing t gates, randomly choose r types of the circuit using the gate from the gate set with the number of timesteps equals to t, the t is enumerated from 0 to T.

2. Select r types of circuits with q timesteps, and repeat it for $\lfloor T/q \rfloor$ times, the q is enumerated from 1 to p.

Based on the method, we generated 235 circuits for the $[[4, 2, 2]]$ code simulation (by setting the T equals to 12,r equals to 6, and p equals to 5), 84 circuits (by setting T equals to 10, r equals to 3, and p equals to 3) for the 1 logical qubit simulation encoded in the Steane code, and 452 circuits (by setting T equals to 16, r equals to 10, and p equals to 6)[†] for the 2 logical qubits simulation encoded in the Steane code. The circuits lists are appended in the appendix.

For more details about the criterion, we refer to [17].

3.2 The stabilizer formalism simulation

3.2.1 An overview of the stabilizer formalism simulation

For the $[[4, 2, 2]]$ code, we first performed a fault-tolerance testing experiment based on the stabilizer formalism simulation. The stabilizer formalism simulation refers to simulate a kind of quantum circuits that only contains or can be decomposed to Clifford gates, namely Phase gate, Hadamard gate, and CNOT gate. The simulator keeps track of the stabilizers of the quantum states. According to Gottesman-Knill theorem [9, 29], those circuits, also called the stabilizer circuits can be efficiently simulated on classical computers, thousands of qubits can be easily simulated in this way.

Computation based on stabilizer circuits is not universal. However, the stabilizer circuits form an important class of quantum circuits. Quantum error correction can be studied with the stabilizer circuits. By considering an important circuit-level error model, the depolarizing error model, simulating QEC code with stabilizer circuits gives a good benchmark of the code's capability.

In the stabilizer formalism simulation under the depolarizing error model, in order to obtain a well-estimated output distribution, usually, it is necessary to run the circuit many times. We have performed experiments to test how many iterations are enough for giving a good estimation for the expected distribution, it turns out 10^4 times is a reasonable number. Therefore, for all the stabilizer formalism simulations mentioned in this thesis, we repeatedly run the circuits for 10^4 times.

[‡]Note that if a gate e.g. a Hadamard gate is applied on two different qubits, the two cases are counted separately as two gates in the gate set.

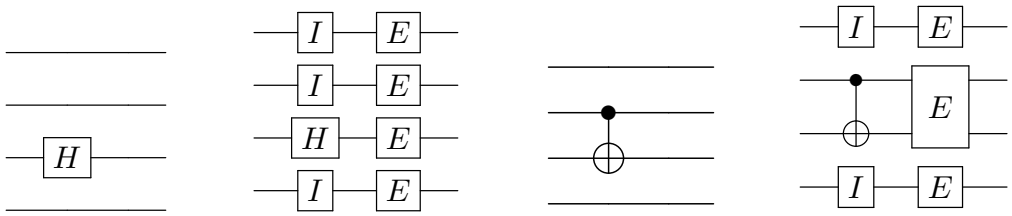
[†]Note that for the circuits used in two logical qubits simulation, we have removed some circuits manually to avoid unnecessary repetitions.

3.2.2 The depolarizing error model

In the stabilizer formalism simulation, an error model called the depolarizing error model was used to insert circuit-level errors. The depolarizing error model is a simple error model that only considers Pauli errors. For the depolarizing error model used in this thesis, given a single error happened with probability p, we follow the following settings:

1. Each single-qubit gate is followed by one-qubit error with probability $p/3$ selected from $\{X, Y, Z\}$.
2. Each CNOT gate is followed by a two-qubit error with the probability $p/15$ selected from $\{I, X, Y, Z\}^{\otimes 2} \backslash \{I \otimes I\}$.
3. The initial state of qubits and the final measurement results suffer from a bit-flipping error with probability p.

In the depolarizing error model, if an error happened after a gate, the spectator qubits (the qubits that do not participate in the quantum gate) are considered as being applied to an identity gate, that is, the identity gates are also followed by a Pauli error. This can be further shown in Figure 3.1.



(a) A Hadamard gate without error.
(b) A Hadamard gate with errors.
(c) A CNOT gate without error.
(d) A CNOT gate with errors.

Figure 3.1: The implementing method of the depolarizing error model used in the stabilizer circuits simulation. The E represents the one-qubit and two-qubit depolarizing errors mentioned above. (a) and (c) are the error-free gates before applying the depolarizing error model. (b) and (d) are examples of how the errors apply to the gates.

In the simulation, different gates might be able to be executed in parallel, which may reduce the errors in the circuits. In our experiment, the parallelism was added to the state preparation procedure, as well as to the single gate in the gate set used for constructing the circuit family. For simplicity, we do not especially add parallelism between different logical operations in the circuits, that is, in each timestep, only one operation is performed.

3.3 The full density-matrix simulation

3.3.1 A brief introduction to quantumsim

Quantumsim [21] is a full density-matrix simulation environment built in Python. It offers a more complicated error model than the depolarizing error model. The quantum operations in quantumsim are Complete-Positive Trace-Preserving (CPTP), all the quantum operations including gates and errors, can be represented in Pauli Transfer Matrix (PTM) form [30]. Under the representation of PTM,

applying quantum operations is converted to multiply matrices. This process can be represented as follows:

$$|\Lambda_2(\Lambda_1(\rho))\rangle\rangle = R_{\Lambda_2} \cdot R_{\Lambda_1} \cdot |\rho\rangle\rangle \tag{3.3}$$

While $R_{\Lambda_2} \cdot R_{\Lambda_1}$ is the PTMs of quantum operation Λ_1 and Λ_2, $|\rho\rangle\rangle$ is obtained by expanding the density operator in Hilbert-Schmidt space.

Since the right side of equation 3.3 is purely matrices multiplication, GPU can be used to accelerate the simulation. Besides, since quantumsim keeps track of the density matrix, the output distribution of a circuit can be extracted directly. This means, if there is no intermediate measurement in the circuit and the final measurement errors are negligible, the final probability distribution can be obtained directly without repeatedly running the circuit.

The quantumsim version used in this thesis is v0.2 [22], and the GPU used for the simulations is NVIDIA's GTX 1080 Ti (11 GB memory).

3.3.2 The noise model

Quantumsim includes several types of error sources. The error sources that are used in this simulation are introduced in the following context. The parameters setting used in this simulation follows the standard-setting in [21]. For further details about the error model, please refer to the supplement material of [21].

The idling qubit error

The qubits at $|1\rangle$ or superposition state tend to suffer from relaxation and dephasing error. The combined effects can be characterized by passing experimentally obtained values T_1,T_2 to quantumsim. The standard relation between T1, T2 is:

$$\frac{1}{T_2} = \frac{1}{T_\phi} + \frac{1}{2T_1} \tag{3.4}$$

Quantumsim uses the idling gate to insert The relaxation and dephasing error (the idling qubit error). This kind of error is the main error source in quantumsim v0.2.

The gate noise

In quantumsim, each gate has a duration. In general, there are three types of gate used in this simulation, the X rotation gate, the Y rotation gate, and the CZ gate. The X rotation and Y rotation gates take 20 ns, while the Z gate used in this thesis is decomposed into X rotation and Y rotation gates, it takes 40 ns. The two-qubit gate (CZ gate) takes 40 ns. All of these gates are modeled by sandwiching an instantaneous gate with two idling gates. For example, the X gate is modeled by sandwiching an π-degree X rotation gate with two idling gates with each takes 10 ns, they sum to the duration of a single-qubit gate. The instantaneous gate in the middle can be modeled by the product of the PTM of the gate without noise and an extra noise term represented in the PTM form. For the single-qubit gates, the extra noise is modeled by a phenomenological depolarizing error extracted from experiments. For the X/Y rotation gates, two parameters are needed to parameterize the extra gate noise, namely the dephasing of X/Y rotations along the axis p_{axis}, and the dephasing of X/Y rotations in the plane p_{plane}.

Likewise, CZ gate can also be represented by an instantaneous CZ gate sandwiched by two idling gates applied on each qubit. The noise of the instantaneous CZ gate comes from an incoherent deviation

from the expected phase value [21], the noise parameter of CZ gate is also needed to set manually.

Note that in the full density-matrix simulation, the CNOT gate is decomposed into a CZ gate sandwiched by two single-qubit Y rotation gates, as shown in Figure 3.2. The Hadamard gate is decomposed into an X gate and a $R_y(-\pi/2)$ rotation gate, as shown in Figure 3.3.

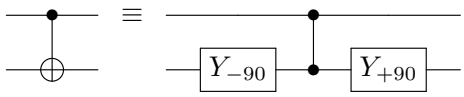


Figure 3.2: The decomposition of a CNOT gate into a CZ gate and two Y rotations.

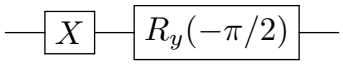


Figure 3.3: The decomposition of a Hadamard gate into an X gate and a $R_y(-\pi/2)$ rotation gate.

The Measurement error

Quantumsim offers a measurement model to approximate the behavior of real measurements. The measurement model is built by sandwiching a readout process with declaration errors ϵ_{R}, with two amplitude-phase damping process. In our simulation, we have tested that under the standard parameters setting, the measurement error does not significantly contribute to the final measurement result of the circuits. Therefore, for simplicity, for circuits that do not have intermediate measurements, we extract the probability distribution from the density matrix directly, which considerably decreases the computational cost.

Summary for the noise parameters setting

The default noise parameters for the density-matrix simulation used in this thesis are shown in Table 3.1, we use the parameters unless otherwise mentioned.

Table 3.1: The standard parameters setting in the full density-matrix simulation.

Parameters	Value
T_1	30000 ns
T_2	30000 ns
Single-qubit gate duration	20 ns
CZ gate duration	40 ns
p_{axis}	10^{-4}
p_{plane}	5×10^{-4}
CZ dephasing	$\frac{0.01}{2\pi}$
Declaration error rate	0.0015

3.3.3 Scheduling

For the full density-matrix simulation, each gate is executed at a specific moment. For many cases, the gates can be executed in parallel to avoid unnecessary qubit idling error. By adding the scheduling

process, we can increase the parallelism of the circuits. In our simulation, we schedule all the gates in an as-late-as-possible (ALAP) way, that is, delaying a quantum gate until it must be executed to continue the computation. OpenQL [31] was used to perform the scheduling process.

Chapter 4

Simulation results

4.1 Simulation results using the stabilizer formalism simulator

4.1.1 The results of simulating the [[4,2,2]] code

To test the quantum fault-tolerance criterion, we encode the [[4,2,2]] code with three different initial states ($|00\rangle_L$, $|0+\rangle_L$, and $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$), and select a circuit family containing 235 different circuits constructed from the gate set $\{X1, X2, Z1, Z2, H_1H_2SWAP, CZ\}$. The depolarizing error rate p is set to 0.01 unless otherwise stated. For both the encoded and unencoded circuits, the output distributions were obtained by running each circuit 10^4 times.

The statistical distance results of the circuits with initial state $|00\rangle_L$ are shown in Figure 4.1. In the figure, the x-axis is the index of the test circuits, the y-axis is the statistical distance (also called the error rate). The orange dots represent the results of the Non-FT circuits, the blue 'x' represent the results of the FT circuits and the green dots are the corresponding post-selection ratios.

