Master Thesis 2011

Surface Code Quantum Computation on a Defective Physical Lattice

Keio University

Name: Shota Nagayama

Abstract of Master's Thesis

Academic Year 2011

Surface Code Quantum Computation on a Defective Physical Lattice

This thesis proposes and evaluates extensions to surface code quantum computation for dealing with fabrication defects that result in faulty qubit devices.

In the quantum computation research field today, fault-tolerant quantum computation is required to deal with imperfections such as physical state error, dynamic loss of the qubit carrier and static loss of the device. It has been shown that the surface code, which is a type of fault-tolerant quantum computation, is highly tolerant to physical quantum state errors. It also has been shown that the surface code in principle is robust against loss errors, but how tolerant it is in realistic situations has not been investigated. In particular, state imperfections and device faults both increase the probability of logical errors, but the balance between the two has not been studied.

In this thesis a method is given for dealing with loss in the surface code. Large amounts of loss can be accommodated by using "superplaquettes" for error correction around faulty qubits. Numerical results give the decrease in error correction capabilities of the lattice as an "effective code distance" for topological qubits and the *threshold* which is the physical error rate where the logical error rate exceeds the physical error rate.

Keywords :
1. Surface code quantum computation, 2. Faulty device, 3. Static loss

Keio University

Shota Nagayama

修士論文要旨 - 2011 年度 (平成 23 年度)

Surface Code Quantum Computation on a Defective Physical Lattice

本論文では量子計算符号である Surface Code Quantum Computation を、量子コンピュータ製造の際に計算チップに必ず発生すると考えられているデバイス欠陥に耐性のあるものに拡張し、評価をおこなう。

今日研究されているフォールトトレラント量子計算では、量子状態エラーや情報を書き込んでいる量子の喪失、もしくは計算チップ上に存在する、製造時の欠陥により情報を書き込むための量子を捉えることが出来ないデバイスに対して耐性のあるシステムが必要とされている。Surface Code はフォールトトレラント量子計算の一種であり、量子状態エラーに特に高い耐性を持つとともに、計算中における量子の動的な喪失にも耐性があることが示されている。しかしながら、現実に起こりうる全ての誤りを考慮に入れた研究は未だなされていない。特に、量子状態エラーと欠陥デバイスの関係は全く研究されたことがない。

本論文では，欠陥デバイスに対応する手法として、量子喪失への対応に利用されている”Superplaquette”を応用する。また、定量評価として二種類のシミュレーションをおこなう。まず、デバイス欠陥があるときのエラー訂正能力が、デバイス欠陥がないときのどの符号距離のエラー訂正能力に対応するかを示す”有効符号距離”を示す。後に、デバイス欠陥が存在する際において、符号距離が有効に働き始める物理エラー確率の閾値を計算する。

本論文の成果によって、量子計算を実現するために必要な現実的なパラメータが明らかとなる。

キーワード
1．Surface Code Quantum Computation, 2．faulty device, 3．static loss

慶應義塾大学 政策・メディア研究科

永山 翔太

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computation today is about to reach the limits of improvement. There are two main directions in which computer technology continues to evolve; composing calculation units (CPUs or GPUs or full sets of a computers) in parallel or fabricating more transistors on a computation chip. Parallelization will have overhead, and it will cause a limit to improvement [1]. To fabricate more and more transistors on a chip, the transistors are fabricated smaller and smaller. Moore's law predicted the improvement of the number of transistors and it has been improved along the law[2]. The line width in VLSI fabrication processes has reached 22nm and it is not so large from the point of view of quantum effects and molecular structure. For example, the crystal lattice cell size of silicon is about 0.5nm. Quantum effect will cause fatal errors on the classical, traditional computers we are using today, thus Si lattice size limits the improvement of computation today. The tunnel effect and the probabilistic nature of quantum behavior cannot be tolerated by classical methods. Quantum computation is one possible solution against this problem. It uses quantum effect to compute.

1.1 Quantum Computation Today

Research on quantum computation became exciting when Shor's algorithm was found in 1994. Shor's algorithm can effectively factor large numbers into prime numbers in polynomial computational complexity, while no method is known for factoring in polynomial time on classical computers[3].

Toward quantum computation, many research projects have been done from the point of view of both physics and computer science and of both theory and experiment. The largest obstacle to quantum computation is called "decoherence". Decoherence is a quantum effect that changes the quantum state along time, caused by the susceptibility of quantum state to noise. Several methods has been considered to deal with decoherence,

for example, quantum error correction, dynamic decoupling, and decoherence free subspaces. Overall, the study of these methods is called *fault tolerant quantum computation* (FTQC)[4].

Fully scalable quantum computers are required to solve meaningful problems because small scale quantum computers with only a few qubits cannot process large programs. For example, processing Shor's algorithm to factor a number represented with 2048 bits needs 10^5 near-perfect qubits [5]. Many architectures have been proposed for a scalable quantum computer. Their feasibility depends on the physical systems in which they are implemented and the physical operations they use.

The surface code is one of the most feasible current proposals for FTQC, requiring a special type of entangled quantum state known as a 2-D cluster state[6][7][8]. This scheme defines the cluster state in terms of "plaquettes" which consist of four qubits on the lattice and are the unit for a stabilizer (error syndrome) measurements. Errors on the qubits are watched by the stabilizer. The surface code defines logical qubits using pairs of defects in the lattice. A *defect* is a region of the lattice where the physical qubits are measured and not stabilized. This leaves a degree of freedom which can be used as a qubit. Error syndromes are measured for each of these plaquettes. If errors are detected on two separate plaquettes, by connecting the two plaquettes with a chain of the same operation as the error, the lattice can recover from the error. In the surface code the longer the distance between these defects is, the more tolerant the error correction is, because an error chain connecting two holes will be a logical error.

In fact, the problems we face concerning errors are not only that qubits can easily be changed but also that some qubits may be lost completely. Some examples of loss mechanisms are: hard faults in the devices when trapping qubit carriers such as single electron; and photon generation failure. The surface code is robust against unintended changes of quantum state, assuming complete construction of qubits, but as originally defined did not take into account these loss errors.

The required cluster state can be made with a nearest neighbor architecture. The nearest neighbor architecture uses quantum interactions only between nearest-neighboring qubits, so it has high feasibility and gives the quantum processor easy extendability by adding one more set of qubits and control devices along an edge. There are many proposals for the architecture to make cluster state on which surface code runs. However, each of them suffers from the problems we mentioned above: if an existence problem occurs, there will be a hole in the cluster state. Jones et al. proposed an architecture for scalable quantum computation with quantum dots[9]. When they work, self-assembled quantum dots are used to trap electrons which are used as qubits. Unfortunately there will be defective quantum dots which cannot trap electrons, leaving holes in the cluster state. Lindner et al. showed cluster state generation by a device they called the photon machine

gun[10]. Their idea is to have a quantum dot emit photons continuously with certain operations, they will then form cluster state in 3-D. If a quantum dot fails to emit a photon at a certain time step, there will be a hole in the cluster state. Finally, Devitt et al. proposed an efficient design for constructing photonic topological cluster state with photon-photon coupling. There will be a hole in the cluster state because of the less-than-perfect probability of the coupling cite.

It has recently been shown by Stace et al that qubit loss is acceptable when performing the surface code quantum computation[11][12]. The lattice can be fixed by restoring lost qubits, but this approach cannot be achieved if the devices to trap the quantums does not work at all. The research of Stace et al. adjusted the surface code to the sudden loss of qubits during processing but countermeasures against the absence of qubits from the start to the end of the computation are required.

1.2 Contributions

In this thesis, we define the concept of a "superplaquette" which consists of several plaquettes around faulty lattice qubits, and the concept of a "donut plaquette" which is a defective plaquette whose syndrome qubit is faulty. We show the acceptability for such special plaquettes by describing some stabilizer circuits and a graph of the relationship among the yield, code distance and effective code distance when the logical error rate of the actual code distance is compared to a defect-free processor. To show the acceptability, we checked error chains around superplaquettes to see whether they generate undetectable error chain terminations and revealed the relationship between the code distance and the effective code distance. This was done by adaptation of the dynamic error management code of Fowler.

1.3 Thesis architecture

This thesis is composed of 7 chapters. Chapter 1 is this introduction. Chapter 2 explains basic technologies around the surface code. The physical implementations are mentioned in it. Chapter 3 states the problem solved in this thesis. Chapter 4 describes the solution against the problem. Chapter 5 describes tools built in this research and shows the evaluation with graphs output by the tools. Chapter 6 summarize the result of this thesis and states future works.

Chapter 2

Surface code quantum computation

Surface code quantum computation is regarded as a promising technique for fault tolerant quantum computation. It is robust against physical quantum state error and it has been shown that the threshold (a physical error rate where the logical error rate rises above the physical error rate) is nearly 1%. In this chapter, the logic of the surface code is described from the point of view of the necessary operations.

2.1 Basics of quantum computation for error correction

2.1.1 Qubit description

Quantum mechanics has been revealed and defined that the measured values of quanta will be determined through a probabilistic way and the states of a quantum is described with the probability, involving imaginary number. When we measure a quantum, a value will be determined depending on the probability and it will be observed. It also means that a quantum state cannot be reproduced from the measured value, because we cannot know the probability from the measured value. For a qubit encoded on such a quantum state, the wave function of the qubit can be described using the state vector or Dirac ket notation,

$$| \Psi\rangle = \alpha \mid 0\rangle + \beta \mid 1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{2.1}$$

Here, $\mid 0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\mid 1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and α and β are imaginary numbers. This quantum state has two degrees of freedom called the value and the phase. Figure 2.1 shows the visualization of the state space. The quantum state will be a point on the surface of the sphere, called the Bloch sphere. Because each point on the surface represents a quantum

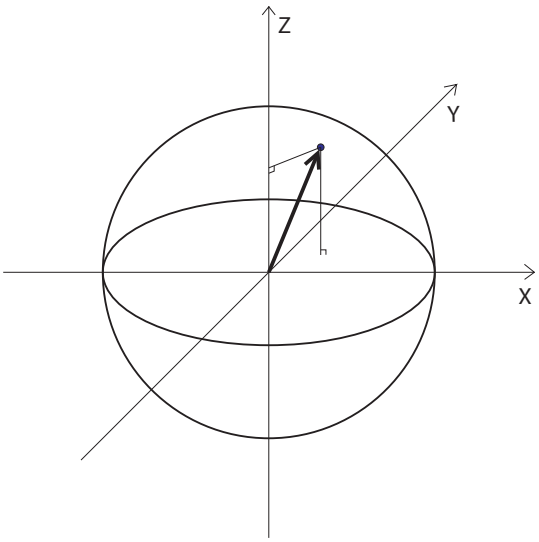

Figure 2.1: A visualization of the state space of a qubit known as the Bloch sphere. The X-Z plane is real and the Y-Z plane is imaginary. For any state represented by a vector on the sphere, the vector pointing in the opposite direction is the orthogonal state.

state, the number of states a qubit can be in is infinite. We can choose any axis to measure a qubit, though by convention we limit our choice to one of the three axes, X, Y or Z, without loss of generality. When we measure a qubit along an axis, a value which corresponds to the maximum or minimum value of the axis in the state space will be measured probabilistically. There are three matrices, one corresponding to each axis.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.2}$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{2.3}$$

and

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.4}$$

The eigenvectors of Z are

$$|\,0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{2.5}$$

$$|\,1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.6}$$

and the probability that 0 will be measured value if we measure $\mid \Psi\rangle$ can be calculated as

$$\langle 0 \mid \Psi \mid 0\rangle = \mid \alpha \mid^2, \tag{2.7}$$

similarly, the probability of 1 can be calculated as

$$\langle 1 \mid \Psi \mid 1\rangle = \mid \beta \mid^2 \tag{2.8}$$

Since there are only two possible outcomes, $\mid \alpha \mid^2 + \mid \beta \mid^2$ must be 1. To keep this relationship, all qubit operations must be unitary. Each base can be described with other two basis so that X and Z are chosen generally.

The notation for a single qubit is described above. The notation for multiple qubits is similar.

$$\begin{aligned} \mid \Phi\rangle &= \alpha \mid 0_a 0_b\rangle + \beta \mid 0_a 1_b\rangle + \gamma \mid 1_a 0_b\rangle + \omega \mid 1_a 1_b\rangle \\ &= \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \omega \end{pmatrix}, \end{aligned} \tag{2.9}$$

where a is the first qubit and b is the second qubit. As above, the square of each coefficient is the probability of the state being measured. The sum of the squares of coefficients is 1. For example, one possible state is written as

$$\begin{aligned} \mid \Phi\rangle &= \frac{1}{\sqrt{2}}(\mid 0_a\rangle + \mid 1_a\rangle) \times \frac{1}{\sqrt{2}}(\mid 0_b\rangle + \mid 1_b\rangle) \\ &= \frac{1}{2} \mid 0_a 0_b\rangle + \frac{1}{2} \mid 0_a 1_b\rangle + \frac{1}{2} \mid 1_a 0_b\rangle + \frac{1}{2} \mid 1_a 1_b\rangle \\ &= \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \end{aligned} \tag{2.10}$$

the possible measured values and the corresponding probabilities are shown in table 2.1.

2.1.2 Qubit operation

Single qubit operations

The three matrices described above also act as basic operations on a single qubit. For example, X exchanges the coefficients of $\mid 0\rangle$ and $\mid 1\rangle$.

$$X \mid \Psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times (\alpha \mid 0\rangle + \beta \mid 1\rangle) = \beta \mid 0\rangle + \alpha \mid 1\rangle \tag{2.11}$$

Table 2.1: An example of the state and probability table of two qubits

state	probability
$\mid 00\rangle$	25%
$\mid 01\rangle$	25%
$\mid 10\rangle$	25%
$\mid 11\rangle$	25%

Because $\mid 0\rangle \rightarrow \mid 1\rangle$ and $\mid 1\rangle \rightarrow \mid 0\rangle$, this is also called a NOT gate.