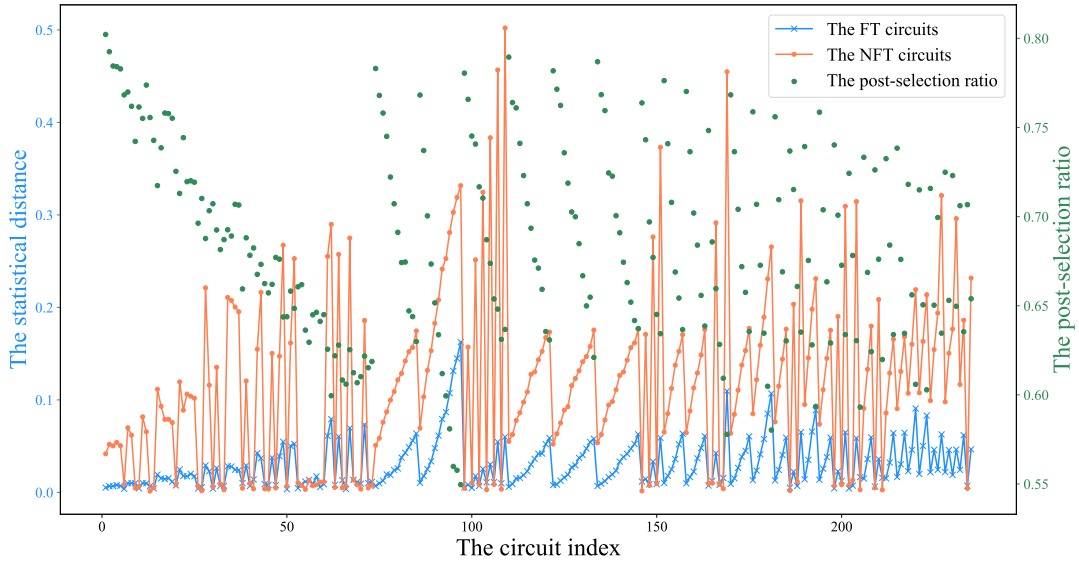


Figure 4.1: The results of testing the FT criterion for the [[4,2,2]] code with the initial state $|00\rangle_L$ based on the stabilizer formalism simulation.

Figure 4.1 shows that for most of the circuits, the statistical distance of the FT encoded circuits is lower than the unencoded ones. However there are some other circuits for which the encoded and unencoded versions have very close statistical distances. We examined those circuits, some of them are shown in Table 4.1.

Table 4.1: Some of the circuits with the initial state $|00\rangle_L$ for which the [[4,2,2]] code encoded circuits and the unencoded ones have very close statistical distances in the stabilizer formalism simulation.

Circuit index	**The content**
9	H_1H_2SWAP X2
12	X1 H_1H_2SWAP
13	Z1 H_1H_2SWAP X1
19	Z2 X1 H_1H_2SWAP X1
25	CZ Z1 X1 CZ H_1H_2SWAP
26	X2 H_1H_2SWAP Z2 Z2 Z2
29	X1 CZ Z2 H_1H_2SWAP Z2

Table 4.1 shows that all those circuits have the H_1H_2SWAP gate. The H_1H_2SWAP gate is able to bring a collapsed state to a superposition state. In this simulation, if a quantum state (whether it is encoded or not) is brought to the superposition (e.g. $|00\rangle + |11\rangle$), it is no longer affected by any logical errors[‡]. For the unencoded circuits, the errors caused by the depolarizing error model are all logical errors. For the encoded circuits, the errors can be either logical errors or non-logical errors. If the error is not a logical error, the measurement result will be rejected by the post-selection process; that is, for the encoded circuits, we also only have logical errors at the output. Since both the logical encoded and unencoded circuits are robust to the errors caused by the depolarizing error model, they both have low statistical distances. These specific circuits are not relevant to the effectiveness of FT implementation of QEC codes, Therefore the circuits can be discarded when we consider whether the FT criterion is satisfied.

In the case the circuits are initialized to the $|0+\rangle_L$ or the $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$ state, the encoded circuits always have lower statistical distance than the unencoded versions, (see Figure 4.2 and 4.3). Therefore, the FT criterion is satisfied.

We can also observe from Figure 4.2 and 4.3 that the post-selection ratios decrease with the increase of statistical distances. This means if there are more errors inside circuits, more measurement outputs will be rejected, showing the effectiveness of the post-selection process.

Test under different error rates

We also analyzed the fault-tolerance of the [[4,2,2]] code under different error rates of the depolarizing error model. From the above results, we conclude there is no specific type of circuits or gate combinations that may result in a low or high statistical distance where using the stabilizer formalism simulator. The factor that affects the statistical distance most is the circuit depth, since the deeper the circuit depth is, the higher the probability that errors might happen. Therefore, here we only test a few circuits. The error rate is swept from 0.001 to 0.1 with a step of 0.001. The results of two selected circuits, one consisting of ten X1 gates, the other one consisting of ten CZ gates are shown in Figure 4.4.

[‡]This is because the logical X does not change the state, and the logical Z only changes the sign before $|1\rangle$, which does not contribute to the measurement result since we measure all qubits in the standard basis.

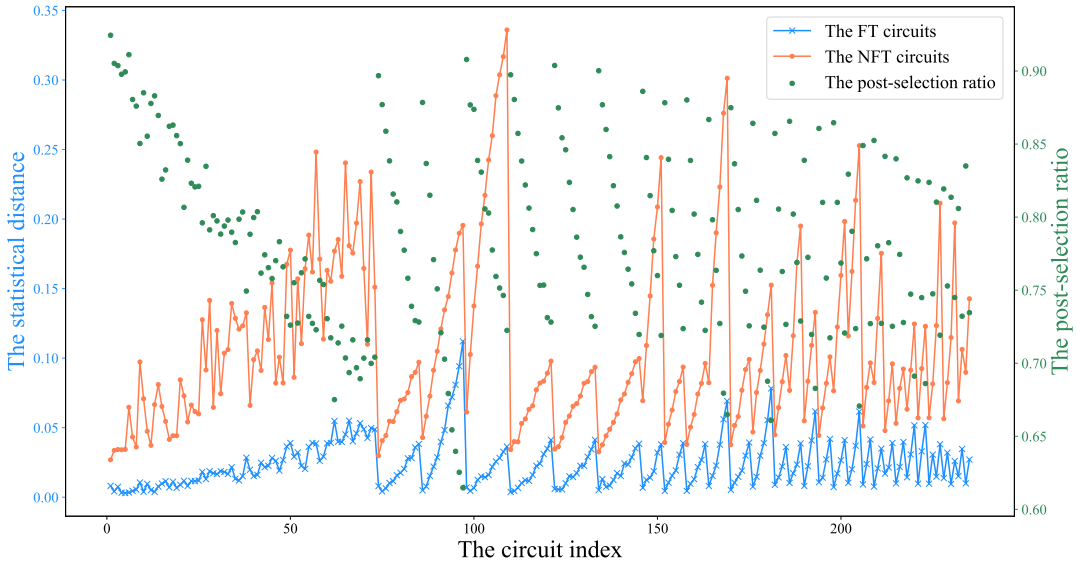


Figure 4.2: The results of testing the FT criterion for the [[4,2,2]] code with the initial state $|0+\rangle_L$ based on the stabilizer formalism simulation.

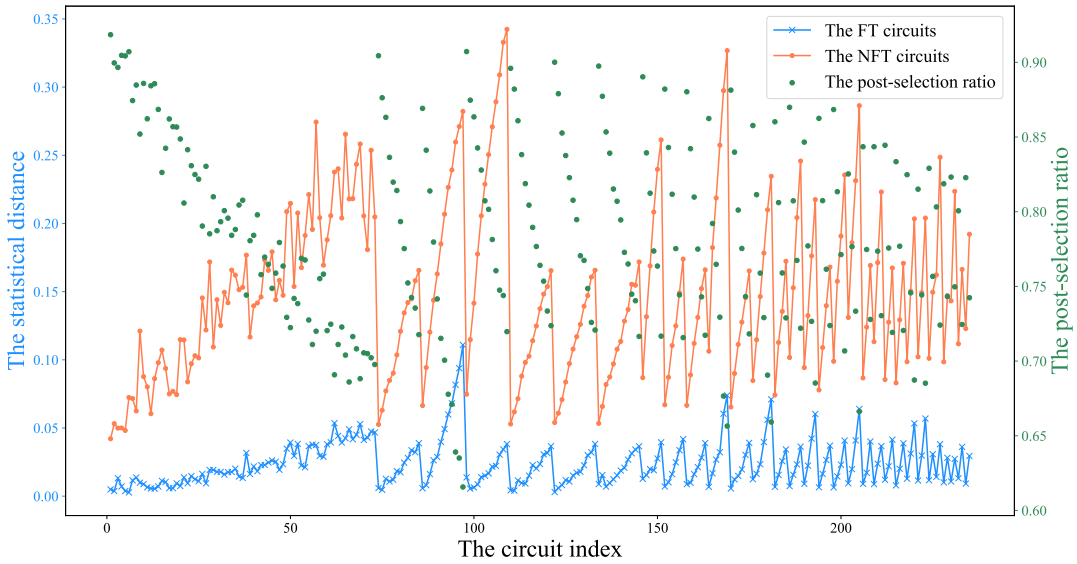
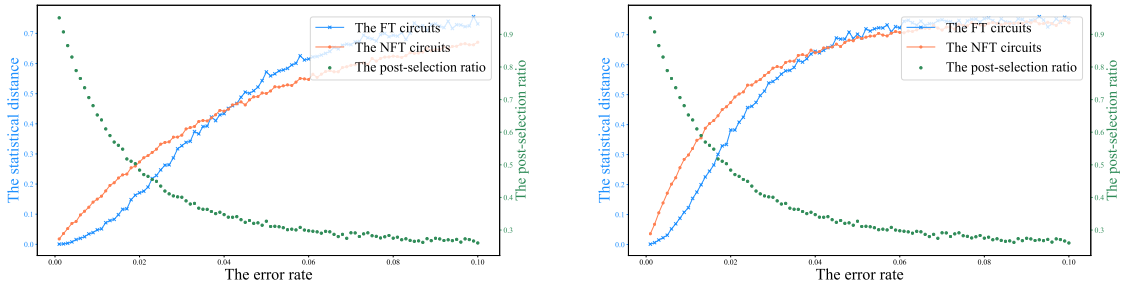


Figure 4.3: The results of testing the FT criterion for the [[4,2,2]] code with the initial state $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$ based on the stabilizer formalism simulation.

We observe that for both circuits, when the error rate increases, the statistical distance of the encoded circuits will soon be larger than the unencoded ones. The result suggests that whether the FT criterion is satisfied also depends on the noise condition of the system.

In general, simulating the stabilizer circuits under the depolarizing error model can be used to evaluate whether the FT criterion is likely to be satisfied in real hardware. In summary, for the circuit family containing 235 circuits used in this simulation, all encoded circuits have lower statistical distances than the unencoded circuits (excluding the cases that both the encoded and unencoded versions are robust to the depolarizing errors). Therefore, we regard the FT criterion is satisfied in this experiment.



(a) The results of simulating a circuit consisting of ten X gates.

(b) The results of simulating a circuit consisting of ten CZ gates.

Figure 4.4: The results of running a circuit consisting of ten X or ten CZ gates based on the stabilizer formalism simulation. The error rate of the depolarizing error model is swept from 0.001 to 0.1 with a step of 0.001.

4.2 Simulation results using the full density-matrix simulator

We performed full density-matrix simulations based on a Python package called quantumsim for the [[4,2,2]] code and the [[7,1,3]] Steane code. The noise parameters used in quantumsim are based on the standard parameter values given in [21].

4.2.1 The results of simulating the [[4,2,2]] code

For the $[[4, 2, 2]]$ code, we used the same circuit family and the three initial states as in the stabilizer formalism simulation. In the density-matrix simulation, since the CZ gate is available, we directly use it instead of decomposing it into a CNOT gate and two Hadamard gates as we did in the stabilizer formalism simulation. The statistical distance of the circuits initialized to state $|00\rangle_L$ is shown in Figure 4.5.

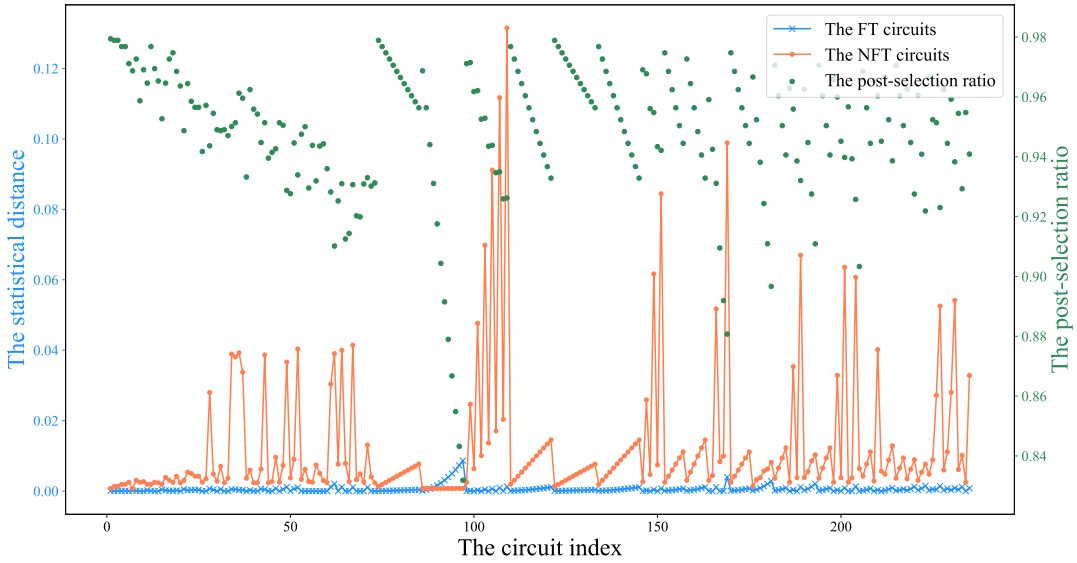


Figure 4.5: The results of testing the FT criterion for the [[4,2,2]] code with the initial state $|00\rangle_L$ in the density-matrix simulation.