X, Y, Z and I are called Pauli operators. I is the identity gate

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.12}$$

Another basic gate known as the Hadamard gate is

$$H = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.13}$$

Here, the eigenstates of X are $\mid +\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\mid -\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$. With this H and the eigenvalues of Z and X,

$$H \times (\alpha \mid 0\rangle + \beta \mid 1\rangle) = (\alpha \mid +\rangle + \beta \mid -\rangle) \tag{2.14}$$

is derived. H swaps the relative relationships of the state with Z axis and with X axis.

These five single qubit gates are often used in quantum error correction.

CNOT gate

The easiest two qubit gate to understand is controlled-not (CNOT) gate:

$$CNOTgate = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.15}$$

This gate exchanges the coefficient of $\mid 0\rangle$ and $\mid 1\rangle$ of the second qubit, only if the first qubit is $\mid 1\rangle$. Table 2.2 shows the truth table of the CNOT gate.

Table 2.2: The truth table of the CNOT gate

a_{in}	b_{in}	a_{out}	b_{out}
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 2.3: State and probability table of entangled two qubits

state	probability
$\mid 00\rangle$	50%
$\mid 01\rangle$	0%
$\mid 10\rangle$	0%
$\mid 11\rangle$	50%

SWAP gate

The SWAP gate also often appears.

$$(SWAPgate) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

This operation swaps the state of the first qubit and the second qubit completely.

2.2 Entanglement

What we can find from the notation of two qubits is that the probabilities of the two qubits can be dependent like:

$$\mid \Phi\rangle = \frac{1}{\sqrt{2}} \mid 0_a 0_b\rangle + 0 \mid 0_a 1_b\rangle + 0 \mid 1_a 0_b\rangle + \frac{1}{\sqrt{2}} \mid 1_a 1_b\rangle, \tag{2.17}$$

The probabilities in table 2.3 correspond to this state. In this state, 0 must be measured on the second qubit if 0 is measured on the first qubit and 1 must be measured on the second qubit if 1 is measured on the first qubit. The opposite is also true. This characteristic is called *entanglement* and it is one of the critical effects that distinguishes quantum systems from classical ones. The entangled state cannot be described in the

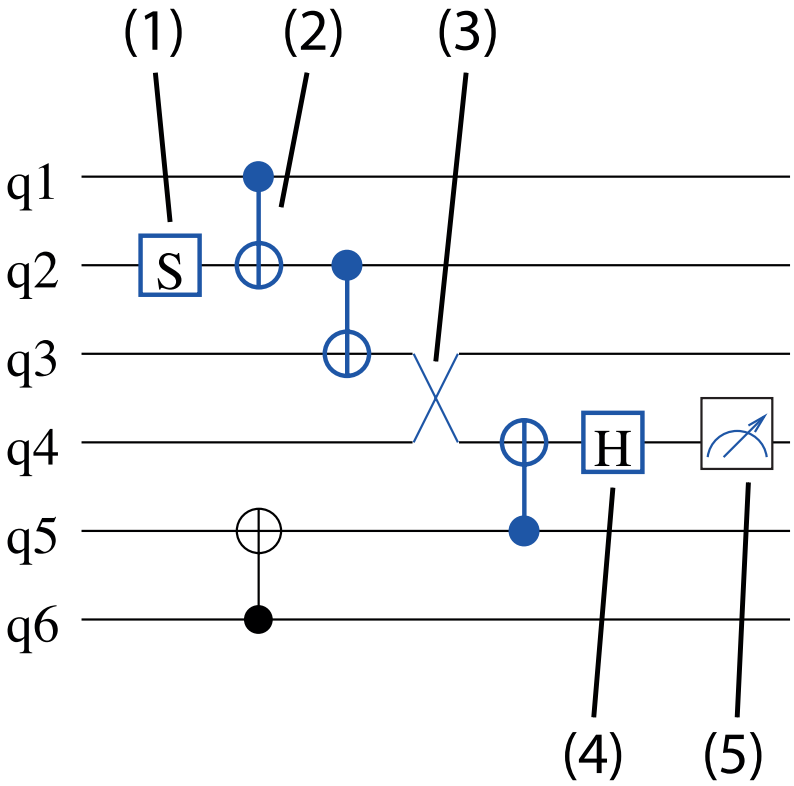

Figure 2.2: An example of a quantum circuit. (1) Initialization gate. (2) CNOT gate. The dot marks the controlled qubit and the circle marks the target qubit. (3) SWAP gate. (4) Hadamard gate. (5) Measurement in Z basis.

product of single qubit state as in equation 2.10. This is the reason why the probabilities of the two qubit is dependent. More and more qubits can be connected by entanglement.

2.2.1 Quantum circuit

Figure 2.2 shows the quantum circuit. Each horizontal line describes a physical qubit. Gates are sequentially operated from the left to the right like a music notation. The symbols of remarkable gates for this research are described in Figure 2.2.

2.3 Possible errors

Errors can be factored into two types of errors; X errors and Z errors. Actually, decoherence changes the probability of measured values. Thus, quantum errors should be analog. However, quantum error correcting codes are designed to measure whether or not an error occurs. Decoherence occurs in analog fashion, but after error correcting

operations, there will be only two possibilities; either the qubit is in error and we have detected the error, or it has returned to the original state. We say that this process "projects" the system into one of the states.

2.4 Stabilizer code

There are operators which do not change certain qubit states. For example, X does not change $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$:

$$X \times \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \times \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \tag{2.18}$$

Such operators are called stabilizer generators of the state. A full set of stabilizer generators can determine a qubit state so that it can be used as a compact notation for describing quantum states.

Stabilizers also can be used for error correction. A couple of qubits is the smallest set to stabilize. To correct an X error on $\frac{1}{\sqrt{2}}(|0_a 0_b\rangle + |1_a 1_b\rangle)$:

1. $|\Psi_0\rangle = (CNOTgate)_{ac} \times \frac{1}{\sqrt{2}}(|0_a 0_b\rangle + |1_a 1_b\rangle) \otimes |0_c\rangle$
2. $|\Psi_1\rangle = (CNOTgate)_{bc} \times |\Psi_0\rangle$
3. measure qubit c in $|\Psi_1\rangle$ in the Z basis

This process is an applied usage of the stabilizer. By repeating these operations, we can detect X error either on qubit a or on qubit$b$ by finding the change of measured value of qubit c.

To correct a Z error:

1. $|\Psi_0\rangle = (CNOTgate)_{ca} \times \frac{1}{\sqrt{2}}(|0_a 0_b\rangle + |1_a 1_b\rangle) \otimes |+_c\rangle$
2. $|\Psi_1\rangle = (CNOTgate)_{cb} \times |\Psi_0\rangle$
3. measure qubit c in $|\Psi_1\rangle$ in the X basis

With this process we can detect a Z error either on qubit a or qubit b, the same as above.

Only a single qubit is stabilized in this example, but in the surface code we will use sets of four qubits.

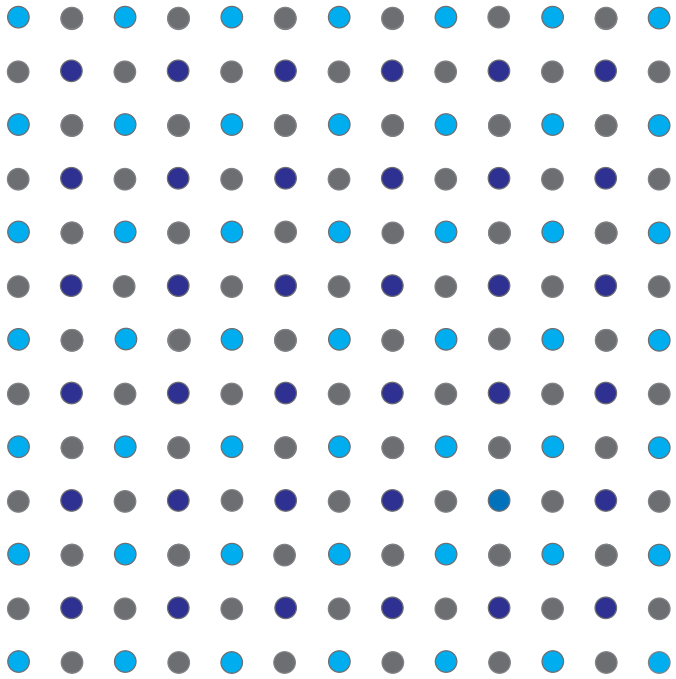

Figure 2.3: Physical placement of qubits on a lattice. Here, color has no meaning.

2.5 Surface code

In this section, I describe the surface code from the point of view of operations. The surface code has two advantages:

High feasibility Requires only nearest neighbor interaction.

High threshold Can tolerate relatively high probability of physical state errors

For simplicity, a specific surface code called the planar code, which encodes a single qubit on a lattice is described at first.

2.5.1 Cluster state

Surface code is a way to encode logical qubits on a form of cluster state[13]. A generic cluster state consists of many qubits. In a regular lattice, each qubit is entangled with its neighbors giving a specific large, entangled state. The surface code is characterized by how to encode the logical qubits and how to fix the physical errors which occur on physical qubits on the cluster state. Figure 2.4 is an example of a quantum computation chip for the surface code. Qubits are placed on a lattice on which two-qubit operations can be executed between neighboring qubits.

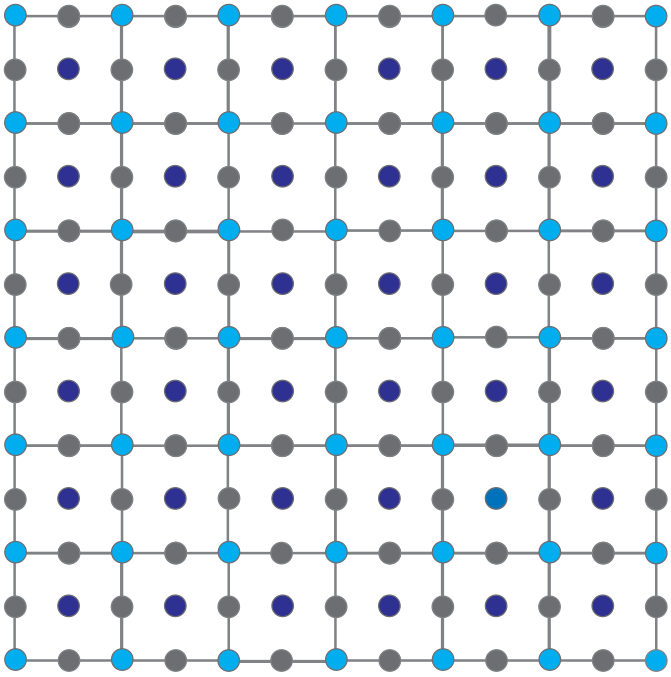

Figure 2.4: The units for surface code error correction. Gray qubits are data qubits. Blue qubits are syndrome qubits. There are two types of units, faces and vertexes. Each unit is composed of five qubits with the syndrome qubit at the center. Each data qubit belongs to two face units and two vertex units.

2.5.2 The units for error correction

Figure 2.5 shows the roles of qubits and the units for the error correction. Gray qubits are data qubits, which hold the information of the logical qubit. Blue qubits are syndrome qubits, used to stabilize data qubits. There are two types of stabilizers, Z-stabilizer and X-stabilizer. The lines divides the lattice into many units, faces and vertexes. In each unit, either a Z-stabilizer or an X-stabilizer is repeatedly executed. Deep blue qubits gather the X error syndrome of the surrounding four data qubits and will be measured. Pale blue qubits gather the Z error syndrome of the surrounding four qubits, and will be measured. Each data qubit is normally stabilized by two Z-stabilizers and two X-stabilizers.

2.5.3 Planar code

Logical qubit encoding

The planar code encodes a logical qubit on a lattice. The complete lattice has no degrees of freedom; a lattice with N physical qubits has N stabilizers, and hence is a fully-specified quantum state. Figure 2.6 shows a planar code whose code distance is 4.

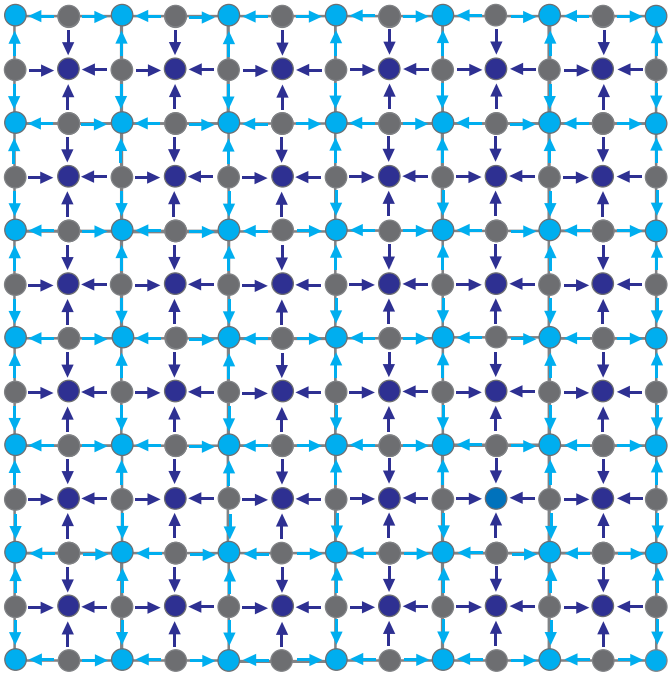

Figure 2.5: The units for error correction. Gray qubits are data qubits. Deep blue qubits are syndrome qubits used to find X errors. Pale blue qubits are syndrome qubits used to find Z errors.

Each data qubit on the smooth boundaries is stabilized by only one Z-stabilizer. This gives the lattice a degree of freedom for a logical bit. Each data qubit on the rough boundaries is stabilized by only one X-stabilizer. This makes a degree of freedom for a logical phase.

Measurement and robustness

To measure the logical qubit all of data qubits are measured. As long as each stabilizer is stabilized, the parity of measured value of data qubits on any chain between the two rough boundaries will be the same. This makes the robustness against measurement errors. Additionally, the chain produces the robustness against physical state errors. For example, closed X error cycles do not affect the logical qubit state, shown in Figure 2.7. Any X error cycle pass across any Z operators even number of times. This results in an identity operator to the lattice.

Logical gate and error

Any chain of bit-flipped data qubits between the two smooth boundaries results in a logical X operator, shown in figure 2.10. This may be done intentionally to execute the

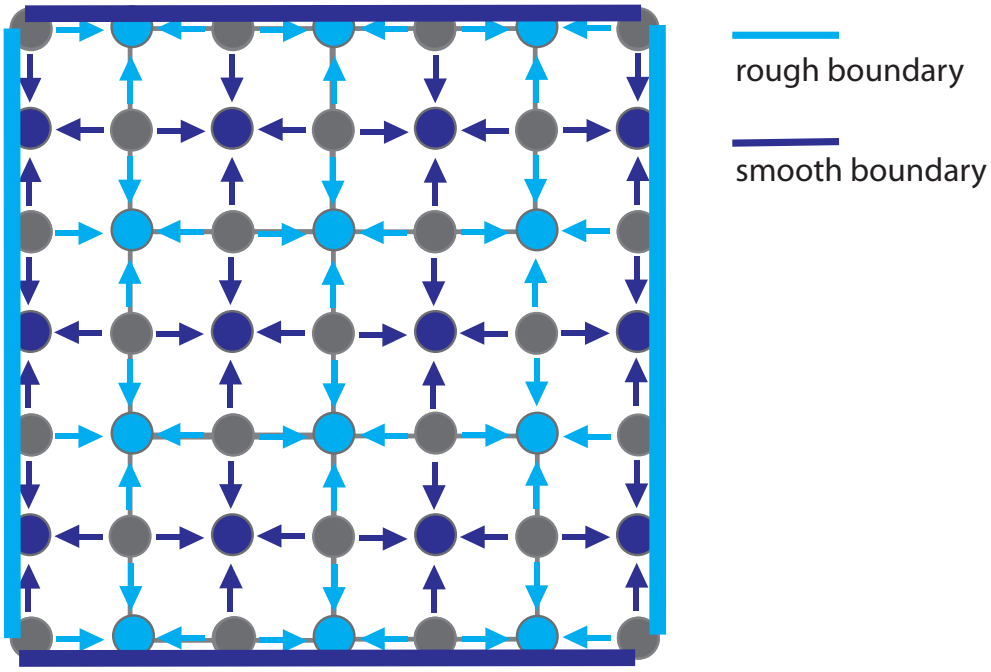

Figure 2.6: A lattice of the planar code. The degree of freedom is at both types of boundaries.

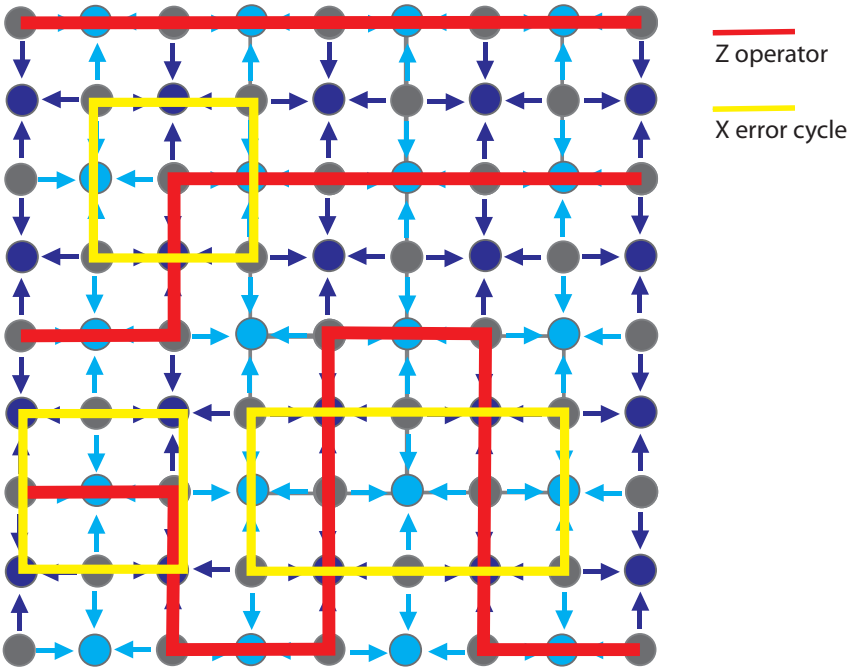

Figure 2.7: Examples of closed error cycle.

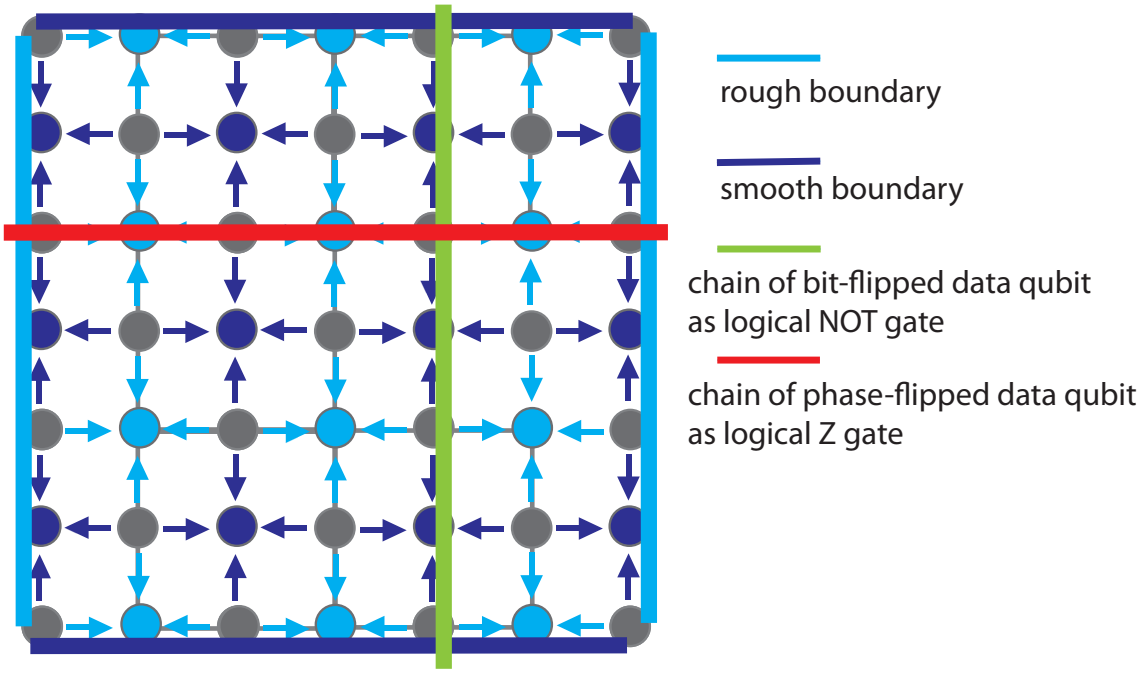

Figure 2.8: Logical operations of the planar code

logical gate, but if a chain of bit flip errors occurs undetected, a logical error to the state is caused.

Similarly, any chain of phase-flipped data qubits between the two rough boundaries results in a logical Z operation.

2.5.4 Surface code

Logical qubit encoding

The surface code encodes arbitary number of logical qubits on a lattice. A logical qubit is encoded as two *defect*s of stabilizer operations on the lattice as shown in Figure 2.9. A defect which consists of unstabilized faces has a degree of freedom expressed in the parity of a loop of physical qubits on the boundary of the defect, or any loop of qubits that encircles one of the defects.

Measurement and robustness

Same as planar code shown in subsubsection , to measure the logical qubit all of data qubits are measured. As long as each stabilizer is stabilized, the parity of measured value of data qubits on any chain between the two defects will be the same. This also makes the robustness against measurement errors. Additionally, the chain produces the robustness

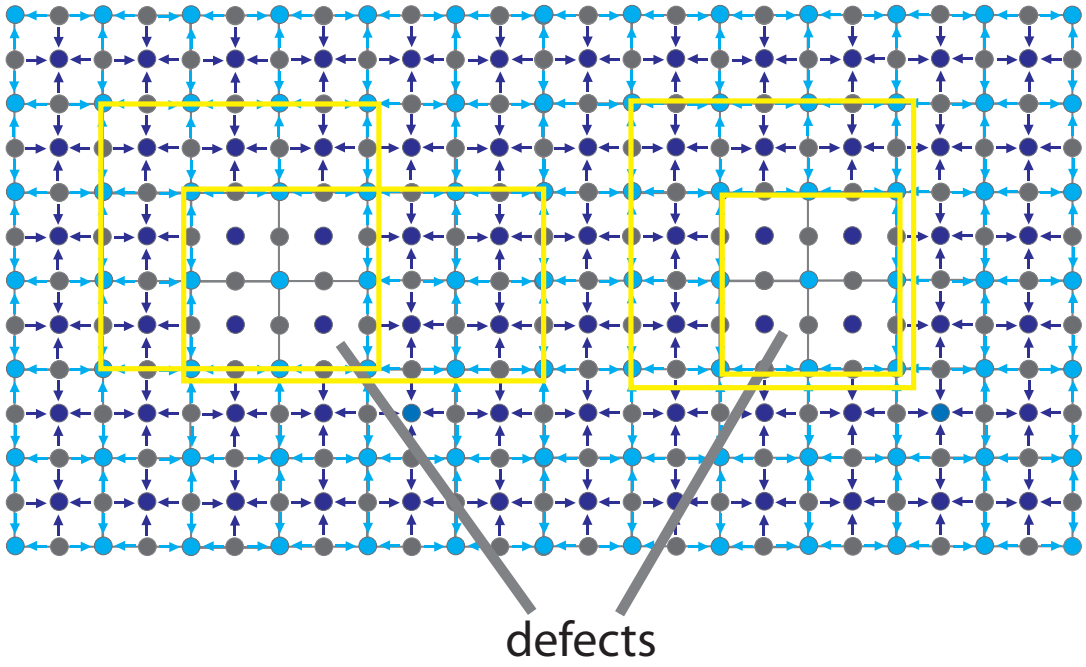

Figure 2.9: A lattice in which a logical qubit is encoded: there are two defects where the stabilizers are not measured as shown by the missing arrows. The degree of freedom on the boundary of the defects determines the state of the logical qubit and the qubits on each of the four yellow rectangles has the same parity (when the state is not in error). Many possible rectangles circle each defect, which provides the surface code's robustness against measurement error.

against physical state errors. Same as planar code, any X error cycle pass across any Z operators even number of times. This results in an identity operator to the lattice.

Logical gate and error

Any chain of bit-flipped data qubits between the two defects results in a logical NOT operation, shown in figure 2.10. This may be done intentionally to execute the logical gate, but if a chain of bit flip errors occurs undetected, a logical error to the state is caused.

Logical error

A change in the error syndrome of a stabilizer indicates that the plaquette (or vertex) includes the termination of an error chain. In the normal case, an isolated bit (or phase) flip, two neighboring plaquettes (vertices) will both show -1 eigenstates, and the error is easily isolated. Multiple errors in a neighborhood can form a longer chain, resulting in a

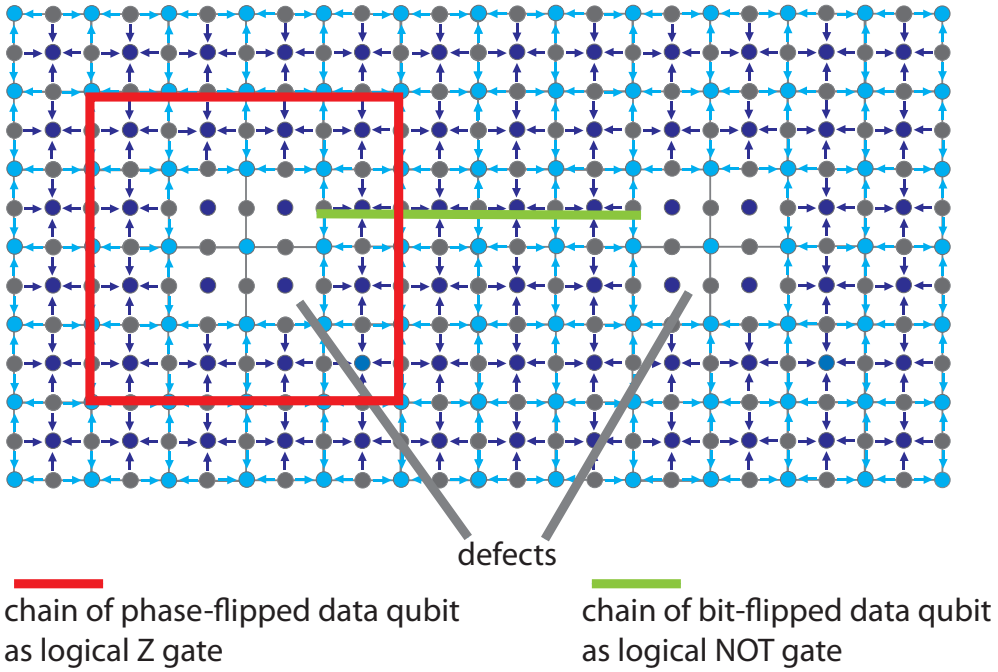

Figure 2.10: Logical gates for a logical qubit of the surface code. Executing single-qubit physical operations (generally, a bit flip or phase flip) on all of the data qubits along the chain on the green or red path between the two defects will be a logical operation. By definition, a logical error is an unintended logical operation. Logical errors occur all of the change on a path connecting two defects or the boundary of the lattice. We can see the concept of code distance here. The longer the distances between the defects and boundary are, the more fault-tolerant the computer is. This is why surface code is robust against memory error and operation error.

more difficult error analysis problem. If the error chain is connected to the holes of the boundary of the lattice, the termination will be hidden. So, an error chain between holes or the boundary will be a logical error.