They show that there are a series of circuits for which the FT criterion is not satisfied. For example, the unencoded circuits with index from No.85 to No.96 in Figure 4.5 have lower statistical distance than the encoded ones, some of those circuits are shown in Table 4.2.

Table 4.2: Some of the test circuits with initial state $|00\rangle_L$ for which the FT criterion is not satisfied in the density-matrix simulation.

Circuit index	The content
85	CZ
86	CZ CZ
87	CZ CZ CZ
88	CZ CZ CZ CZ
89	CZ CZ CZ CZ CZ

The table shows that those circuits only contain a series of CZ gate. Thus the reason that the FT criterion is not satisfied is rather direct: for the unencoded states the CZ gate is directly applied to the two physical qubits. The initial state $|00\rangle$ will not be affected by CZ gate (suppose CZ gate is error-free here). Therefore, in our simulation, the statistical distance of the unencoded states will always remain close to zero no matter how many CZ gates are applied. In contrast, the encoded state $|00\rangle_L$ is a superposition state as shown in equation 2.17 and it constantly suffers from the idling error. For those circuits with an index from No.85 to No.96, the statistical distance of the encoded circuits tend to increase with the increase of the circuit depth.

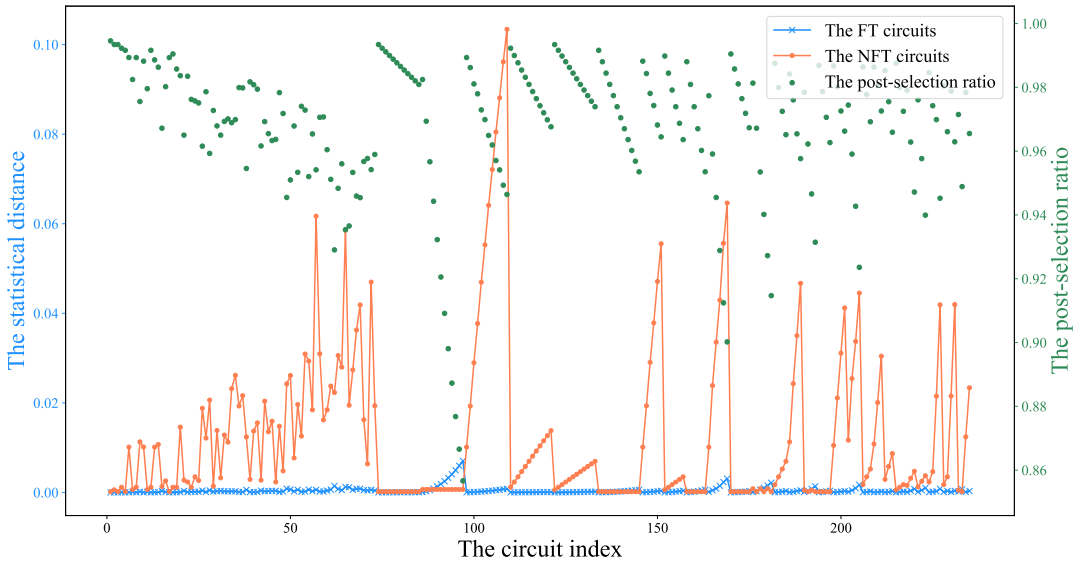


Figure 4.6: The results of testing the FT criterion for the [[4,2,2]] code with the initial state $|0+\rangle_L$ based on the density-matrix simulation.

The results of the circuits with initial state $|0+\rangle_L$ are shown in Figure 4.6. It shows that there are more circuits do not satisfy the FT criterion. For the circuits with small statistical distance for both encoded and unencoded versions, we found that except for the CZ gate discussed above, the circuits also include two other types of gates, the X2 gate (applying X gate on the second qubit ($|+\rangle$ state)) and the Z2 gate (applying Z gate on the second qubit ($|+\rangle$ state)). Some of the circuits are shown in Table 4.3.

To explain why those unencoded circuits also have a low statistical distance, we take the No.80 circuit

Table 4.3: Some of the circuits with the initial state $|0+\rangle_L$ that have close statistical distances for the [[4,2,2]] encoded and unencoded circuits based on the density-matrix simulation.

Circuit index	The content
80	X2 X2 X2 X2 X2 X2 X2 X2
85	CZ
86	CZ CZ
169	Z2 X2
170	Z2 X2 Z2 X2
193	Z2 X2 Z2

in Table 4.3 as an example. For the unencoded state, applying multiple X gates on a $|+\rangle$ qubit state can relieve the relaxation process of the qubit. This is because in the $|+\rangle$ state, part of the decay that the qubit is suffering in the upper half of Bloch sphere, is compensated by the decay in the lower half of Bloch sphere when the qubit is flipped by an X gate. Therefore, applying multiple X gates to a qubit at $|+\rangle$ state would reduce the statistical distance, leading to the results of the unencoded circuits with index from No.74 to No.85 shown in Figure 4.6. Since in our simulation, the Z gate is decomposed into X and Y gates, the circuits consisting of multiple Z2 gates (e.g. from No.157 to No.162) can also be explained in this way.

In the case of the circuits with initial state $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$, Figure 4.7 shows that for most circuits, the FT criterion is satisfied. While only a small number of circuits with index from No.93 to No.97 do not satisfy the criterion due to the same effect of the unencoded CZ gate as discussed above.

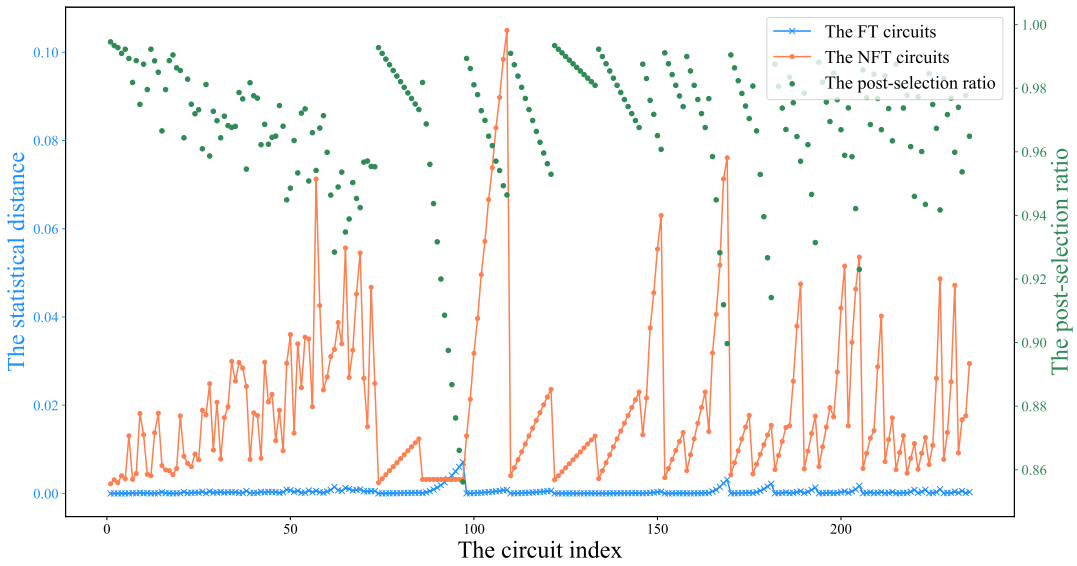


Figure 4.7: The results of testing the FT criterion for the [[4,2,2]] code with initial state $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$ based on the density-matrix simulation.

In summary, in the density-matrix simulation, the system encoded in the [[4,2,2]] code does not fully satisfy the FT criterion. However, the fault-tolerant encoded circuits indeed have a lower error rate than the unencoded ones for most the cases.

Test under different T_1,T_2

We also tested the [[4,2,2]] code under different T_1 and T_2. With the initial state $|00\rangle_L$, two different simulations were performed. T_1=T_2 were set to 3000 ns and 300000 ns.

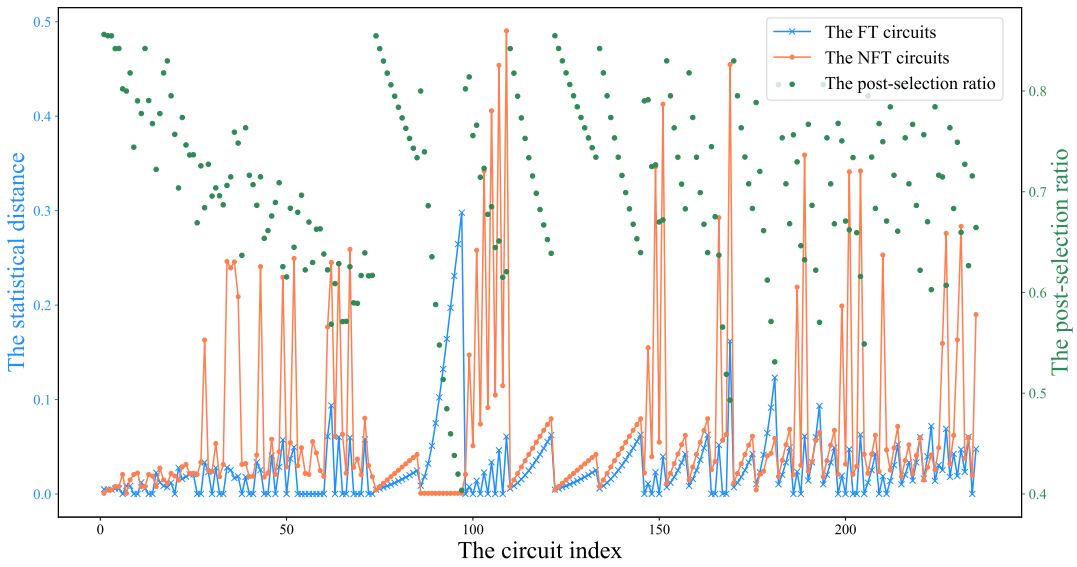


Figure 4.8: The density-matrix simulation for the [[4,2,2]] code with T_1=T_2=3000 ns. The other parameters follow the standard parameters setting.

Figure 4.8 shows the result of setting T_1,T_2 to 3000 ns. In general, both the statistical distance of the encoded and unencoded circuits increases. For some circuits such are the circuits with index No.1, No.7, and No.11, the increase of the statistical distance of the encoded circuit is higher than the unencoded ones comparing to Figure 4.5 where the T_1,T_2 are set to 30000 ns. This means under small T_1, T_2, the FT criterion is not likely to be satisfied. Figure 4.9 shows the results of setting T_1,T_2 to 300000 ns. Comparing to Figure 4.5, the shapes of the curves do not significantly change. The statistical distances decrease for both encoded and unencoded circuits, and the post-selection ratios of the [[4,2,2]] code increase. This suggests that in an idling error dominated system, the encoded circuits might be better than the unencoded ones even though the physical qubit also has quite low error rate.

4.2.2 The results of simulating the [[7,1,3]] code

Test for the parallel QEC cycle and the sequential QEC cycle

As we mentioned in Chapter 2, there are two ways to implement the flag-based error correction: the sequential flag-based method, and the parallel flag-based method. We simulated the syndrome extraction process for the two methods. Before running the simulation, the syndrome extraction process is scheduled manually for both methods to reduce idling qubit errors. Note that the scheduling process needs to be done carefully, since the circuits might not be fault-tolerant if some gates are switched during the scheduling process, even in the noise-free case they are equivalent.

We use logical fidelity to evaluate the two methods. The logical fidelity refers to the probability that without applying any logical operation, the final measurement result is the same as the initial state. Since the syndrome measurement process includes intermediate measurements, it is necessary to

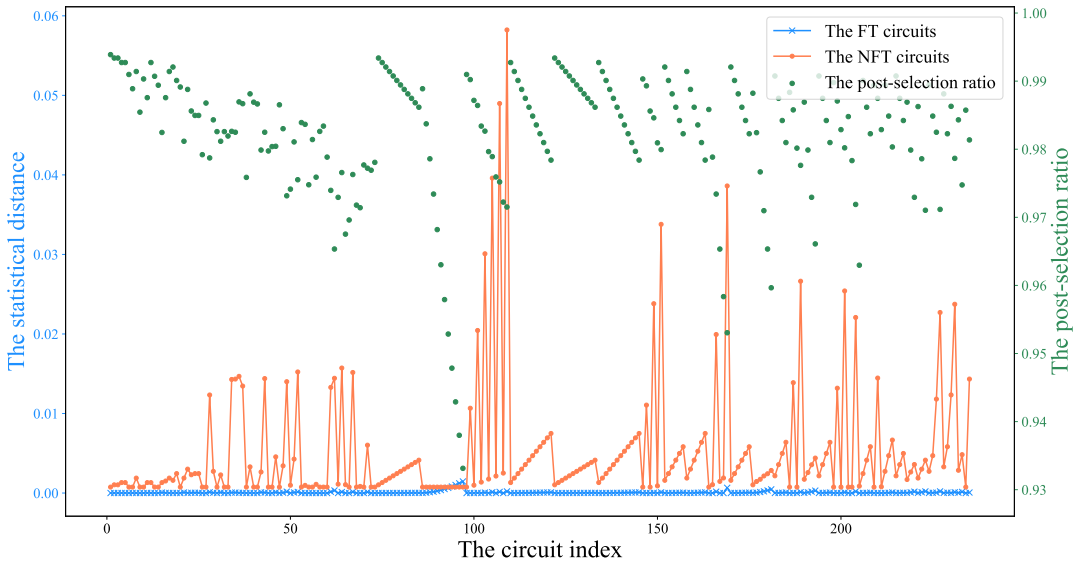


Figure 4.9: The density-matrix simulation for the [[4,2,2]] code with T_1=T_2=300000 ns. The other parameters follows the standard parameter values.

run the circuits many times to obtain an average logical fidelity. Table 4.4 shows the average logical fidelity obtained by averaging the results over 10^4 repetitions for both methods. The results suggest the parallel method has a slightly higher logical fidelity. Therefore, in the following simulation, the parallel flag-based method is selected as the syndrome extraction method.

Table 4.4: The average logical fidelity of the sequential and the parallel flag-based method.

The flag-based method	**The average logical fidelity**
The sequential method	0.8300
The parallel method	0.8665

Simulating one logical qubit with logical operations

Since inserting error correction procedures at different places (e.g. apply the QEC cycle after each logical operation, or only apply it at the end of the circuit) of a circuit can lead to different results, we performed simulations to test four different ways to insert the QEC cycle. The four ways are:

1. Many QEC cycles are inserted: the QEC cycle is applied after each procedure.
2. Two QEC cycles are inserted: one is applied after the Non-FT encoding procedure and the other one is applied at the end of the circuit.
3. One QEC cycle is inserted after the Non-FT encoding procedure.
4. No QEC cycle is inserted: the Non-FT encoding is followed by a series of FT logical operations and the final measurement, which is followed by a post-selection process.

In this simulation, we tested some specific circuits, such as the circuits consisting of ten X gate, or ten Z gates. The statistical distance is used as the metric again. The results of the circuit consisting of ten

Table 4.5: The results of testing the four ways to insert QEC cycles in the density-matrix simulation. A circuit containing ten Z gates is selected as the test circuit.

The QEC type	The statistical distance
With many QEC cycles after each operation	0.3968
With two QEC cycles in total	0.1538
With one QEC cycle in total	0.1155
No QEC cycle, only with encoding & post-selection process	0.0001

Z gates are shown in Table 4.5. Similar results are obtained for the ten X gates circuit, which are not listed here.

They show that the case without inserting any QEC cycle has the lowest statistical distance followed by the case inserting one QEC cycle. The worst case is when inserting a QEC cycle after each procedure has the highest statistical distance. This suggests that, increasing the number of QEC cycles will make the output results worse. Therefore, for the following experiment simulating one logical qubit, we will not insert QEC cycle; that is, we only encode the physical qubits in the Steane code and apply logical operations; after that we measure all qubits, followed by a post-selection process. However, if no QEC cycle is applied to the encoding process, the circuit is not fault-tolerant. Therefore, our simulation here might not be considered as an experiment to test the FT criterion, but an experiment to test the effectiveness of encoding, as well as FT logical operations.

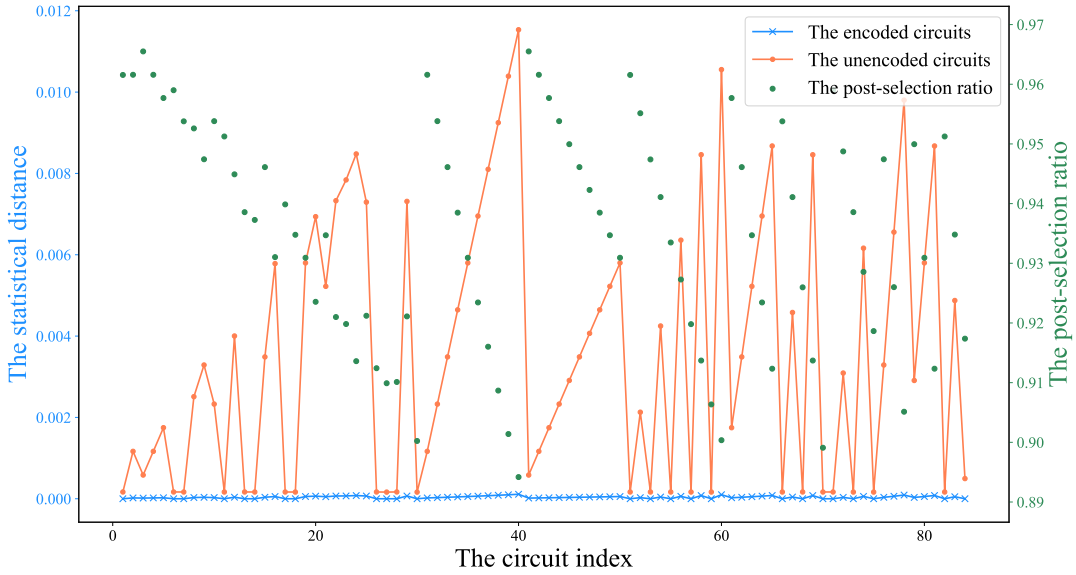


Figure 4.10: The results of simulating the 84 circuits encoded in the Steane code in the density-matrix simulation with no QEC cycle applied.

The results of simulating the 84 circuits generated from the gate set $\{X, Z, H\}$ encoded in the Steane code with no QEC cycle are is shown in Figure 4.10. For all circuits, the encoded ones have lower statistical distances than the unencoded ones. Though, there are some circuits that the statistical distance of the encoded and unencoded versions are quite close. Some of those circuits are listed in Table 4.6.

We can observe that all circuits listed in Table 4.6 are in superposition states at the output of the

Table 4.6: Some of the circuits that have close statistical distances for the encoded and unencoded circuits in simulating one logical qubit based on the density-matrix simulation.

Circuit index	The content
1	H
6	X H
7	H X X
11	X H X X
13	H H X Z H
14	Z Z H Z X
26	Z H Z H Z H X Z X

circuits. The reason the unencoded circuits also have low statistical distances can be regarded as a combined effect of single-qubit decay which can be described in Bloch sphere. The general idea can be illustrated as follows: before a state is brought to the superposition, it remains in the y-z plane of Bloch sphere, while a $R_y(\pi/2)$ rotation brought by Hadamard gate (remember that Hadamard gate is decomposed into an X gate and a $R_y(\pi/2)$ rotation) brings any state in the y-z plane to the equatorial plane of Bloch sphere. If we measure the state right after applying the Hadamard gate, we will have equal probability to get +1 or -1 (in the case we only consider the idling error); that is, before applying a Hadamard gate to the qubit, the qubit decay (e.g. from $|1\rangle$ decays to $|0]\rangle$) does not contribute to the final statistical distance. Therefore, for those unencoded circuits which states are in superposition in the end, the statistical distance can be reasonably low.

In general, our results show the effectiveness of the encoding process as well as the FT logical operations.

Simulating two logical qubits with logical operations

Initially, we planned to simulate two logical qubits in quantumsim, with each implemented with the parallel flag-based quantum error correction. However, since each encoded block of qubits requires to simulate $(7 + 4)$ qubits, the two blocks require 22 qubits in total. This means the size of the density matrix in quantumsim is $2^{22} \times 2^{22}$, which is far intractable for our simulation hardware (GTX 1080Ti). It turns out, the GPU we used can maximally simulate 14 qubits. Therefore, for the two logical qubits simulation, we can only simulate the case without adding quantum error correction, which is the same as we did in the one logical qubit simulation. This simulation also does not satisfy the FT criterion, since without the QEC procedure, the encoding process is not fault-tolerant.

We simulated two logical qubits encoded in the Steane code. 452 circuits were generated from the gate set $\{X, Y, Z, H, CNOT\}$. The results are shown in Figure 4.11 and 4.12.

For all circuits, the encoded ones have a lower distance than the unencoded versions. This again shows the effectiveness of the encoding process as well as the FT logical operations.

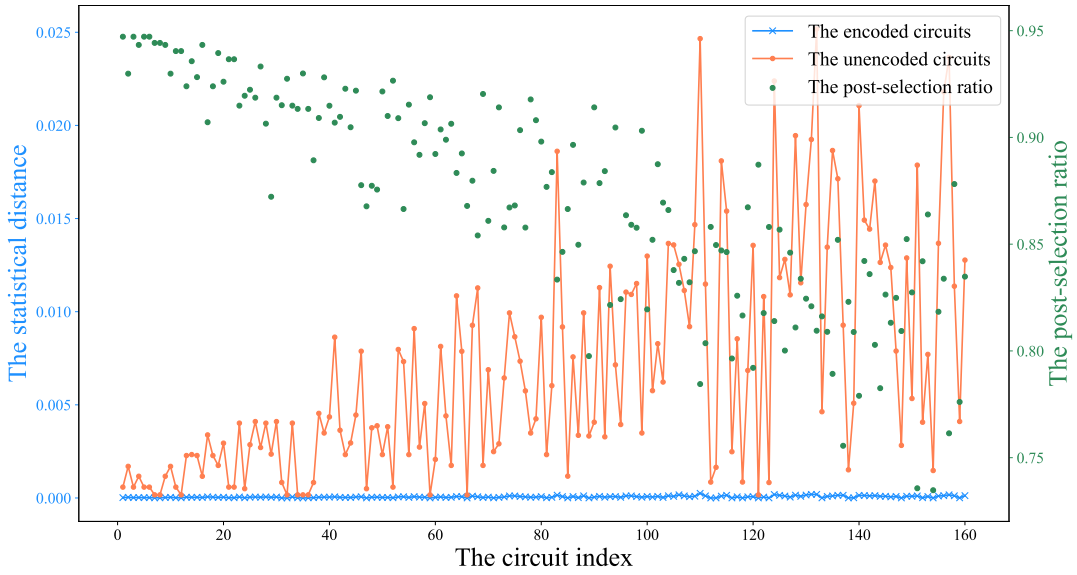


Figure 4.11: The results of simulating two logical qubits encoded in the Steane code without QEC cyle. There are 452 circuits tested in total, here are the results of the first 160 circuits.