2.5.5 Error correction

To connect the detected error terminations with same flip operation to the error will work as error correction, because it fix the states of each stabilizers. There is an algorithm called Blossom V[14] to determine the location of qubits to flip. Like the code distance in classical error correction, the distances between the same type of boundaries in the planar code and between the two defects in the surface code are the code distances.

Chapter 3

Problem Analysis

3.1 Problem definition

The problem I focus on is the handling of "faulty devices" on a quantum computation chip. A faulty device is a device which is fabricated imperfectly and cannot hold a quantum to use as a qubit. Devices like this exists on classical computation chips, and the probability that correctly working device is fabricated is called the "yield". In classical systems, chips on which faulty devices affect the operation are often discarded. The yield is so high that the solution to discard works efficiently in the current economic environment. But in quantum computation, the yield is low today and it is said that it will not achieve the level we can discard chips which involves faulty devices, both in research and commercial situation. Thus, we have to deal with faulty devices. In surface code with faulty devices, we will have flaws in the 2-D lattice.

Figure 3.1 shows an example of faulty data qubits. It will break the unit of error correction of the surface code and we need a method to deal with such flaws. Even if we can deal with them, additionally, the threshold of the surface code will be negatively affected. Note that the robustness of the surface code completely depends on the code distance, the number of data qubits between the two defects or between the defect and the boundary of the lattice. We need to calculate how much the threshold will decrease as a result of the lack of qubits.

Figure 3.2 depicts the influence of faulty syndrome qubits. The units which have a faulty syndrome qubit cannot gather their error syndrome onto the central syndrome qubit. They need another approach to measure the error syndrome.

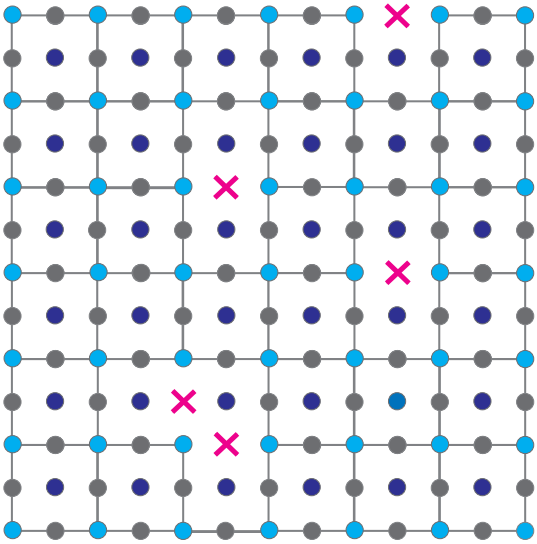

Figure 3.1: Defective lattice example. Dots describes qubits. Pink crosses mark the faulty device. In this figure, all faulty devices are data qubits. The lines describe the units of error correction and it is seen that faulty data qubits break the units of error correction.

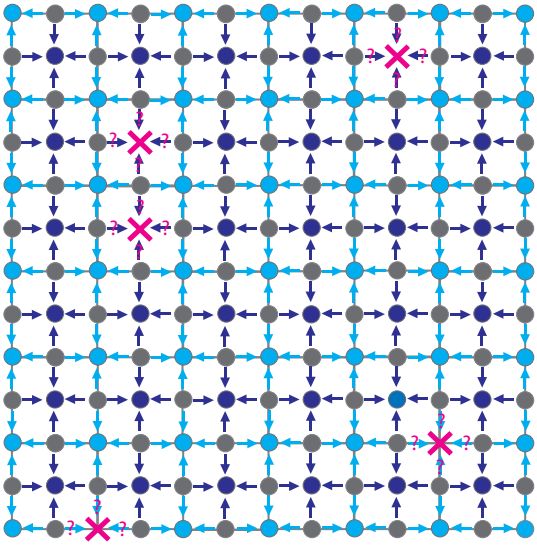

Figure 3.2: Defective lattice example with faulty syndrome qubits. In this figure, all faulty devices are syndrome qubits. It is shown that the units which include faulty syndrome qubits cannot measure the error syndrome.

3.2 Related work

Topological quantum computation, the dimensionally generalized name of the surface code originally became known as Kitaev's toric code[15]. There were multiple qubits

encoding and logical operators. The lattice was on the torus and had no boundaries. It was completely theoretical stage. Boundaries of the lattice were suggested by Bravyi et al.[16]. Logical operators got some more feasible with the two types of boundaries: smooth boundary and rough boundary. Raussendorf et al suggested that the topological quantum error correction can be used for one-way quantum computation[7]. The threshold of topological code is calculated as 0.75%. This work encouraged the research of topological quantum computation. In this paper, the surface code is summarized and explained easily to understand by Fowler et al.[8]. Assuming all errors on operation, memory error, measurement error and gate error, not assuming loss, the threshold of the surface code is calculated as 0.6%. Fowler et al. calculated the threshold of the physical error rate as 0.9%, using improved matching algorithm for the error correction[17]. My work fill the gap between this work and the realistic situation of hard fault. In my dynamic analysis, I use the library which is used in this paper. [1]

Just by restocking the place of a lost qubit with another quantum and process error correction, loss is tolerable and the tolerability against loss is described with the error rate (E_{err}) and the loss rate (E_{loss}) as

$$E_{err} + E_{loss} < 0.7\%. \tag{3.1}$$

As far as *heralded loss*, the loss which can be found on ahead, Stace et al. discussed that the threshold of qubit loss is

$$E_{loss} < 50\% \tag{3.2}$$

with the percolation theory[12][18]. Their assumptions with problems are:

1. The loss can be detected (*heralded loss*).
2. Lost qubit can be fixed by restoring another new quantum at the place and processing error correction.
3. They have perfect error detection and correction, errors don't occur during stabilization or error correction.

Their assumption (1) is practically possible in 3-D topological code using photons, but is not possible in solid system. The assumption (2) has problem both in photonic system and in solid system. In photonic system, photon generators can be faulty. There are two types of loss – dynamic loss and static loss. As long as the devices which loose qubits still have the ability to hold a quantum, the dynamic loss can be fixed by restoring a new quantum in both systems. On the other hand, the static loss can not be fixed because

[1] I am deeply indebted to Austin Fowler, both for the loan of the code and unflagging dedication in helping me debug adaptations of the code to faulty lattice.

Table 3.1: The characteristics of the different types of loss

	Dynamic loss	Static loss
3-D photonic system	Detectable Fixable	Detectable Unfixable
2-D solid system	Undetectable Fixable	Detectable Unfixable

the devices do essentially not have the ability to hold a quantum. The assumption (3) is completely wrong. The stabilization and error correction are also sets of operations, we cannot avoid errors during them. This assumption makes their analysis which results in equation 3.2 unreasonable. Errors during stabilization causes mis-correction and errors during error correction leave errors still or generate new errors, and they also can be the big source of logical errors. Anymore, no errors occur during restoring new quantum so that E_{err} and E_{loss} are treated in the same way and the left value of the equation is just the sum of them. Lindner et al. showed cluster state generation by a device they called the photon machine gun[10]. Their work aims 3-D topological model and if a device is faulty, there is a loss. Jones et al. proposed an architecture for scalable quantum computation with quantum dots[9]. Their self-assembled quantum dots holds a electron to use it as a qubit. If a dot is faulty, there is a lost qubit. Devitt et al. summarized the requirements of topological quantum computing system. They mentioned about the loss, but they just assumed that loss are tolerable by referring Stace's work[12].

Previous work does not have completely feasible assumptions and we need a new set of assumptions to analyze precisely the influence of losses to design practically feasible system. We need physically possible assumptions to research more accurately. At first, state errors must occur anytime during the computation, including during stabilization. The second is loss – dynamic loss and static loss. Table 3.1 summarizes the detectability of the two types of loss. In photonic system, dynamic loss can be detected and fixed, and static loss can be detected but cannot be fixed. In solid system, dynamic loss cannot be detected but can be fixed, and. static loss can be detected but cannot be fixed. These are the practically required assumptions and at last we need full of these assumptions. In this thesis, I focus on the static loss. My assumptions are:

1. The dynamic loss does not occur.

2. The placement of faulty devices is known (the static loss can be detected in both system by testing the fabricated computer).

3. Memory error, measurement error and gate error occur any time, and with the same

probability (errors may occur during stabilization).

Chapter 4

The logic of surface code on a defective lattice

In this chapter, I describe the theory of treating broken stabilizers due to faulty devices as a superstabilizer from the time the program starts until it ends. Z stabilizer and X stabilizer are symmetric, so we describe Z errors in Z stabilizer especially.

A faulty qubit results in one of two possible types of changes. In either case, the potential problems are the validitiy of the stabilizer circuits and logical measurements and the detectability of physical state errors. Figure 4.1(a) shows the first type of change, a superplaquette which consists of two plaquettes, connected due to a defective lattice qubit (which I call a 2-plaquettes superplaquette). In this case, the shape of the lattice is changed, so we need to take into consideration unusual error chains and measurement loops different from the usual rectangles. The stabilizer circuit itself is also changed, so the range of error propagation widens. Additionally we need to pay attention to the influence on neighboring stabilizers. We have to implement operations in a manner that does not break the stabilizer operations of neighboring plaquettes. Figure 4.1(b) shows the second tyep of change caused by faulty qubits. It describes the physical description of a plaquette whose syndrome qubit is defective (donut plaquette). This defective qubit changes the stabilizer circuit, but does not change the shape of the stabilizer unit. So we don't need to mind special error chain and measurement: for donut plaquette, we only need to take care the stabilizer circuit and error propagation.

4.1 Stabilizer circuits in special stabilizers

First, we show the stabilizer circuit in a superplaquette. Figure 4.2(a) shows the stabilizer circuit for a 2-plaquette superplaquette. The indexes of qubits in Figure 4.2(a) and Figure 4.1 (a) correspond and Figure 4.2(b) and Figure 4.1 correspond. We can

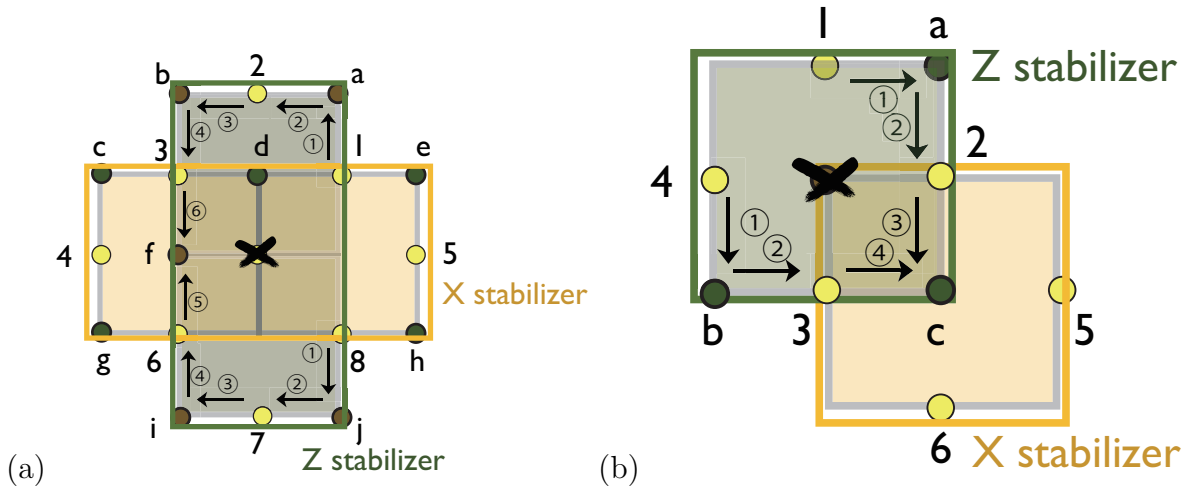

Figure 4.1: Yellow dots are data qubits and gray dots are syndrome qubits. (a) The physical description of the 2-plaquettes superplaquette. The indexes of the qubits correspond the indexes of qubits in Figure 4.2. The number of arrows are the order of syndrome-gathering operation (the order for reverse operation is omitted for visiblity in this figure: it is just reverse order operation of gathering operation). Only nearest-neighbor operation is used. (b) The physical description of a plaquette whose syndrome qubit is defective. The path to gather the error syndrome is changed but the size of the stabilizer unit is not changed.

expect that the number of circuit steps depends on the distance between the measured qubit and the farthest qubit.

See Figure 4.3. The circuit in Figure 4.3(a) shows an example of error propagation in a superplaquette. The red box labeled "Z" is the original Z error. In this example, the Z error propagates during calculation of a Z stabilizer. A Z error propagates at a CNOT operation from the target qubit to the control qubit. The blue box labeled "Z" is the progress of error propagation. The line labeled with blue box indicates that the qubit the line describes has a Z error at the timing by error propagation from the original error. In Figure 4.3(a), an error occurs at $|\ q_3\rangle$ during the stabilizer operation. At the end of Z stabilizer operation, four qubits will have Z error by the propagation from $|\ q_3\rangle$. Actually, one of them is syndrome qubit, not lattice qubit so three lattice qubit will have Z errors. This phenomenon can result in the circuit in Figure 4.3(a) being equivalent to the one in 4.3(b). It describes a circuit in which Z errors occurs after Z stabilizer operation, before operating X stabilizer (showing only the original errors: red box). Clearly these errors can be detected by stabilizer operation of surface code: they can be handled by normal operation of the X stabilizer normally. Thus they don't misdirect the measurement of error syndrome.