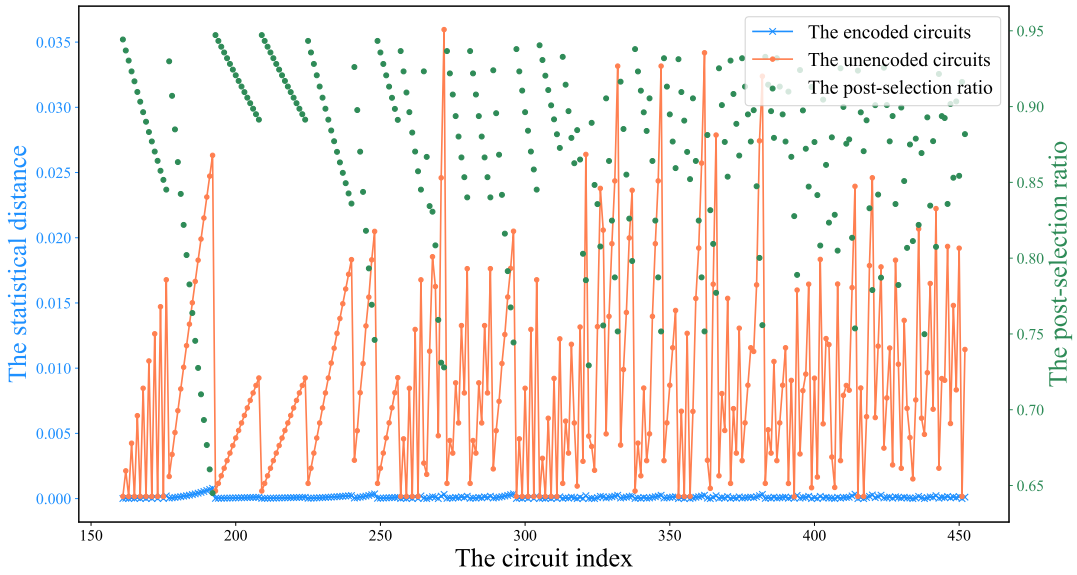


Figure 4.12: The results of simulating two logical qubits encoded in the Steane code without QEC cyle. There are 452 circuits tested in total, here are the results of the last 292 circuits.

Chapter 5

Conclusion

5.1 Conclusion

Quantum fault-tolerance is necessary for quantum error correction and building large-scale quantum machines capable of outperforming classical computers. In this thesis, we have explored the fault-tolerance of two small quantum correction codes: the [[4,2,2]] code, and the [[7,1,3]] Steane code, that are good candidates to be applied to NISQ processors. We have tested if they satisfy the fault-tolerance criterion proposed by Daniel Gottesman using 2 simulators, the stabilizer formalism simulator that includes simple error models and a full density matrix simulator called quantumsim, which includes more realistic noise. The criterion for checking fault-tolerance is a method to test the effectiveness of fault-tolerant encoding and operations. It requires for all the test circuits that the error rate (the statistical distance) of encoded circuits is lower than their unencoded versions.

We first simulated the [[4,2,2]] code on the two simulators. A circuit family consisting of 235 circuits was used to perform the FT testing experiment. Under the condition that the error rate of the depolarizing error model is 0.01, the results show for all the circuits (with 3 different initial states $|00\rangle$,$|0+\rangle$,$\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$) that the error rate of the encoded circuits is lower than their unencoded versions, which suggests the FT criterion is satisfied. If the error rate is swept from 0.001 to 0.1, the encoded circuits may have a higher error rate than the unencoded ones in the end, which suggests the effectiveness of FT encoding depends on the noise condition of the environment.

In the full density-matrix simulation, for the [[4,2,2]] code we performed the same fault-tolerance testing experiment on the 235 circuits. The results show that, under the standard parameters setting extracted from a near-term device, the FT criterion is not fully satisfied. Specifically, for the initial states $|00\rangle$, $|0+\rangle$, and $\frac{|00\rangle_L+|11\rangle_L}{\sqrt{2}}$, 9 circuits out of the 235 circuits, 32 circuits out of the 235 circuits, and 5 circuits out of the 235 circuits do not satisfy the criterion, respectively. In general, two types of circuits make the FT testing criterion not satisfied. The first type is the circuits that have CZ gates. The second type the circuits initialized to the $|0+\rangle$ state and have the X2 (applying X gate on the second qubit ($|+\rangle$ state)) and the Z2 (applying Z gate on the second qubit ($|+\rangle$ state)) gates. However, our simulation results show that encoded logical qubits indeed lead to lower error rates for most of the circuits.

For the Steane code, we only performed the full density-matrix simulation as it is more realistic. We simulated the parallel flag-based implementation of the quantum error correction cycle. For the simulation of one logical qubit with logical operations, we first compared three different cases to insert

QEC cycles and one case is that no QEC cycle is inserted. The results show that under the standard parameters setting, if no QEC cycle is used, the error rate is around 0.0001, while adding one QEC cycle has an error rate of 0.1136, which is 1000x larger. It seems that adding more QEC cycles increases the error rate. The best result was obtained where no QEC cycle was inserted, that is, only with the encoding process and logical operations. Therefore, we decided to test the circuit family consisting of 84 circuits without inserting any QEC cycle. Note that in this case, the process of encoding the physical qubits into logical qubits is not fault-tolerant. Thus, this simulation only tested the effectiveness of the encoding process, not the FT criterion, which requires all the procedures to be fault-tolerant. The results show that the encoding process improves the statistical distance of all 84 circuits. All encoded circuits have a lower error rate than the unencoded ones.

Regarding the simulation of two logical qubits with logical operations using the Steane code, it was not possible to run such an experiment. This is due to the extremely high memory cost of simulating two logical qubits with QEC cycles. It requires to simulate 22 physical qubits (the required size of the density matrix is $2^{22} \times 2^{22}$). We were only able to simulate the encoding (without the QEC cycle) and logical operations; that is, we encoded two blocks of qubits containing 14 qubits in total and performed logical operations. The results show again that the encoding process is effective for all 452 circuits; the encoded versions have lower error rates than the unencoded ones.

Our final remark is whether satisfying the FT criterion is a tradeoff among many factors, e.g. the noise of the system, the time the QEC cycle takes, the QEC code's capability, the decomposition method of gates, the scheduling approach, etc. In general, the FT criterion is more likely to be satisfied if a better system (low noise, a short period of QEC cycle, efficient decomposition of gates, etc.) is used.

5.2 Future work

In our simulation, the Steane code with flag-based error correction simulating one logical qubit requires only eleven qubits. Therefore, it might be possible to map the circuits to the IBM Quantum Experience devices and compare the experimental results with the simulation.

In general, to make a more accurate demonstration of quantum fault-tolerance, many aspects have to be considered. For example, a broad family of circuits, generated from a universal gate set is required. For the error correction procedure, it is also interesting to use more qubits in the syndrome extraction procedure instead of reusing the ancilla qubits. However, this is not feasible here due to the high computational costs. Besides, in simulation, we often omit a lot of hardware constraints, such as connectivity, the classical decoding time, different gate fidelity, etc. As we mentioned before, whether satisfying the FT criterion depends on many factors. Therefore, to really demonstrate quantum fault-tolerance, a comprehensive experiment on hardware is needed.

Indeed, simulation results can offer insights to hardware experiments. In the full density-matrix simulation of this thesis, the version of quantumsim we used is v0.2. Though it includes a few kinds of error sources, one significant error source such as the leakage error in the superconducting qubit system is not taken into account. Quantumsim will release its 1.0 version in the future, in which a new CPhase gate called the Netzero CPhase that includes the leakage error is implemented. It would be interesting to see how the FT protocol performs in the presence of leakage.

Apart from the density matrix simulation, another type of simulation is the Hamiltonian simulation. In [20], a Hamiltonian simulation coupling the data qubits to some extra qubits to study the effect of

decoherence on the FT criterion is performed. It would be interesting to compare the decoherence due to the coupling to the environment in the Hamiltonian simulation and the decoherence due to the T_1 and T_2 decay in the density matrix simulation.

Bibliography

[1] Wikipedia contributors, "Quantum computing — Wikipedia, the free encyclopedia," 2019.

[2] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information* **2** (2016) 15023.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* **21** no. 2, (1978) 120–126.

[4] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters* **103** no. 15, (2009) 150502.

[5] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv:1203.5813* (2012) .

[6] K. A. Frank Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *nature* **574** (2019) 505.

[7] S. Aaronson and L. Chen, "Complexity-theoretic foundations of quantum supremacy experiments," in *32nd Computational Complexity Conference (CCC 2017)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2017.

[8] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum* **2** (2018) 79.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th ed., 2011.

[10] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical review A* **52** no. 4, (1995) R2493.

[11] A. M. Steane, "Error correcting codes in quantum theory," *Physical review letters* **77** no. 5, (1996) 793.

[12] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature* **299** no. 5886, (1982) 802.

[13] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, vol. 68, pp. 13–58. 2010.

[14] P. W. Shor, "Fault-tolerant quantum computation," in *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 56–65, IEEE. 1996.

[15] A. M. Steane, "Active stabilization, quantum computation, and quantum state synthesis," *Physical review letters* **78** no. 11, (1997) 2252.

[16] E. Knill, "Scalable quantum computing in the presence of large detected-error rates," *Physical review A* **71** no. 4, (2005) 042322.

[17] D. Gottesman, "Quantum fault tolerance in small experiments," *arXiv:1610.03507* (2016) .

[18] R. Harper and S. T. Flammia, "Fault-tolerant logical gates in the ibm quantum experience," *Physical review letters* **122** no. 8, (2019) 080504.

[19] C. Vuillot, "Is error detection helpful on ibm 5q chips?," *arXiv:1705.08957* (2017) .

[20] D. Willsch, M. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Testing quantum fault tolerance on small systems," *Physical review A* **98** no. 5, (2018) 052348.

[21] T. O'Brien, B. Tarasinski, and L. DiCarlo, "Density-matrix simulation of small surface codes under current and projected experimental noise," *npj Quantum Information* **3** no. 1, (2017) 39.

[22] "Quantumsim v0.2.". https://github.com/quantumsim/quantumsim/tree/stable/v0.2.

[23] R. Chao and B. W. Reichardt, "Quantum error correction with only two extra qubits," *Physical review letters* **121** no. 5, (2018) 050502.

[24] A. M. Steane, "A tutorial on quantum error correction," in *PROCEEDINGS-INTERNATIONAL SCHOOL OF PHYSICS ENRICO FERMI*, vol. 162, p. 1, IOS Press; Ohmsha; 1999. 2007.

[25] D. Gottesman, "The heisenberg representation of quantum computers," *arXiv quant-ph/9807006* (1998) .

[26] H. Goto, "Minimizing resource overheads for fault-tolerant preparation of encoded states of the steane code," *Scientific reports* **6** (2016) 19578.

[27] T. J. Yoder and I. H. Kim, "The surface code with a twist," *Quantum* **1** (2017) 2.

[28] L. Lao and C. G. Almudever, "Fault-tolerant quantum error correction on near-term quantum processors using flag and bridge qubits," 2019.

[29] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A* **70** no. 5, (2004) 052328.

[30] D. Greenbaum, "Introduction to quantum gate set tomography," 2015.

[31] "Openql.". https://github.com/QE-Lab/OpenQL.

Appendix A

The test circuit families

Table 1. The test circuit family for the [[4,2,2]] code simulation.

No.	The content	No.	The content
1	empty	32	HHSWAP X1 X1 Z2 X1 CZ
2	X1	33	HHSWAP Z1 CZ Z2 X2 X2
3	X2	34	CZ X2 HHSWAP Z2 Z1 HHSWAP
4	Z1	35	CZ Z1 X2 X1 HHSWAP HHSWAP
5	Z2	36	X2 X2 Z2 X2 HHSWAP HHSWAP
6	HHSWAP	37	HHSWAP Z1 HHSWAP Z1 X1 X2
7	CZ	38	CZ HHSWAP Z1 Z2 CZ Z2 X2
8	Z1 Z2	39	Z1 X1 X2 X1 Z2 X1 X2
9	CZ HHSWAP	40	X2 X1 X2 HHSWAP X1 Z1 Z1
10	HHSWAP X2	41	X2 Z2 Z1 X1 HHSWAP X2 Z1
11	Z2 CZ	42	CZ Z2 X1 X1 X1 X2 CZ
12	X1 X1	43	X1 X1 X2 HHSWAP HHSWAP X2 CZ
13	X1 HHSWAP	44	X1 HHSWAP Z1 Z1 X2 Z2 Z1 CZ
14	Z1 HHSWAP X1	45	X1 X2 Z2 Z1 CZ HHSWAP X1 Z2
15	CZ Z2 CZ	46	Z2 X2 CZ Z2 Z2 Z1 Z1 Z2
16	X1 CZ X2	47	X1 Z1 Z1 Z1 X2 X2 Z2 HHSWAP
17	Z2 X2 X2	48	X1 CZ X1 Z1 X2 X2 X2 Z1
18	X2 X1 X1	49	CZ HHSWAP Z2 X2 CZ CZ HHSWAP X2
19	Z2 Z2 Z1	50	HHSWAP CZ X2 HHSWAP X2 Z1 Z1 Z2 CZ
20	Z2 X1 HHSWAP X1	51	Z1 X1 CZ Z1 Z1 Z1 X1 X1 X2
21	CZ Z2 CZ Z1	52	CZ Z2 CZ Z2 X2 HHSWAP HHSWAP X2 X2
22	Z2 Z2 Z1 Z1	53	X1 Z1 HHSWAP Z2 X2 Z1 Z2 X2 X2
23	X2 CZ Z2 Z1	54	HHSWAP X1 X2 X1 X2 HHSWAP X1 HHSWAP X1
24	Z1 Z1 CZ Z1	55	X1 X1 HHSWAP Z1 CZ HHSWAP X1 Z1 CZ
25	CZ Z1 Z2 Z1	56	X2 X1 X1 X1 X2 CZ X1 X1 X2 HHSWAP
26	CZ Z1 X1 CZ HHSWAP	57	X1 Z1 HHSWAP HHSWAP X1 CZ HHSWAP*3 X1
27	X2 HHSWAP Z2 Z2 Z2	58	X2 X2 Z1 X1 HHSWAP HHSWAP Z1 HHSWAP Z2 X2
28	HHSWAP CZ CZ HHSWAP Z2	59	Z1 Z2 Z1 X2 HHSWAP X1 X2 Z2 Z2 X1
29	Z1 CZ X2 Z2 Z2	60	Z2 X1 X1 Z1 X1 CZ Z1 HHSWAP Z1 X1
30	X1 CZ Z2 HHSWAP Z2	61	CZ Z1 CZ Z1 Z1 HHSWAP X2 HHSWAP Z1 X2
31	CZ X1 X2 CZ Z2	62	HHSWAP Z1 HHSWAP CZ CZ X1 Z2 Z2 CZ CZ X1

63	HHSWAP HHSWAP X1 HHSWAP Z2 CZ X1 X1 X2 CZ X2	108	HHSWAP *11
64	Z1 Z2 HHSWAP Z1 Z1 Z2 HHSWAP CZ Z2 X2 Z1	109	HHSWAP *12
65	CZ X1 Z1 HHSWAP CZ HHSWAP HHSWAP X1 HHSWAP CZ Z2	108	HHSWAP *11
66	CZ CZ Z2 Z1 Z2 CZ X1 Z2 HHSWAP Z2 X1	110	Z1 *1
67	HHSWAP CZ Z1 CZ X1 Z1 Z2 X2 X2 X2 HHSWAP	111	Z1 *2
68	HHSWAP X2 Z1 HHSWAP Z2 Z1 X1 HHSWAP X1 CZ X1 CZ	112	Z1 *3
69	HHSWAP X2 HHSWAP Z1 CZ HHSWAP X1 X1 Z1 Z1 CZ X1	113	Z1 *4
70	X2 X1 X1 CZ X2 Z2 Z2 Z1 X2 HHSWAP Z2 Z1	114	Z1 *5
71	Z1 X1 X2 Z2 X1 Z1 Z1 Z2 CZ Z2 X1 Z1	115	Z1 *6
72	X1 Z2 X1 Z1 HHSWAP X2 X1 HHSWAP CZ X2 Z2 HHSWAP	116	Z1 *7
73	CZ Z1 Z2 Z1 X2 X1 X1 HHSWAP Z1 X2 Z1 X1	117	Z1 *8
74	X2 *1	118	Z1 *9
75	X2 *2	119	Z1 *10
76	X2 *3	120	Z1 *11
77	X2 *4	121	Z1 *12
78	X2 *5	122	X1 *1
79	X2 *6	123	X1 *2
80	X2 *7	124	X1 *3
81	X2 *8	125	X1 *4
82	X2 *9	126	X1 *5
83	X2 *10	127	X1 *6
84	X2 *11	128	X1 *7
85	X2 *12	129	X1 *8
86	CZ *1	130	X1 *9
87	CZ *2	131	X1 *10
88	CZ *3	132	X1 *11
89	CZ *4	133	X1 *12
90	CZ *5	134	Z2 *1
91	CZ *6	135	Z2 *2
92	CZ *7	136	Z2 *3
93	CZ *8	137	Z2 *4
94	CZ *9	138	Z2 *5
95	CZ *10	139	Z2 *6
96	CZ *11	140	Z2 *7
97	CZ *12	141	Z2 *8
98	HHSWAP *1	142	Z2 *9
99	HHSWAP *2	143	Z2 *10
100	HHSWAP *3	144	Z2 *11
101	HHSWAP *4	145	Z2 *12
102	HHSWAP *5	146	(HHSWAP X2) *1
103	HHSWAP *6	147	(HHSWAP X2) *2
104	HHSWAP *7	148	(HHSWAP X2) *3
105	HHSWAP *8	149	(HHSWAP X2) *4
106	HHSWAP *9	150	(HHSWAP X2) *5
107	HHSWAP *10	151	(HHSWAP X2) *6

152	(X1 Z2) *1	192	(X2 CZ Z2)*3
153	(X1 Z2) *2	193	(X2 CZ Z2)*4
154	(X1 Z2) *3	194	(Z2 X2 Z2)*1
155	(X1 Z2) *4	195	(Z2 X2 Z2)*2
156	(X1 Z2) *5	196	(Z2 X2 Z2)*3
157	(X1 Z2) *6	197	(Z2 X2 Z2)*4
158	(Z2 Z2)	198	(X2 Z1 HHSWAP)*1
159	(Z2 Z2)*2	199	(X2 Z1 HHSWAP)*2
160	(Z2 Z2)*3	200	(X2 Z1 HHSWAP)*3
161	(Z2 Z2)*4	201	(X2 Z1 HHSWAP)*4
162	(Z2 Z2)*5	202	(X2 CZ HHSWAP)*1
163	(Z2 Z2)*6	203	(X2 CZ HHSWAP)*2
164	(HHSWAP CZ)*1	204	(X2 CZ HHSWAP)*3
165	(HHSWAP CZ)*2	205	(X2 CZ HHSWAP)*4
166	(HHSWAP CZ)*3	206	(Z1 X1 X2 Z2)*1
167	(HHSWAP CZ)*4	207	(Z1 X1 X2 Z2)*2
168	(HHSWAP CZ) *5	208	(Z1 X1 X2 Z2)*3
169	(HHSWAP CZ) *6	209	(X2 HHSWAP X2 X2)*1
170	(Z2 X2)*1	210	(X2 HHSWAP X2 X2)*2
171	(Z2 X2)*2	211	(X2 HHSWAP X2 X2)*3
172	(Z2 X2)*3	212	(Z2 X1 Z1 Z1)*1
173	(Z2 X2)*4	213	(Z2 X1 Z1 Z1)*2
174	(Z2 X2)*5	214	(Z2 X1 Z1 Z1)*3
175	(Z2 X2)*6	215	(Z2 X1 X2 X2)*1
176	(CZ X2)*1	216	(Z2 X1 X2 X2)*2
177	(CZ X2)*2	217	(Z2 X1 X2 X2)*3
178	(CZ X2)*3	218	(X2 X1 X2 CZ)
179	(CZ X2)*4	219	(X2 X1 X2 CZ)*2
180	(CZ X2)*5	220	(X2 X1 X2 CZ)*3
181	(CZ X2)*6	221	(X1 Z2 X1 CZ)*1
182	(Z1 X1 Z2)*1	222	(X1 Z2 X1 CZ)*2
183	(Z1 X1 Z2)*2	223	(X1 Z2 X1 CZ)*3
184	(Z1 X1 Z2)*3	224	(X2 X1 Z2 Z1 X1)*1
185	(Z1 X1 Z2)*4	225	(X2 X1 Z2 Z1 X1)*2
186	(HHSWAP Z2 Z1)*1	226	(HHSWAP HHSWAP Z1 CZ Z1)*1
187	(HHSWAP Z2 Z1)*2	227	(HHSWAP HHSWAP Z1 CZ Z1)*2
188	(HHSWAP Z2 Z1)*3	228	(Z2 Z1 Z2 Z2 X1)
189	(HHSWAP Z2 Z1)*4	233	(Z2 Z1 Z2 Z2 X1)*2
190	(X2 CZ Z2)*1	234	(HHSWAP Z2 Z1 Z1 Z2)*1
191	(X2 CZ Z2)*2	235	(HHSWAP Z2 Z1 Z1 Z2)*2

Table 2. The test circuit family for the 1 logical qubit simulation encoded in the [[7,1,3]] code.

No.	Content	No.	Content
1	H	42	X X
2	Z	43	X X X
3	X	44	X X X X
4	X X	45	X X X X X
5	Z X	46	X X X X X X
6	X H	47	X X X X X X X
7	H X X	48	X X X X X X X X
8	H X H	49	X X X X X X X X X
9	H H Z	50	X X X X X X X X X X
10	X X X X	51	H
11	X H X X	54	H H
12	H Z H X	55	H H H
13	H H X Z H	56	H H H H
14	Z Z H Z X	57	H H H H H
15	X Z X X X	58	H H H H H H
16	H Z Z H X Z	59	H H H H H H H
17	X Z X X H Z	60	H H H H H H H H
18	Z H X H X H	61	H H H H H H H H H
19	Z X X Z X X Z	62	H H H H H H H H H H
20	X H Z Z H Z Z	63	X Z
21	X X X X Z Z X	64	X Z X Z
22	H H X Z X Z X Z	65	X Z X Z X Z
23	H Z H X Z X H H	66	X Z X Z X Z X Z
24	X Z H H Z Z Z Z	67	X Z X Z X Z X Z X Z
25	H X Z H H X Z X H	68	H Z
26	Z H Z H Z H X Z X	69	H Z H Z
27	H Z X H H X Z Z Z	70	H Z H Z H Z
28	Z H Z H Z Z X X Z H	71	H Z H Z H Z H Z
29	H Z X Z H X X X X X	72	H Z H Z H Z H Z H Z
30	Z H Z Z Z Z X H H Z	73	X H
31	Z	74	X H X H
32	Z Z	75	X H X H X H
33	Z Z Z	76	X H X H X H X H
34	Z Z Z Z	77	X H X H X H X H X H
35	Z Z Z Z Z	78	H H Z
36	Z Z Z Z Z Z	79	H H Z H H Z
37	Z Z Z Z Z Z Z	80	H H Z H H Z H H Z
38	Z Z Z Z Z Z Z Z	81	Z X Z
39	Z Z Z Z Z Z Z Z Z	82	Z X Z Z X Z
40	Z Z Z Z Z Z Z Z Z Z	83	Z X Z Z X Z Z X Z
41	X	84	X H Z

Table 3. The test circuit family for the 2 logical qubits simulation encoded in the [[7,1,3]] code[1].