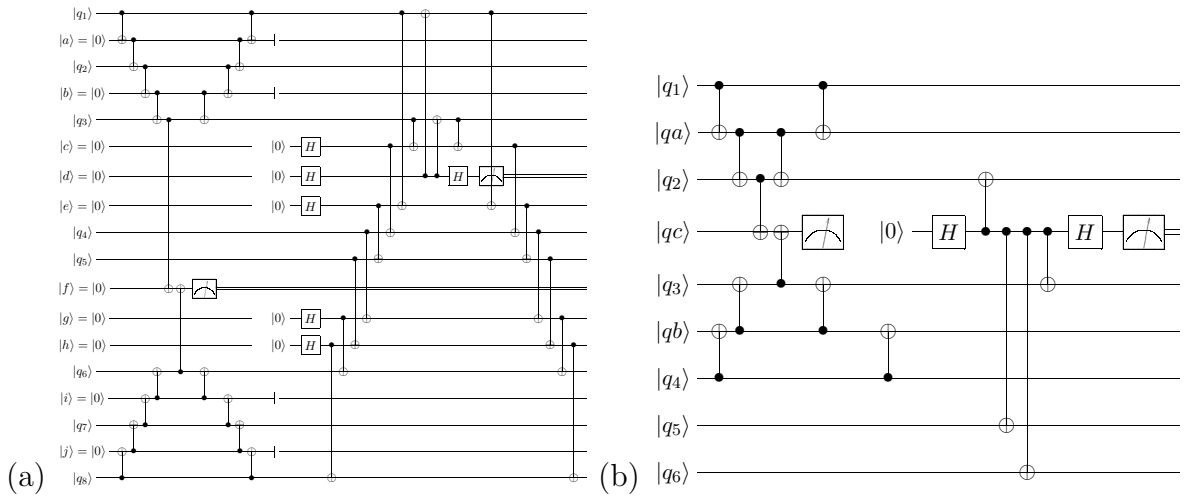

Figure 4.2: Circuits for stabilizers in Figure 4.1. (a) Stabilizer circuit for a 2-plaquette superplaquette, in Figure 4.1(a). The error syndrome is gathered in order and after gathering the error syndrome the state of each qubit will be fixed by reverse operation. The shape of this circuit resembles a ripple so we call this kind of stabilizer circuits and the method ripple syndrome-gathering circuit (RSGC) and ripple syndrome-gathering (RSG). (b) The stabilizer circuit for the donut plaquettes, in Figure 4.1(b). It gathers the error syndrome to the syndrome qubit on a vertex which is not used generally in a Z stabilizer, and measure the error syndrome on the vertex qubit. This circuit corrects the state of each qubit.

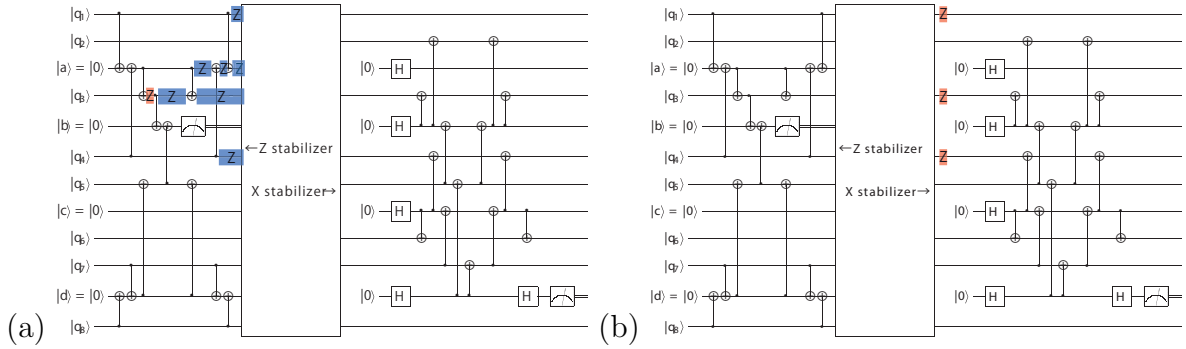

Figure 4.3: The description how the propagated errors are corrected in RSGC. The white box between Z stabilizer and X stabilizer represents nothing, just to separate the stabilizers clearly. (a) The red "Z" is the original error. It propagates along two qubit operations and at the last of Z stabilizer, three data qubits and a syndrome qubit are errored. (b) The red "Z" are the errors remaining from the previous Z stabilizer. They will be stabilized in this X stabilizer.

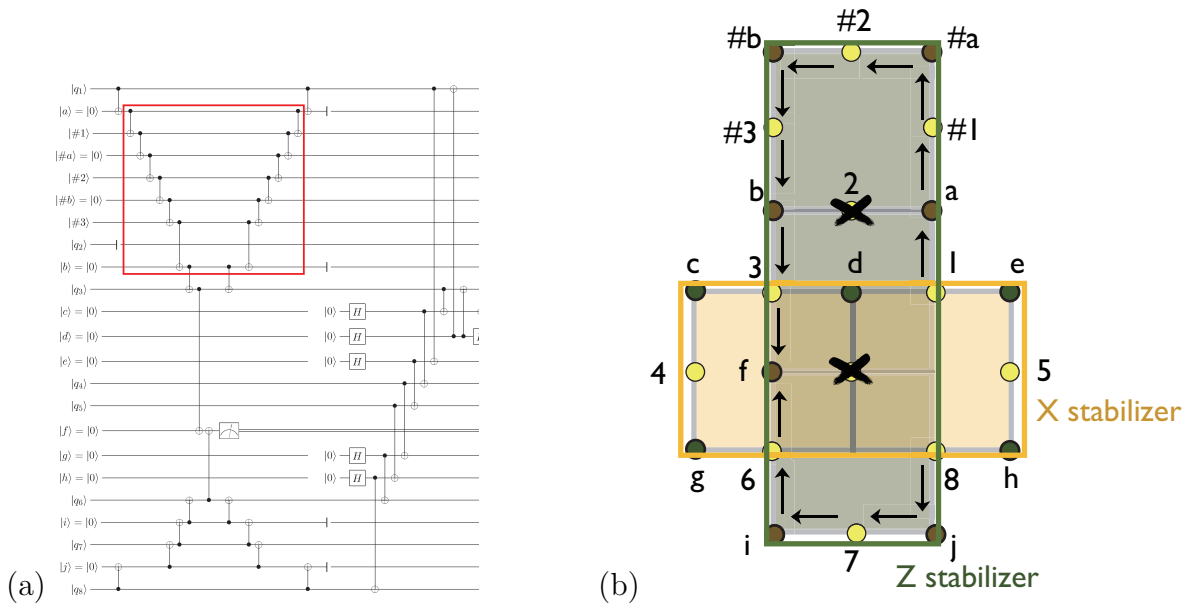

Figure 4.4: (a) Extended stabilizer circuit of Figure 4.2(a). $|\ \#2\rangle$ is the new hard fault and $|\ \#1\rangle$, $|\ \#a\rangle$, $|\ \#2\rangle$, $|\ \#b\rangle$, $|\ \#3\rangle$ are the added qubits. (b) The superplaquette description of (a).

.

So far we have shown the validity of a 2-plaquette connected superplaquette. Definitely we have to check whether or not larger superplaquettes can gather error syndrome correctly. See Figure 4.4. Figure 4.4(a) is an extended circuit of Figure 4.2(a). The superplaquette which corresponds to the Figure 4.4(a) is described in Figure 4.4(b). $|\ 2\rangle$ is the hard fault making the superplaquette larger forcing the addition of new qubits, marked with "#". The extended part of the circuit corresponds to the extended part of the superplaquette. We can see that the size of the circuit can be extended simply, only the ripple will be linearly longer. This shows that the error syndrome of any superplaquette can be gathered at the expense of a larger circuit.

4.2 The relationship between errors and measurement

Figure 4.5 shows the simplest error cycles which can be made around superplaquettes. They shape the smallest square we can see. Clearly, the simplest error cycles in complete lattice occur among four plaquettes. In a defective lattice, some of the four plaquettes will be a superplaquette or several superplaquettes. The situations in Figure 4.6 do not result in any problems. Each measurement line (red line) passes over the flipped qubit

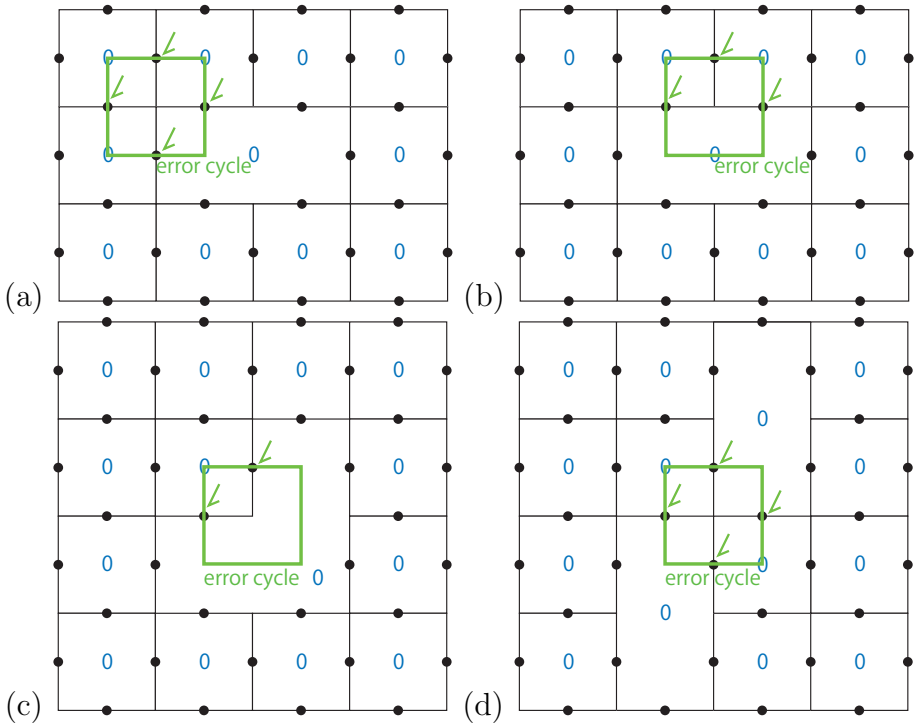

Figure 4.5: (a) One of the plaquettes is a superplaquette. (b) Two of the plaquettes are superplaquettes neighboring. (c) Three of the plaquettes are superplaquettes. (d) This is the pattern that two of the plaquettes are superplaquettes which are on diagonal position.

twice, so the parity on it will not be changed. This logic is the same as the logic in the complete lattice. Figure 4.5(b), (c) and (d) also do not have any problems. However, we must also consider bigger error cycles, for example, Figure 4.8. Actually, Figure 4.8(a) can be treated as a connected error cycle combining Figure 4.5(a) and Figure 4.5(b). There is a measurement line on the error cycle in figure 4.8(a). Definitely, there should be many other measurement line and we have to check the relationship between all such measurement lines and the error cycle. See Figure 4.9(b). There are two simple error cycles, the one in Figure 4.5(a) and Figure 4.5(b) and a measurement line on both of the error cycles. Note that an error line is a state flip (the type of flip whether bit-flip or phase flip depends on the error cycle type) line essentially. Thus, a not-errored qubit can be considered as a qubit flipped an even number times and the set of error cycles in Figure 4.9(b) is essentially same as the one in Figure 4.9(a). The measurement line passes across the error line four times and the number depends on the simple error cycles whom the measurement line itself is on. We already know the relationship between measurement lines and all simple error cycles and the number that each measurement line passes across the error line twice. In this way, the number of simple error cycles of whom a bigger error cycle consist can be extended linearly and however big the error cycle

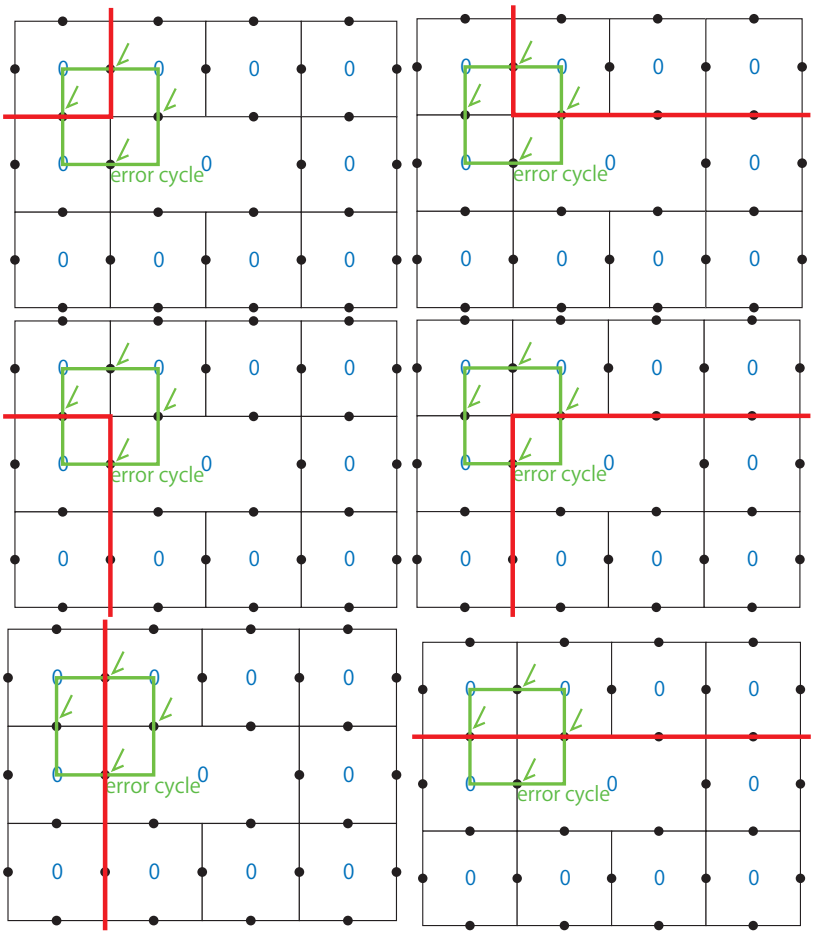

Figure 4.6: Measurements lines (red lines) on an error cycle described in Figure 4.5(a). Obviously, each measurement line passes across the error cycle twice. This fact means that this type of positional relationship between an error cycle and a superplaquette does not cause any problems.

is, each measurement line passes over the flipped qubit even number times. Thus, any error cycles around superplaquettes won't have any problems in measurement. Another example is described in Figure 4.10 which is for the error cycle described in figure 4.8(b). The measurement line passes across the error line even number times apparently.