No.	Content	No.	Content
1	Y1	52	X2 H2 X2 Y2 X2 Y1
2	CNOT1	53	Y1 Y2 CNOT1 X1 Y2 X1
3	Y2	54	CNOT1 CNOT2 H1 Y2 Z1 CNOT2
4	Z2	55	X1 Z1 H2 X1 H2 H2
5	X1	56	H2 H2 Z1 Y1 CNOT2 Z1
6	X2	57	X1 X2 X2 CNOT1 CNOT1 H1
7	H2	58	Y1 Y1 Y2 Y1 CNOT2 H2
8	H1	59	Y2 Z2 Z2 X2 H2 H1
9	Z1	60	Y1 CNOT1 X2 Y2 H1 CNOT1
10	CNOT2	61	Y1 CNOT2 X1 Y2 Y1 Y2 Z1
11	H1 Y2	62	Y2 Z1 Y2 H2 Y1 CNOT2 Y2
12	H2 Y2	63	CNOT1 Y1 X1 X2 Y2 X2 H1
13	CNOT2 X2	64	CNOT2 X2 Z2 H1 Y1 H1 CNOT2
14	Z2 Z1	65	H2 H2 CNOT1 Z2 Z2 Z1 Y1
15	CNOT2 Y1	66	X1 H2 X2 CNOT2 X2 H1 X2
16	X1 Y1	67	X1 H1 CNOT1 H1 Y1 CNOT2 Z1
17	CNOT2 CNOT2	68	H1 CNOT2 Y2 X2 CNOT1 CNOT1 CNOT2
18	CNOT1 X1	69	Y1 Y2 X2 Z2 X1 H2 Y1
19	Z1 Y1	70	Y2 Z1 Y1 CNOT1 H2 Z2 Z1
20	Y2 CNOT2	71	X2 CNOT1 H1 H2 X2 CNOT2 X2 Y1
21	H1 X2 X1	72	Z2 X2 X2 Y1 Y2 H1 X1 Y1
22	Y1 H2 Y2	73	X2 CNOT1 H2 X2 H1 H1 H2 H2
23	Z2 Z1 CNOT2	74	X2 X1 X2 CNOT1 CNOT1 CNOT1 X1 Y2
24	Y1 H1 CNOT1	75	Y1 Y1 Y1 CNOT1 Z1 Z1 Y1 CNOT1
25	CNOT2 Y2 Y1	76	Y2 X2 Z2 X2 Z1 H2 H2 Z1
26	X2 Z2 CNOT2	77	Z2 CNOT2 Z2 Z2 Y1 Y2 H1 X1
27	H2 H2 Y2	78	X1 H2 Y1 Y1 X1 X2 Y1 X1
28	CNOT1 Z1 Z1	79	Y2 X1 Z1 H2 X2 H2 H1 Z2
29	CNOT1 H2 CNOT2	80	X2 Y1 Z1 X1 CNOT1 Y2 Y1 X1
30	Z1 Y1 CNOT1	81	H2 CNOT2 Z2 Z1 Y1 Z2 Z2 Y1 X2
31	H2 H1 CNOT1 Y2	82	X2 X1 H2 Z1 X1 H1 H2 CNOT1 Z1
32	H2 Z1 H1 Y1	83	CNOT1 X1 X2 CNOT1 Z2 CNOT1 Z2 CNOT1 CNOT2
33	Y2 Z1 Y2 CNOT2	84	X2 CNOT2 X1 Z2 H1 X1 CNOT2 Z2 X1
34	Z1 H1 Y1 CNOT1	85	Z1 H2 CNOT1 Y2 Y1 CNOT1 Z1 H1 X1
35	H1 Y1 H2 Y2	86	Y2 X2 Z1 X1 Y1 X1 CNOT1 X2 X1
36	Z1 H1 X1 CNOT1	87	H2 Y2 CNOT2 H2 CNOT2 H2 X2 X2 CNOT1
37	H1 CNOT1 Z1 CNOT1	88	Y2 H2 H1 H1 CNOT2 H2 Y1 Z2 Y2
38	X2 H2 CNOT1 Z1	89	Z2 Y2 Z2 CNOT2 H2 CNOT1 CNOT2 H2 CNOT1
39	Z2 Z2 X2 X2	90	Y1 Y2 X2 H1 Y2 X2 Y2 Y2 X2
40	CNOT2 Y2 Z1 X1	91	Z2 X2 X2 X2 CNOT2 X2 X2 Y1 CNOT1 Y2
41	Y1 Z2 CNOT2 Y2 X2	92	H1 Y1 X1 Y2 Z1 H2 H2 Y1 X2 CNOT1
42	Y2 Z1 Y1 H1 CNOT2	93	Z2 X2 H1 CNOT2 X1 H2 Y1 CNOT2 Y2 H1
43	Z1 X2 H2 Z1 Y2	94	X2 X2 H1 X1 H1 Y1 Z2 X1 Y1 Y2
44	H2 Z1 CNOT2 H2 H1	95	H2 CNOT2 Z2 Z1 H2 Y2 CNOT2 H2 CNOT2 CNOT1
45	Z2 Y1 H1 H1 Y2	96	X2 Y2 X1 Z1 CNOT1 Z2 X1 Z1 X1 CNOT1
46	CNOT2 X1 CNOT1 CNOT1 Y1	97	CNOT2 CNOT1 Z2 CNOT2 X1 X1 Z1 X2 X2 X2
47	H1 CNOT1 H2 Z2 Y2	98	Y2 Z2 Y1 Z2 CNOT1 CNOT1 Y1 H1 Z1 Y1
48	Z2 CNOT2 Y1 Y2 H1	99	Z1 X1 Y2 Z2 X1 Y2 X2 X2 H2 Z1
49	H2 Z1 CNOT2 CNOT2 CNOT1	100	H1 X2 H1 CNOT2 CNOT1 CNOT1 X1 X1 H1 X1
50	Z1 Y2 H2 Z1 Y2	101	Z1 CNOT1 H2 X1 H1 H2 Y2 CNOT1 Z2 Y2 CNOT1
51	Y1 X1 Y2 CNOT1 Y2 H1	102	Y1 Z2 Z2 Y2 Z2 Y2 Z2 H2 H1 X2 H2

[1] Note that CNOT1 refers to apply the CNOT from the first qubit to the second qubit, CNOT2 refers to apply the CNOT gate in the opposite way.

103	Y1 H2 CNOT2 X1 H2 Z2 H1 Y1 Z1 H2 H2	160	Y2 Z2 Z2 H1 H2 X2 H1 X1 H2 H1 H2 CNOT2 Z2
104	Y2 Z1 X2 X2 CNOT1 X2 X1 CNOT1 X2 Z1 Z2	161	H1 *1
105	Z2 X1 Y2 CNOT2 CNOT2 H2 X2 Z2 CNOT1 Z1 Y1	162	H1 *2
106	Z2 X1 CNOT1 CNOT1 H1 Z1 H1 CNOT1 X2 X1 X1	163	H1 *3
107	Y2 Y2 X1 CNOT1 H1 Z1 CNOT1 H1 Y2 CNOT2 X2	173	H1 *13
108	CNOT1 H2 H1 Z2 CNOT1 Y2 Z1 H2 CNOT1 X2 X2	174	H1 *14
109	CNOT1 Z1 X2 X2 Z2 CNOT2 X1 CNOT2 Z1 X1 Z2	175	H1 *15
110	CNOT1 H1 H1 CNOT1 Z1 CNOT1 X2 Z2 CNOT1 CNOT1 CNOT1	176	H1 *16
111	CNOT1 Z1 Z1 X1 X1 CNOT2 Z1 H1 Y1 Z1 X1 CNOT2	177	CNOT1 *1
112	H1 Y1 CNOT1 Y1 CNOT2 X2 Y1 Y2 Y1 Y2 X2 H1	178	CNOT1 *2
113	X1 Z2 CNOT2 X1 Y1 Y1 H2 H1 CNOT2 Z1 Y2 CNOT1	179	CNOT1 *3
114	Z2 Y1 CNOT1 Z1 CNOT2 X1 Y2 X2 X1 Z2 Z2 X2	180	CNOT1 *4
115	Y1 Y1 X2 Z1 Y2 Y1 CNOT2 Y2 Y1 CNOT2 Z1 CNOT1	181	CNOT1 *5
116	H1 H2 CNOT1 X1 H1 Z2 CNOT1 Z1 Z2 Z1 Y1 H2	182	CNOT1 *6
117	X2 CNOT1 Y2 H1 Z1 X1 CNOT1 X1 H1 H2 X1 Y2	183	CNOT1 *7
118	Z2 X2 H2 H1 X1 H1 CNOT1 H2 Y2 X2 H2 H2	184	CNOT1 *8
119	Y1 Y1 Y1 X2 Y1 X2 CNOT1 H1 X2 CNOT2 Z2 Y1	185	CNOT1 *9
120	CNOT1 H2 CNOT2 X1 Y1 Z1 X1 Z2 CNOT1 Z2 Z1 CNOT1	186	CNOT1 *10
121	Z1 H2 H2 X2 Y1 X2 X2 Z1 Y1 X1 H1 H2 Z2	187	CNOT1 *11
122	H1 CNOT1 H2 CNOT1 H2 CNOT2 CNOT2 H2 H1 Z1 Z1 Y2 Y1	188	CNOT1 *12
123	H1 H2 Y1 Z1 Y1 H1 H1 X2 CNOT1 Z2 H1 H1 Z2	189	CNOT1 *13
124	H2 CNOT1 Z1 X1 H1 H2 X1 CNOT2 H1 X1 H2 Z2 H2	190	CNOT1 *14
125	Y1 X1 Z1 Z2 H1 X2 H1 Y2 Y2 Y2 CNOT2 CNOT2 Y1	191	CNOT1 *15
126	X1 X1 Z1 CNOT1 Y2 Z2 Z1 CNOT1 CNOT2 X2 X1 H1 Z1	192	CNOT1 *16
127	H2 CNOT1 CNOT2 Z1 Y2 X2 X2 Y2 Z1 X1 CNOT2 X1 X2	193	(Y1) *1
128	Z1 CNOT2 CNOT1 Z1 CNOT1 Y2 CNOT1 CNOT2 X2 X2 Z1 X2	194	(Y1) *2
129	Y2 X1 Z2 X2 H1 CNOT1 CNOT2 Y2 CNOT1 X2 X1 Z2 CNOT1	195	(Y1) *3
130	Z2 H1 Y1 H1 CNOT1 Y1 H2 H2 CNOT1 CNOT2 Z2 Y2 Z2	196	(Y1) *4
131	H1 Z1 Z2 Y2 X2 Z1 Z2 CNOT1 Y2 CNOT2 X2 Z1 CNOT1 H2	197	(Y1) *5
132	H2 X1 Y2 Z1 Z2 CNOT2 CNOT1 CNOT2 Y1 H1 Y2 CNOT2 Y1 CNOT1	198	(Y1) *6
133	Y2 Y2 CNOT2 Y1 CNOT1 H2 CNOT1 Y1 H2 Y1 H2 CNOT1 CNOT1 H1	199	(Y1) *7
134	H1 X2 H2 Z2 X2 X1 Y2 Z1 X1 CNOT1 Y2 Y1 CNOT2 H2	200	(Y1) *8
135	Z1 X1 CNOT2 H1 Z1 CNOT2 Z1 H1 Z1 CNOT2 Y2 Z1 Z2 H1	201	(Y1) *9
136	Z1 X1 Z1 Z1 Z2 Z1 Z1 Y2 Z2 CNOT1 Y1 Y2 Z1 X2	202	(Y1) *10
137	X2 CNOT2 CNOT2 CNOT2 CNOT2 H2 CNOT2 Y2 Z1 CNOT2 CNOT1 CNOT1	203	(Y1) *11
138	Z2 X1 CNOT1 Y2 H1 H1 Z1 H2 X1 H1 X1 X1 CNOT2 CNOT2	204	(Y1) *12
139	CNOT1 Z2 H2 Z2 X1 H2 H1 X2 H2 X2 X2 CNOT1 CNOT1 CNOT2	205	(Y1) *13
140	H1 CNOT2 H1 CNOT1 X1 Y2 X2 H1 H2 CNOT1 CNOT1 H1 H2 H2	206	(Y1) *14
141	H2 Z2 Z1 Y1 Y2 H2 Y1 CNOT1 CNOT2 Z1 Z1 Y2 Y1 X2 X1	207	(Y1) *15
142	Y2 Z1 X2 CNOT2 X1 CNOT1 Y2 Z2 Y2 Y1 CNOT1 X1 Y1 Y2 X1	208	(Y1) *16
143	X1 CNOT1 CNOT2 CNOT2 Y1 Y1 CNOT2 H2 Z1 X1 H1 Y1 H2 Z2 X2	209	X1 *1
144	Y1 X2 Y2 Z1 CNOT1 H1 Z1 CNOT1 X2 H1 H2 Y1 Z2 CNOT2 Z2	210	X1 *2
145	Y1 Y2 H1 Y2 X1 CNOT2 Z1 CNOT2 Y1 Y2 Z1 CNOT2 CNOT1 X1 Z2	211	X1 *3
146	H1 X2 X2 CNOT1 H2 Z2 Z2 Y1 Y1 H2 Y2 CNOT1 X2 CNOT2 CNOT1	212	X1 *4
147	H2 H2 Z1 CNOT1 CNOT1 X2 Z1 H1 X2 Z2 Z2 CNOT1 Z2 Y2 Y2	213	X1 *5
148	Y1 CNOT2 Y2 X1 CNOT1 H2 X2 CNOT2 Z1 Y1 CNOT1 X1 H2 X1 X2	214	X1 *6
149	CNOT2 Y2 X2 CNOT1 Y2 Y1 X1 Y2 Y1 H2 Y1 X2 Z1 Y2 H2	215	X1 *7
150	H2 H1 X2 Z2 X2 X1 CNOT1 Y2 X1 Y2 CNOT1 Y2 CNOT1 H1 CNOT2	216	X1 *8
151	H1 CNOT1 X2 H2 CNOT1 Z1 CNOT2 CNOT1 Z1 Y2 Z2 CNOT2 H1 Y2 CNOT1	217	X1 *9
152	H1 CNOT1 Y1 Y2 Z2 X1 Y1 H1 Y1 Z2 H1 Y1 X2 X2 H1 H1	218	X1 *10
153	X1 H2 H2 Z1 Z2 Z2 Z1 Y1 Y2 H2 Z1 H1 X1 H2 X1 Y2	219	X1 *11
154	CNOT1 Z1 X1 X1 X1 CNOT2 Z2 Y2 Y2 X2 CNOT2 H1 CNOT2 Y2 H1 Y2	220	X1 *12
155	Y1 CNOT1 X2 Y2 CNOT1 X1 Y2 Z2 X1 Z2 CNOT2 Y1 Y2 H2 X1 Z2	221	X1 *13
156	H1 H2 Z1 Y2 X1 CNOT1 Y2 H1 Y1 CNOT2 Y1 Z1 Z1 H2 Z1 Y2	222	X1 *14
157	CNOT1 X2 H1 CNOT2 X2 H1 CNOT1 Z1 CNOT1 CNOT1 Y2 Y1 Y1 H2 Z1 Y2	223	X1 *15
158	X2 Y2 Z2 Y2 X2 Y2 Z1 H2 X1 Y2 X1 Y2 Y1 Y1 H2 Y1	224	X1 *16
159	H2 CNOT1 X1 CNOT1 Z2 CNOT2 Z1 H1 Y1 Y2 Y2 Y2 H2 CNOT2 H1 CNOT2	225	Z1 *1

226	Z1 *2	283	(Z2 H1) *3
227	Z1 *3	284	(Z2 H1) *4
228	Z1 *4	285	(Z2 H1) *5
229	Z1 *5	286	(Z2 H1) *6
230	Z1 *6	287	(Z2 H1) *7
231	Z1 *7	288	(Z2 H1) *8
232	Z1 *8	289	(CNOT2 X2)*1
233	Z1 *9	290	(CNOT2 X2)*2
234	Z1 *10	291	(CNOT2 X2)*3
235	Z1 *11	292	(CNOT2 X2)*4
236	Z1 *12	293	(CNOT2 X2)*5
237	Z1 *13	294	(CNOT2 X2)*6
238	Z1 *14	295	(CNOT2 X2)*7
239	Z1 *15	296	(CNOT2 X2)*8
240	Z1 *16	297	(Z1 H1) *1
241	(X1 CNOT1)*1	298	(Z1 H1) *2
242	(X1 CNOT1)*2	299	(Z1 H1) *3
243	(X1 CNOT1)*3	300	(Z1 H1) *4
244	(X1 CNOT1)*4	301	(Z1 H1) *5
245	(X1 CNOT1)*5	302	(Z1 H1) *6
246	(X1 CNOT1)*6	303	(Z1 H1) *7
247	(X1 CNOT1)*7	304	(Z1 H1) *8
248	(X1 CNOT1)*8	305	(H2 X2) *1
249	(X1 Y2)*1	306	(H2 X2) *2
250	(X1 Y2)*2	307	(H2 X2) *3
251	(X1 Y2)*3	308	(H2 X2) *4
252	(X1 Y2)*4	309	(H2 X2) *5
253	(X1 Y2)*5	310	(H2 X2) *6
254	(X1 Y2)*6	311	(H2 X2) *7
255	(X1 Y2)*7	312	(H2 X2) *8
256	(X1 Y2)*8	313	(Z2 H1 Y1) *1
257	(H1 Z1) *1	314	(Z2 H1 Y1) *2
258	(H1 Z1) *2	315	(Z2 H1 Y1) *3
259	(H1 Z1) *3	316	(Z2 H1 Y1) *4
260	(H1 Z1) *4	317	(Z2 H1 Y1) *5
261	(H1 Z1) *5	318	(H1 CNOT2 H1)*1
262	(H1 Z1) *6	319	(H1 CNOT2 H1)*2
263	(H1 Z1) *7	320	(H1 CNOT2 H1)*3
264	(H1 Z1) *8	321	(H1 CNOT2 H1)*4
265	(H1 CNOT2) *1	322	(H1 CNOT2 H1)*5
266	(H1 CNOT2) *2	323	(Y2 CNOT1 H2)*1
267	(H1 CNOT2) *3	324	(Y2 CNOT1 H2)*2
268	(H1 CNOT2) *4	325	(Y2 CNOT1 H2)*3
269	(H1 CNOT2) *5	326	(Y2 CNOT1 H2)*4
270	(H1 CNOT2) *6	327	(Y2 CNOT1 H2)*5
271	(H1 CNOT2) *7	328	(CNOT1 Y2 CNOT2)*1
272	(H1 CNOT2) *8	329	(CNOT1 Y2 CNOT2)*2
273	(Z1 H2) *1	330	(CNOT1 Y2 CNOT2)*3
274	(Z1 H2) *2	331	(CNOT1 Y2 CNOT2)*4
275	(Z1 H2) *3	332	(CNOT1 Y2 CNOT2)*5
276	(Z1 H2) *4	333	(CNOT1 Z2 X2) *1
277	(Z1 H2) *5	334	(CNOT1 Z2 X2) *2
278	(Z1 H2) *6	335	(CNOT1 Z2 X2) *3
279	(Z1 H2) *7	336	(CNOT1 Z2 X2) *4
280	(Z1 H2) *8	337	(CNOT1 Z2 X2) *5
281	(Z2 H1) *1	338	(X1 H1 X2) *1
282	(Z2 H1) *2	339	(X1 H1 X2) *2

340 (X1 H1 X2) *3
341 (X1 H1 X2) *4
342 (X1 H1 X2) *5
345 (CNOT2 X1 CNOT1)*1
346 (CNOT2 X1 CNOT1)*2
347 (CNOT2 X1 CNOT1)*3
346 (CNOT2 X1 CNOT1)*4
347 (CNOT2 X1 CNOT1)*5
348 (Z1 Y1 Z2)*1
349 (Z1 Y1 Z2)*2
350 (Z1 Y1 Z2)*3
351 (Z1 Y1 Z2)*4
352 (Z1 Y1 Z2)*5
353 (H1 Z2 H2) *1
354 (H1 Z2 H2) *2
355 (H1 Z2 H2) *3
356 (H1 Z2 H2) *4
357 (H1 Z2 H2) *5
358 (CNOT1 X1 CNOT2)*1
359 (CNOT1 X1 CNOT2)*2
360 (CNOT1 X1 CNOT2)*3
361 (CNOT1 X1 CNOT2)*4
362 (CNOT1 X1 CNOT2)*5
363 (CNOT1 Y1 H2 Z2)*1
364 (CNOT1 Y1 H2 Z2)*2
365 (CNOT1 Y1 H2 Z2)*3
366 (CNOT1 Y1 H2 Z2)*4
367 (H1 Z2 Z1 Y2)*1
368 (H1 Z2 Z1 Y2)*2
369 (H1 Z2 Z1 Y2)*3
370 (H1 Z2 Z1 Y2)*4
371 (Y2 Y2 Z1 H1) *1
372 (Y2 Y2 Z1 H1) *2
373 (Y2 Y2 Z1 H1) *3
374 (Y2 Y2 Z1 H1) *4
375 (Z1 Y2 Y2 X1)*1
376 (Z1 Y2 Y2 X1)*2
377 (Z1 Y2 Y2 X1)*3
378 (Z1 Y2 Y2 X1)*4
379 (Y1 CNOT2 Y2 CNOT2)*1
380 (Y1 CNOT2 Y2 CNOT2)*2
381 (Y1 CNOT2 Y2 CNOT2)*3
382 (Y1 CNOT2 Y2 CNOT2)*4
383 (Y2 Y1 H2 X1)*1
384 (Y2 Y1 H2 X1)*2
385 (Y2 Y1 H2 X1)*3
386 (Y2 Y1 H2 X1)*4
387 (Y1 X2 X1 Z2)*1
388 (Y1 X2 X1 Z2)*2
389 (Y1 X2 X1 Z2)*3
390 (Y1 X2 X1 Z2)*4
391 (CNOT2 H2 Y2 X2)*1
392 (CNOT2 H2 Y2 X2)*2
393 (CNOT2 H2 Y2 X2)*3
394 (CNOT2 H2 Y2 X2)*4
395 (H2 Z2 H2 H1)*1
396 (H2 Z2 H2 H1)*2
397 (H2 Z2 H2 H1)*3
398 (H2 Z2 H2 H1)*4
399 (H2 H1 Y2 CNOT1)*1
400 (H2 H1 Y2 CNOT1)*2
401 (H2 H1 Y2 CNOT1)*3
402 (H2 H1 Y2 CNOT1)*4
403 (X2 Z2 CNOT1 H1 Y1)*1
404 (X2 Z2 CNOT1 H1 Y1)*2
405 (X2 Z2 CNOT1 H1 Y1)*3
406 (CNOT1 H2 X2 Z1 X2)*1
407 (CNOT1 H2 X2 Z1 X2)*2
408 (CNOT1 H2 X2 Z1 X2)*3
409 (X1 X1 Y1 H2 Z1)*1
410 (X1 X1 Y1 H2 Z1)*2
411 (X1 X1 Y1 H2 Z1)*3
412 (CNOT1 CNOT1 Z2 H2 H2)*1
413 (CNOT1 CNOT1 Z2 H2 H2)*2
414 (CNOT1 CNOT1 Z2 H2 H2)*3
415 (X1 H1 H2 Z2 Y1)*1
416 (X1 H1 H2 Z2 Y1)*2
417 (X1 H1 H2 Z2 Y1)*3
418 (CNOT1 CNOT1 Z2 Z1 Y2)*1
419 (CNOT1 CNOT1 Z2 Z1 Y2)*2
420 (CNOT1 CNOT1 Z2 Z1 Y2)*3
421 (H2 H2 Y2 CNOT2 Z2)*1
422 (H2 H2 Y2 CNOT2 Z2)*2
423 (H2 H2 Y2 CNOT2 Z2)*3
424 (X1 X2 H1 H1 Y2)*1
425 (X1 X2 H1 H1 Y2)*2
426 (X1 X2 H1 H1 Y2)*3
427 (Y2 CNOT2 Y1 H2 CNOT2)*1
428 (Y2 CNOT2 Y1 H2 CNOT2)*2
429 (Y2 CNOT2 Y1 H2 CNOT2)*3
430 (Z1 CNOT2 H2 Z2 Z1)*1
431 (Z1 CNOT2 H2 Z2 Z1)*2
432 (Z1 CNOT2 H2 Z2 Z1)*3
433 (X1 CNOT1 Y1 Z1 H2 Y2)*1
434 (X1 CNOT1 Y1 Z1 H2 Y2)*2
435 (X1 CNOT2 X2 X1 CNOT2 H1)*1
436 (X1 CNOT2 X2 X1 CNOT2 H1)*2
437 (CNOT1 CNOT1 H1 Z2 CNOT2 Z2)*1
438 (CNOT1 CNOT1 H1 Z2 CNOT2 Z2)*2
439 (Z1 X1 CNOT2 X2 CNOT1 Y2)*1
440 (Z1 X1 CNOT2 X2 CNOT1 Y2)*2
441 (CNOT1 CNOT1 Y1 X2 Z1 Z1)*1
442 (CNOT1 CNOT1 Y1 X2 Z1 Z1)*2
443 (Z2 H1 X1 Y1 Y2 X2)*1
444 (Z2 H1 X1 Y1 Y2 X2)*2
445 (CNOT1 X2 CNOT1 Y2 X1 X1)*1
446 (CNOT1 X2 CNOT1 Y2 X1 X1)*1
447 (CNOT1 Y2 Y1 Z2 X1 Z1)*1
448 (CNOT1 Y2 Y1 Z2 X1 Z1)*2
449 (Y2 H1 CNOT1 Y1 H1 X2)*1
450 (Y2 H1 CNOT1 Y1 H1 X2)*2
451 (H1 H1 H1 Y1 H2 Y2)*1

452	(H1 H1 H1 Y1 H2 Y2)*2