4.3 Error detection

In this section, I examine about the validity of the superplaquette from the point of view of error detection. The requirements for superplaquette in error detection are:

- The superplaquette must have a stabilizer state which corresponds in error detection to the stabilizer states of plaquettes which should normally, originally exist.

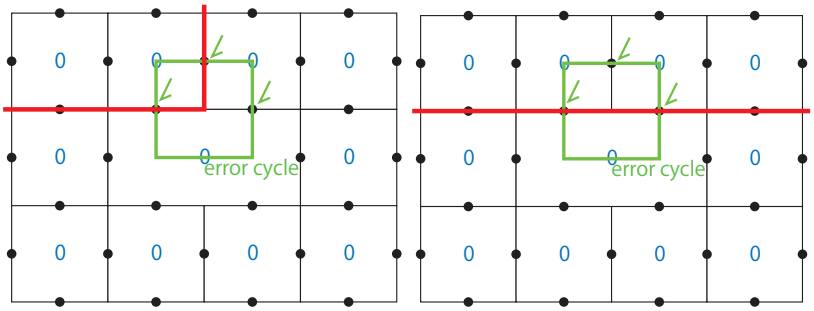

Figure 4.7: Six possible cases for the intersection of a line of measurements with a superplaquette. Measurement lines (red lines) on an error cycle described in Figure 4.5(b). Each measurement line passes across the error cycle twice. This fact means that this type of positional relationship between an error cycle and a superplaquette also does not result in any problems for correct operation of the surface code.

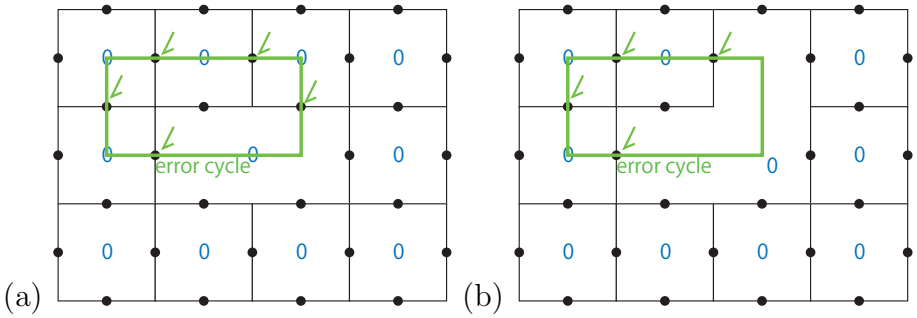

Figure 4.8: Examples of possible bigger error cycles.

- The stabilizer of the superplaquette must change with flips of an odd number of qubits which belong to the superplaquette.

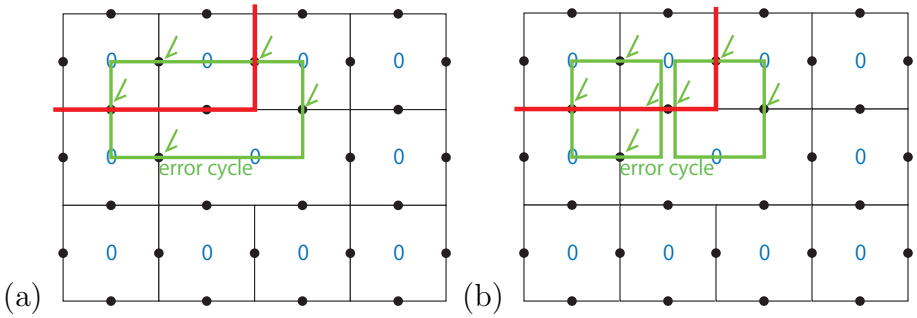

Figure 4.9: (a) A practically possible error cycle. The measurement line passes across the error line even number times, twice. (b) One way to think of the error cycle in (a). The measurement line passes across the error line an even number of times, four times.

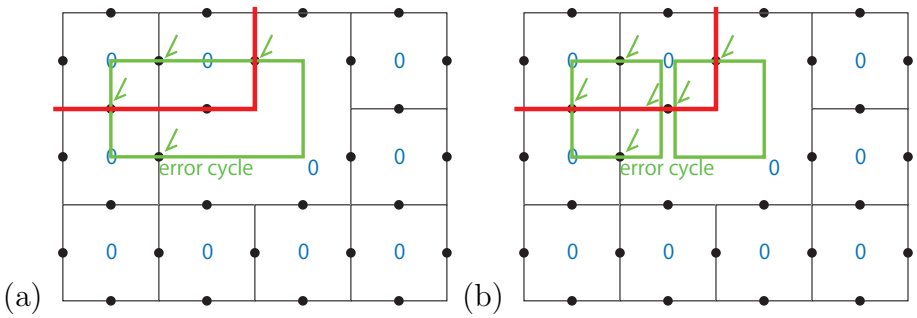

Figure 4.10: (a) Actually exsitable error cycle. The measurement line passes across the error line even number times, twice. (b) One way to think of the error cycle in (a). The measurement line passes across the error line even number times, four times.

At first, we describe the case of a 2-plaquette connected superplaquette. So we convert the two normally plaquettes into a superplaquette, satisfying the requirements above.

Note that the stabilizer state of normal plaquettes should be in +1 and if it is in -1, we take that there is an error chain termination. Note also that four physical qubits belong to a plaquette.

The stabilizer states of the normal plaquettes should be +1. If we measure the stabilizer state of only three qubits of the four qubits in a plaquette, it should be +1 or -1. When it is +1 the other qubit should be in +1 if it is measured with Z operater and when it is -1, it should be in -1. The situation of superplaquette is similar to this. If we measured stabilizer states of two neighboring plaquettes excluding the shared qubit (stabilizers of three qubits each), we would see "+1 +1", "-1 -1", "+1 -1" (blue lines in Figure 4.11). In case of "+1 +1" and "-1 -1", the stabilizer state of the superplaquette will be in +1 and in case of "+1 -1", it will be in -1. Here we have to consider the state of the shared qubit. Actually, if the lattice is a complete lattice, it affects both stabilizers in the same way. When it flips one of the stabilizer states, it flips the other, too. The cases of "+1 +1" and "-1 -1" are valid because the stabilizer states of both plaquettes can be in +1 only with controlling the shared qubit, without changing any other plaquettes. In contrast, the case of "+1 -1" is invalid because the shared qubit cannot make both stabilizers in +1 simultaneously without any changes on other plaquettes. So being "+1 +1" or "-1 -1" is the restriction for the six qubits which are the alive qubits if the two plaquettes make up a 2-plaquette connected superplaquette. That the stabilizer states of the two set of three qubits which make up a superplaquette are "+1 +1" or "-1 -1" implies that the stabilizer state of the superplaquette should be +1. In other words, if it is +1, the first requirement is satisfied.

Now we have to satisfy the second requirement. It is clearly satisfied because the modified stabilizer is also a stabilizer.

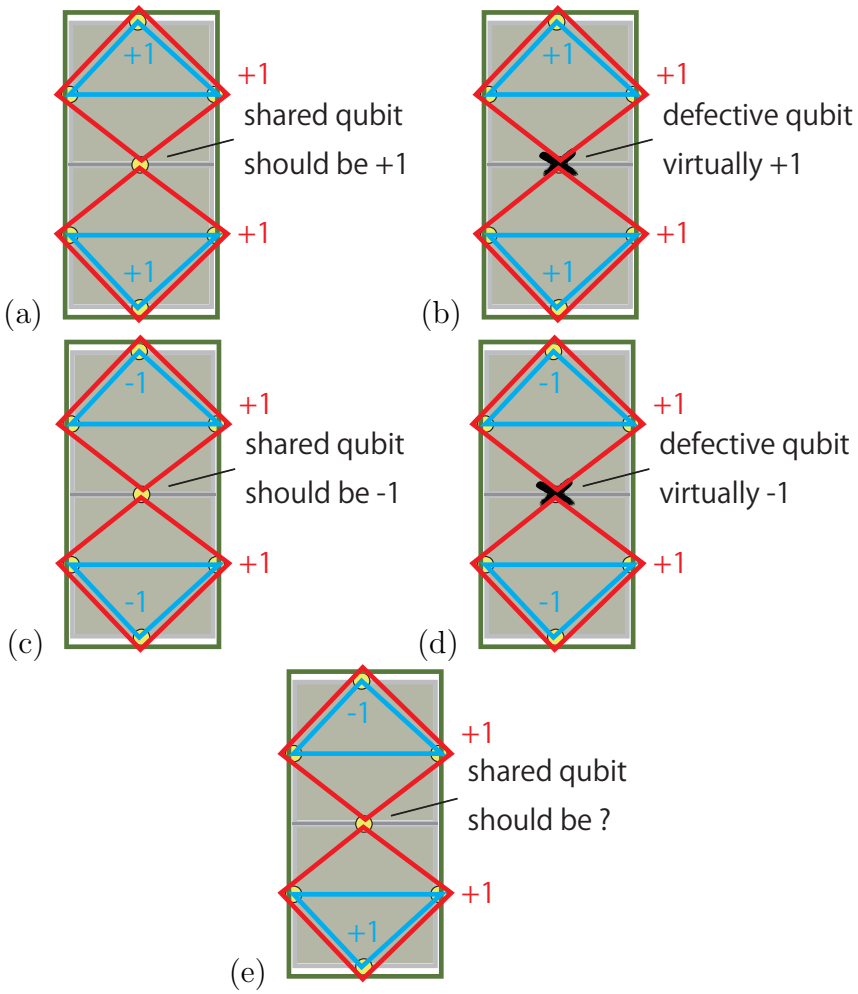

Figure 4.11: (a) The relationship of two neighboring plaquettes. The two three-qubit-stabilizers (blue lines) should be in "+1 +1" or "-1 -1" to make both four-qubit-stabilizers (red lines) be in +1. (b) Two neighboring plaquettes when the shared qubit is defective. Not to change any states of other plaquettes, the two three-qubit-stabilizers (blue lines) should be same as (a). (c)(d) In case of "-1 -1" of (a) and (b). (e) The case that three-qubit-stabilizers are each in +1 and -1. In this case, the shared cannot have valid state to make the both four-qubit-stabilizers be in +1 simultaneously.

So we should have the stabilizer state of the superplaquette be in +1. Then the superplaquette corresponds to two normal plaquettes in error detection.

Next, we have to consider larger superplaquettes. There are three types of plaquettes which have faulty devices: one which has a faulty device, one which has two faulty devices and one which has three faulty devices (Figure 4.12). Figure 4.12(a) has three stabilizers (red lines) in +1. If we make a new stabilizer of the stabilizers (shared qubits are counted twice), the new stabilizer will be in +1, and, if we remove the shared qubits from it,

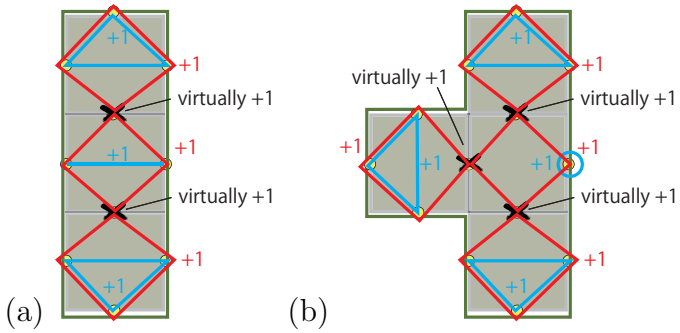

Figure 4.12: (a) One-defective plaquette and two-defective plaquette. (b) Three-defective plaquette.

the new stabilizer should be still in +1 because the removed qubits are counted twice (flipping twice does not change the stabilizer and not fliping twice does not change it too). Obviously this can be applied to each defective qubits in a plaquette, regardless of the nuber of the defective qubits.

We have a question, what does it imply that the new stabilizer is measured as in -1? It implies that odd numbers of stabilizers of original plaquettes are in -1. Of course it must be fixed into +1. So the stabilizer states of superplaquettes should be in +1 as other normal plaquettes and superplaquettes can detect physical errors as normal plaquettes.

Chapter 5

Analysis

In this section, I will describe the implementation for the two analyses. Both of them first simulate the creation of a physical system with lattice on which faulty devices are placed at random. After that, they have different processes.

5.1 Static simulation

We describe here the specification of the static simulator. It calculates the largest probability that the two defects or one of the defects and the boundary of the lattice are connected via a physical error chain. Data qubits can be faulty with a certain probability, known as the yield. Faulty syndrome qubits are not analyzed (so it calculates the effect of the superplaquettes, not of donut plaquettes). The probability the simulator outputs is the lattice refresh error rate. The lattice refresh error rate is affected by physical error rates like local gate error rate, measurement error rate and memory error rate.

5.1.1 Assumptions

This simulation assumes:

1. The placement of faulty devices is known (e.g.) can be determined after fabrication by measuring the device.

2. Only data qubits will be faulty.

3. Only X errors occur (Z errors can be ignored for the moment because they are assumed to be symmetric).

4. A logical qubit is encoded with two defects. Figure 5.1 shows an example of the lattice.

5. All physical error rates are 0.0001.

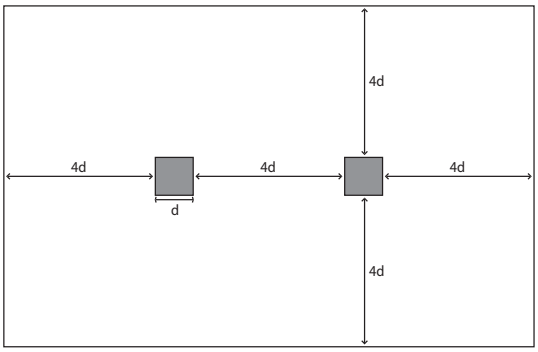

Figure 5.1: The shape of lattice for the simulator in which one logical qubit is encoded. The shaded squares are the holes and the distance between them, between the defects and the lattice boundary, and the length around the defect affect the logical error rate.

We assumed that all physical error rates are 0.0001 in each time step and call them all simply as the physical error rate. We may have error propagation within stabilizer operation. How much the propagation affects is defined by the stabilizer circuit. So we define a new probability called the adjusted error rate with the physical error rate and the KQ of the stabilizer circuit where the K is the depth of the circuit and Q is the number of the qubit. Each different superplaquette has its own adjusted error rate. Adjusted error rates apply the simulation to each error propagation of each superplaquettes. However, the number of superplaquette types is essentially infinite, in this simulation we limit superplaquettes which have a genuine adjusted error rate with the sizes of themselves: under-4-plaquette connected superplaquettes have the genuine adjusted error rate and we give larger superplaquettes same adjusted error rate. The static analysis seeks "the shortest error chain" between the defects or between a defect and the boundary. This is by that the logical error rate depends most on the shortest path.

5.1.2 Algorithm

What the static simulator does is:

1. Create a simulated defective lattice.

2. Compose of superplaquettes. A weight is given to each plaquette depending on the size / shape of the plaquette. The physical error rate of qubits in the plaquette is timed with the weight. It corresponds to error propagation around the stabilizer of the plaquette.

3. Seek the shortest error chain on the lattice.

4. Time the fixed physical error rates of qubits on the shortest error chain. The calculated number is treated as the lattice refresh error rate. This is similar to "Shortest Path First" algorithm, the difference is that the path cost does not pulsed but timed here.

We use Dijkstra's shortest path first algorithm cite in the simulation. There are numbers of nodes, some pairs of which are connected with a link which has a certain link cost. Dijkstra's algorithm searches for the path which has the smallest total link cost between two end nodes. In our simulation nodes are qubits, the cost of each link is the adjusted error rate and the end nodes are the boundary of the defects and that of the lattice. It calculates the largest probability that a defect and the other defect or the boundary of the lattice are completely connected by the physical error chain, using the input values for yield, a code distance and a physical error rate.

5.1.3 Implementation

For the assumption that only data qubits will be faulty and have errors, Figures 5.2 – 5.4 are the only important data structures for this simulation. Figure 5.2 shows the data structure for a data qubit. In this simulation, there is no data structure for a syndrome qubit. Syndrome qubits are incorporated in the plaquette structure.

Figure 5.4 shows the data structure for a normal plaquette. The *p_adj* is the adjusted error rate, which is treated as the physical error rate of data qubits which belong to the plaquette. This simulation handles the error chain of two defects of the surface code. *Multiplied_p_l* or *multiplied_p_r* holds the "distance value" from either the left defect of the right defect. The "distance value" is calculated by multiplying the *p_adj*s of data qubits on the path from either the defects. If the plaquette has faulty qubit and composes a superplaquette, the pointer is held in **superplaquette*.

Figure 5.4 shows the data structure for a superplaquette. When multiplying a `p_adj` and a multiplied_p, if the normal plaquette belongs to a superplaquette, the `p_adj` and the multiplied_p of the superplaquette is used instead of those of the normal plaquette.

Table 5.1 shows the shapes of plaquettes and their corresponding IDs. As described above, under-4-plaquette connected superplaquettes can be distinguished in this simulation. All superplaquettes larger than 5 are treated as belonging to the same class.

$$p\_adj = (1 - g\_p\_phy)^{KQ} \tag{5.1}$$

Equation 5.1 is the function to give a `p_adj` to each shape of plaquettes. The second argument of `p_adj_INIT` is KQ of each shape of plaquettes.

```
struct qubit_tag{
  u_int number;
  u_char faulty; // whether faulty or not. 1 is a defect, 0 is not.
  char boundary;

  //if defect, plaq1 and plaq2 are NULL.
  plaquette *plaq1;
  // plaquette this qubit belongs to.
  // The *LEFT* side plaquette, if it is on vertical line.
  // The *UP* side plaquette, if it is on horizontal line.
  plaquette *plaq2;
  // plaquette this qubit belongs to.
  // The *RIGHT* side plaquette, if it is on vertical line.
  // The *DOWN* side plaquette, if it is on horizontal line.
};
```

Figure 5.2: Data structure of a qubit

```
struct plaquette_tag{
  u_int number;
  superplaquette *superplaquette; // NULL means not superplaquette
  qubit_list *includes; // qubits it includes.
  int defect;
  double p_adj;
  double multipled_p_r;
  double multipled_p_l;
};
```

Figure 5.3: Data structure for a plaquette

5.1.4 Evaluation

In this section, the static analysis which describes the relationship between the code distance and the effective code distance is shown. The effective code distance is a rough measure comparing a defective lattice to a perfect one. If the effective code distance of

```
struct superplaquette_tag{
  plaquette_list *includes;
  double p_adj;
  double multiplied_p_r;
  double multiplied_p_l;
  int superplaquette_type;
};
```

Figure 5.4: Data structure for a superplaquette

Table 5.1: Table of the shapes of superplaquettes and its ID.

ID	shape	KQ
0	A normal plaquette	5*5
1	Not distinguished. Too large. (5-plaquette or larger)	20*20
2	A superplaquette, connecting 2 plaquette.	8*9
3	A superplaquette, connecting 3 plaquette straight.	13*12
4	A superplaquette, connecting 3 plaquette perpendicularly.	13*12
5	A superplaquette, connecting 4 plaquette straight.	17*12
6	A superplaquette, connecting 4 plaquette like "T".	15*12
7	A superplaquette, connecting 4 plaquette like "$_{\|}$"	17*14
8	A superplaquette, connecting 4 plaquette like box.	15*14
9	A superplaquette, connecting 4 plaquette like "L".	17*14

a defective lattice is e.g. 10, we expect it to be roughly as resistant to logical errors as a distance 10 perfect lattice.

Figure 5.5 shows logical error rate, lattice refresh error rate and effective code distance. This graph has both the lattice refresh error rate and effective code distance on the y-axis. The x-axis is the yield, which is the probability that functional qubit is provided. This graph has a range of yield from 0.52 to 1.00 and a range of code distance from 4 to 12, note that $thiscodedistancestands \times 4$ stands for the actual number of plaquettes between two defects as shown in figure 5.1.

We can see from the graph that the effective code distance at 0.90 for yield is under half of the actual code distance for any code distance. While the yield goes down 0.1, the effective code distance goes down by half. That is, for y=0.9, we need a code of twice the

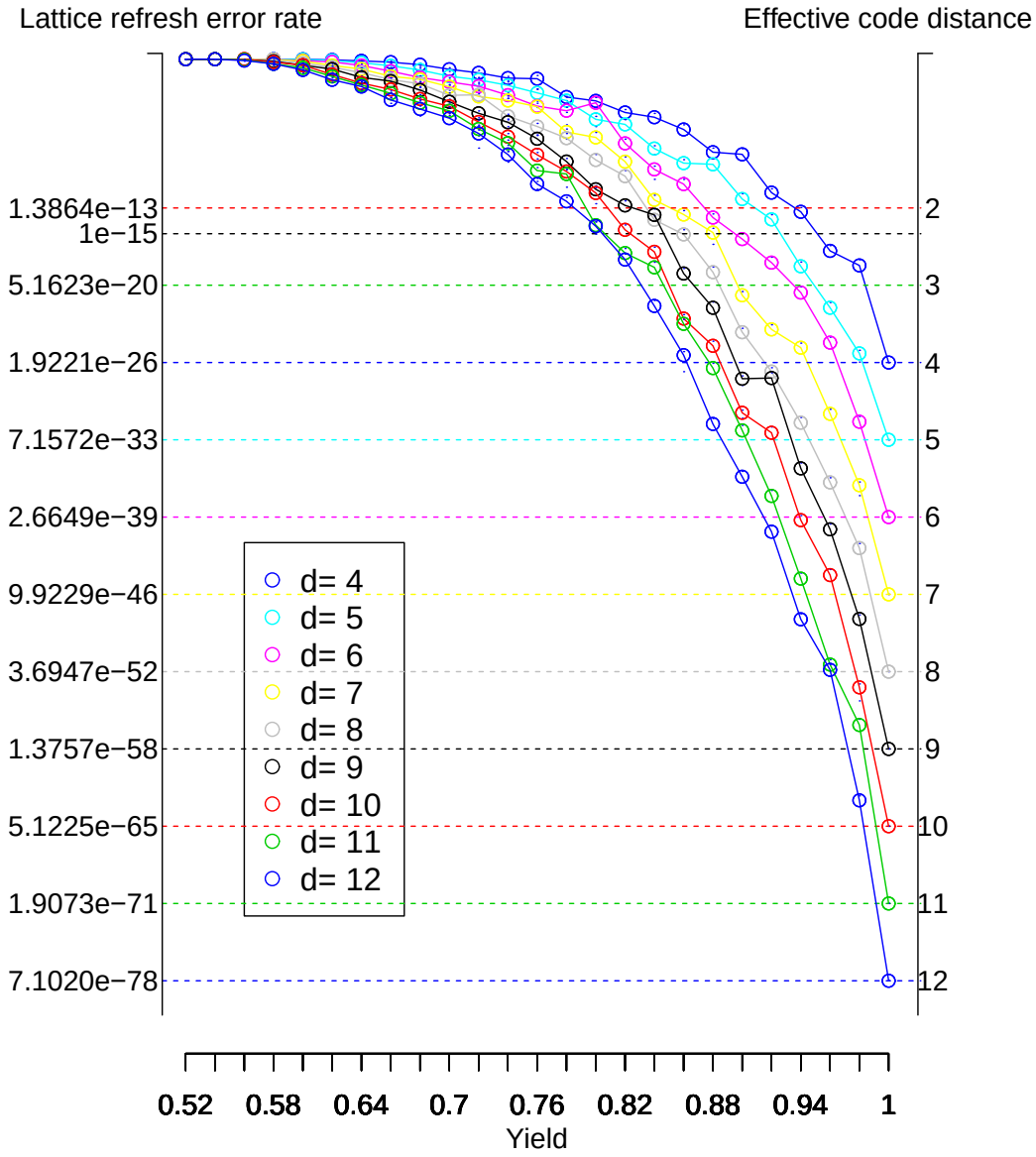

Figure 5.5: Static analysis of yield, effective code distance and lattice refresh error rate describing the average and the standard deviasion. The assumption of lattice shape is described in figure 5.1. Lattice refresh error rate is the probability that a logical error occurs while the whole lattice is stabilized once. The lattice whose yield is 1 is undefective lattice so its effective code distance and its code distance are equivalent.

distance (four times the area) to achieve the same error resistance. We need 10^{-15} lattice refresh error rate to successfully factor a 2048 bit number factoring. For example, if we have 0.80 for the yield and 0.001 for the physical error rate we need d=11. This means

that 44 plaquettes are required between the two defects and 30745 physical qubits are required for a logical qubit.

5.2 Dynamic simulation

The static analysis in the previous section gives only a rough indication. For more accurate analysis, dynamic analysis should be used. Dynamic simulation consists of two separate tools. The first one is the defective lattice circuit maker. This makes a error correction circuit for a simulated defective lattice. The second is the surface code error simulator, using the surface code error library called autotune, created by Fowler et al. This reads a circuit and, simulates the error occurrence and emulates error correction along the circuit.

5.2.1 Assumptions

Dynamic analysis assumes:

1. The placement of faulty devices is known.
2. All qubits may be faulty.
3. Both X errors and Z errors may occur.
4. All qubit state errors may happen; memory error, gate error and measurement error.
5. A logical qubit is encoded with the planar code.

5.2.2 Algorithm

Defective lattice circuit maker

1. Simulate a defective lattice.
2. Find the biggest isolated group of qubits which are connected along nearest-neighbor links.
3. Compose superstabilizers on the lattice.
4. Make the complete stabilizer operation circuit for each stabilizer.
5. Schedule the stabilizer operation circuits.
6. Output the circuit into a file.

Number 5 and 6 are repeated until the circuit reaches a certain time step.

The surface code error simulator

1. Read a circuit from a file.
2. Simulate the error occurrence along the gates in the circuit, using a classical stochastic approach (not a full simulation of the quantum state).
3. Emulate the error correction.

These are repeated until both logical Z errors or logical X errors occur a certain number of times. If the circuit file ends before enough logical errors accumulate, the simulator restarts the circuit from step 1.

5.2.3 Implementation

Defective lattice circuit maker

Figure 5.6 shows the data structure for qubits. The *faulty* flag is given at random and causes superstabilizers. The difference between *lattice* and *used* is; the biggest isolated group of qubits is lattice, but, the shape of the lattice may be arbitrary and the smallest size of stabilizer is fixed. Thus, some of qubits on the edge of the lattice may not be used.

Figure 5.7 shows the data structure for a stabilizer The *dataq* and the *synq* are linked lists for data qubits and syndrome qubits. The *extq* is tricky. There may be cases that the stabilizer is separated by faulty qubits in the stabilizer itself, though they are connected via outside of the stabilizer. Then the superstabilizer needs to use qubits outside of the superstabilizer, to carry a qubit on which the error syndromes are gathered. The extq holds the linked list of the path between the separated areas. When making a stabilizer circuit for the stabilizer, it first composes a tree; nodes are alive qubits and edges are connections between qubits. This tree is used to design the order to gather the error syndromes one by one. Figure 5.8 shows the ID of gate operations, to ship the circuit from the circuit maker to the error simulator.

5.2.4 Evaluation

In this section, the evaluation of the dynamic analysis is shown. Dynamic simulation simulates the behavior of the real-time classical control of the quantum computer as closely as possible, allowing as to calculate an accurate threshold, The simulations presented here are over 5,000 hours of CPU time on 2.27GHz Intel processors.

Figures 5.9 - 5.12 shows the result of the dynamic analysis. The threshold of physical error rate can be seen where the logical error rates of multiple code distances cross. It has already been shown that the threshold assuming that the yield is 1 and that the error

```
struct qubit{
      int num;
      int y;
      int x;
      u_char faulty;          // 1: faulty
      int group;              // to seek the largest isolated qubit group.
                              // Later, it'll indicate whether this qubit
                              // belong to the lattice or not.
      char lattice;           // qubits in the group and faulty qubits surrounded by
      u_char used;            // DATA: This qubit is used for actual surface code.
                              // SYN : The stabilizer this qubit is on the center of
                              // is used.
      u_char smooth_b_north; // 1: smooth boundary of the lattice
      u_char smooth_b_south; // 1: smooth boundary of the lattice
      u_char rough_b_east;   // 1: rough boundary of the lattice
      u_char rough_b_west;   // 1: rough boundary of the lattice

      u_char checked_SYNF;
      u_char checked_SYNS;
      int dijkstra_tmp;
}
```

Figure 5.6: Data structure of a qubit for the lattice circuit maker

correction circuit is optimized is 0.9%[17]. Figures 5.9 and 5.10 show that the logical error rate of longer code distances is higher where the physical error rate is about 0.6%. On the other hand, the logical error rate of longer code distance is smaller where the physical error rate is under 0.6%. Thus, when the physical error rate is under 0.6%, the error correction works in the desired fashion, suppressing rather than exacerbating errors. Thus, Figure 5.9 and 5.10 shows that the threshold is around 0.6%. This result matches prior work well. Raussendorf et al. at first estimated that the threshold with the same assumption was 0.75%[7]. One problem here is that the logical error suppression is smaller than that of previous work for low error rates. Under the threshold, the gradient of the line for code distance 9 should be more steep and the difference of the logical error rate of distance 9 and 5 should be bigger, as the physical error rate gets smaller[8].

One possible reason for this is that the error correction library has not been completely

```
struct stab{
      u_char type;                    // ZSTAB or XSTAB
      struct qubitlist *dataq;        // qubits, target of stabilizer
      struct qubitlist *synq;         // qubits, not target of stabilizer
      struct qubitlist *extq;

      struct ctreenode *tree;
      struct circuit *circ;
      u_char finished;

      u_char stabilized;             // 1: already stabilized in this phase
      u_char reach_hardmax;
      u_int last_stabilized_step;
      u_char boundary; // ZSTAB or ZSTAB or NORTH or SOUTH or EAST or WEST
      u_char id;
      int central_synq_y;
      int central_synq_x;

      struct stab *next;
};
```

Figure 5.7: Data structure of a stabilizer

turned to work with asynchronous stabilizer circuits. Thus, it worked ideally in the previous work, but does not ideally work in my work and some logical errors occur, slipping through the error correction. Since the threshold is found properly, we believe the results are valid, but are continuing to investigate possible sources of differences.

Figures 5.11 and 5.12 show graphs the physical error rate versus the logical Z or X error rate of the three code distances, 5, 7, and 9 for yield 0.9. On both graphs, the logical error rates of distance 7 and 9 cross between 0.01% and 0.1% physical error rates. But they don't cross the logical error rate of distance 5 yet. This indicates that the threshold may be smaller. Such a physical error rate is not practically feasible.

```
enum opcodes{
      /* no operation */
      NOP,
      /* initialization */
      OPIX,
      OPIZ,
      /* 1-qubit gate */
      OPX,
      OPZ,
      OPY,
      OPH,
      /* 2-qubits gate */
      OPCX,
      OPCZ,
      OPCY,
      OPSWAP,
      /* measurement */
      OPMZ,
      OPMX,
      OPMY
};
```

Figure 5.8: Data structure for gate operations to ship to error simulator

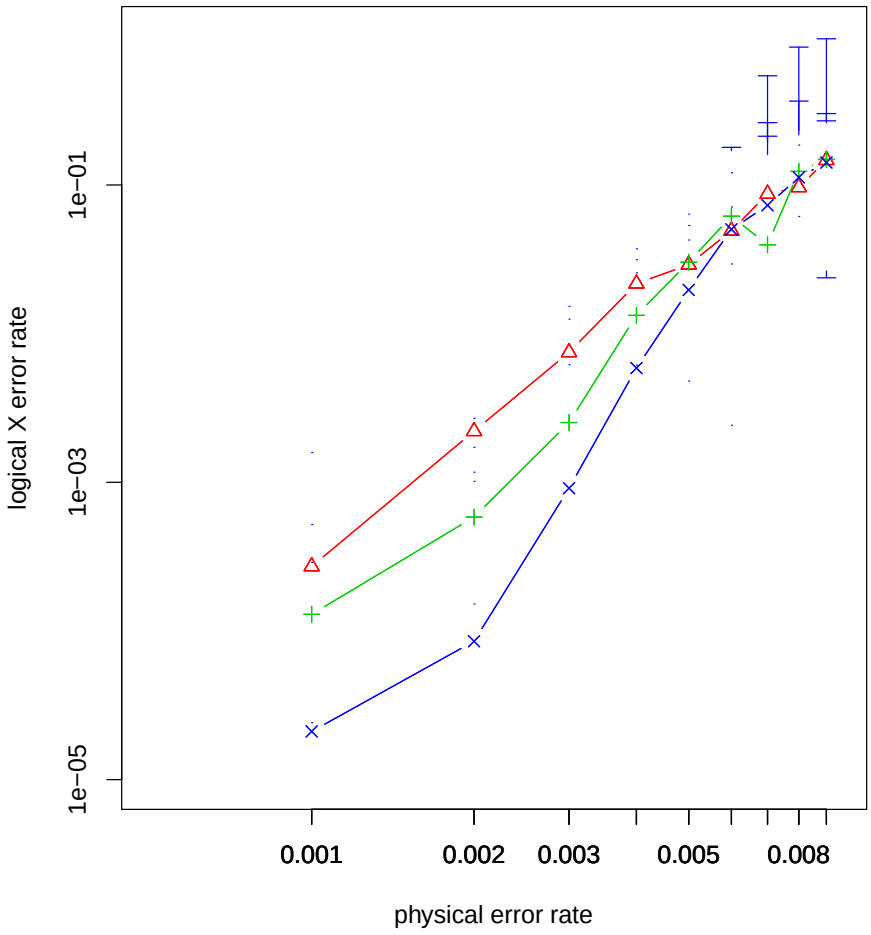

Figure 5.9: The graph of physical error rate versus logical X error rate in yield 1.0 Blue points are of distance 9, green points are of distance 7, red points are of distance 5.

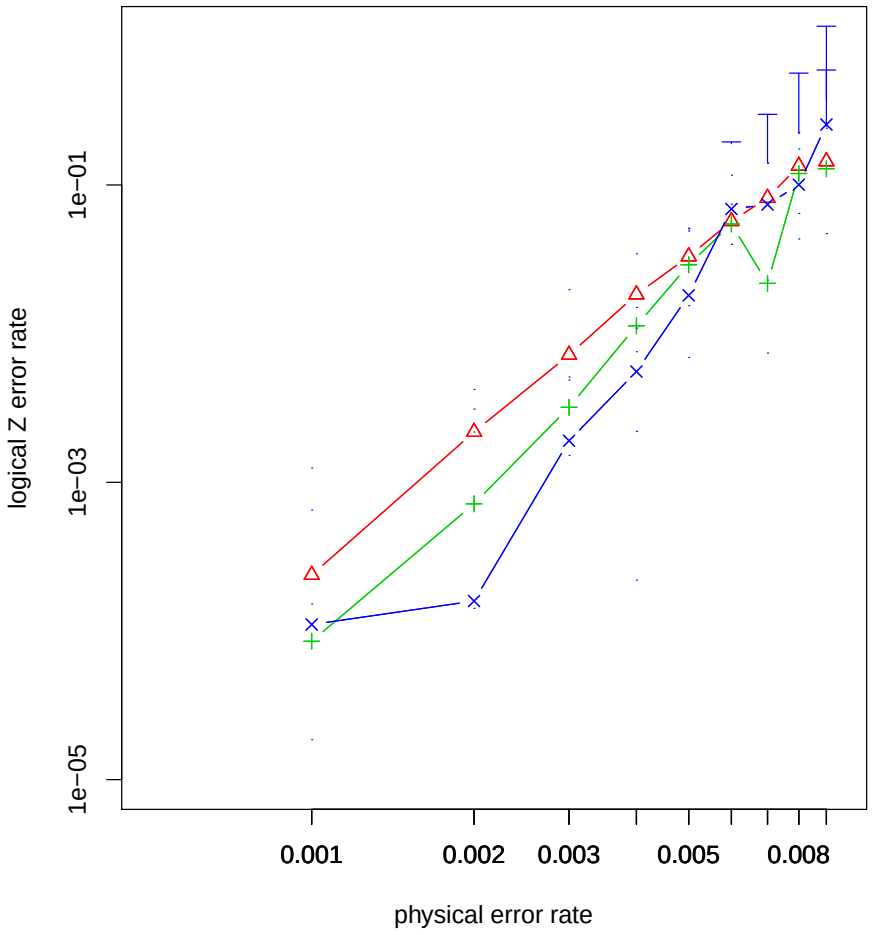

Figure 5.10: The graph of physical error rate versus logical Z error rate in yield 1.0 Blue points are of distance 9, green points are of distance 7, red points are of distance 5.

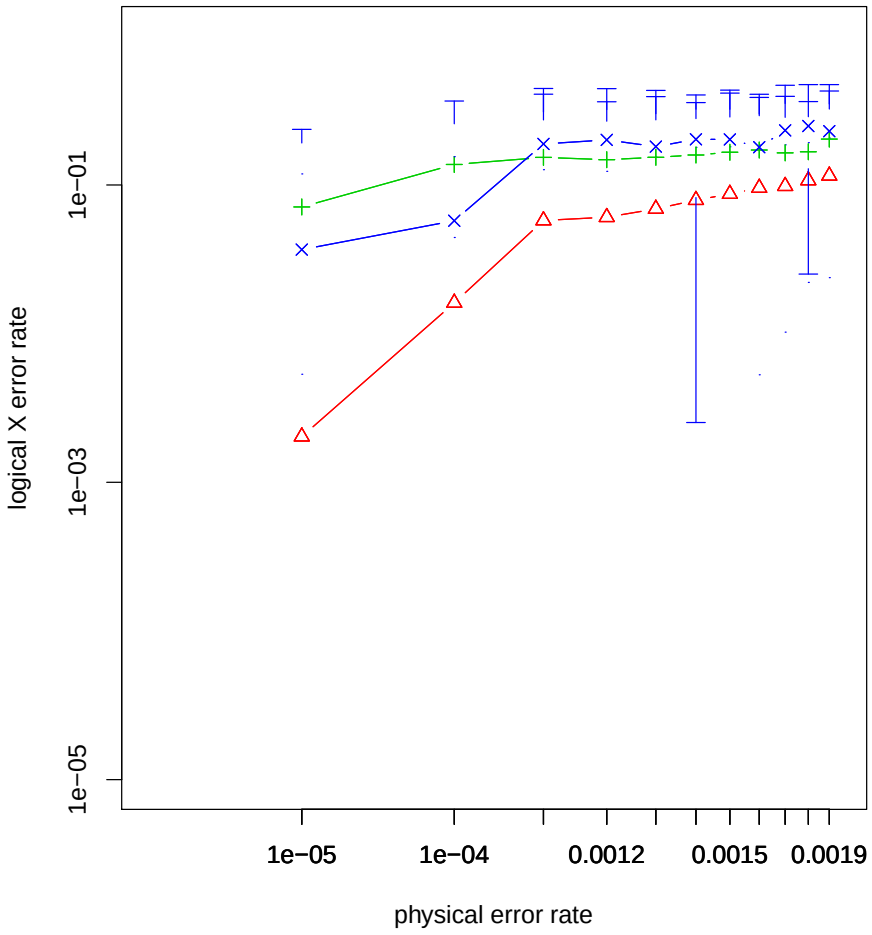

Figure 5.11: The graph of physical error rate versus logical X error rate for yield 0.9. Blue points are of distance 9, green points are of distance 7, red points are of distance 5.

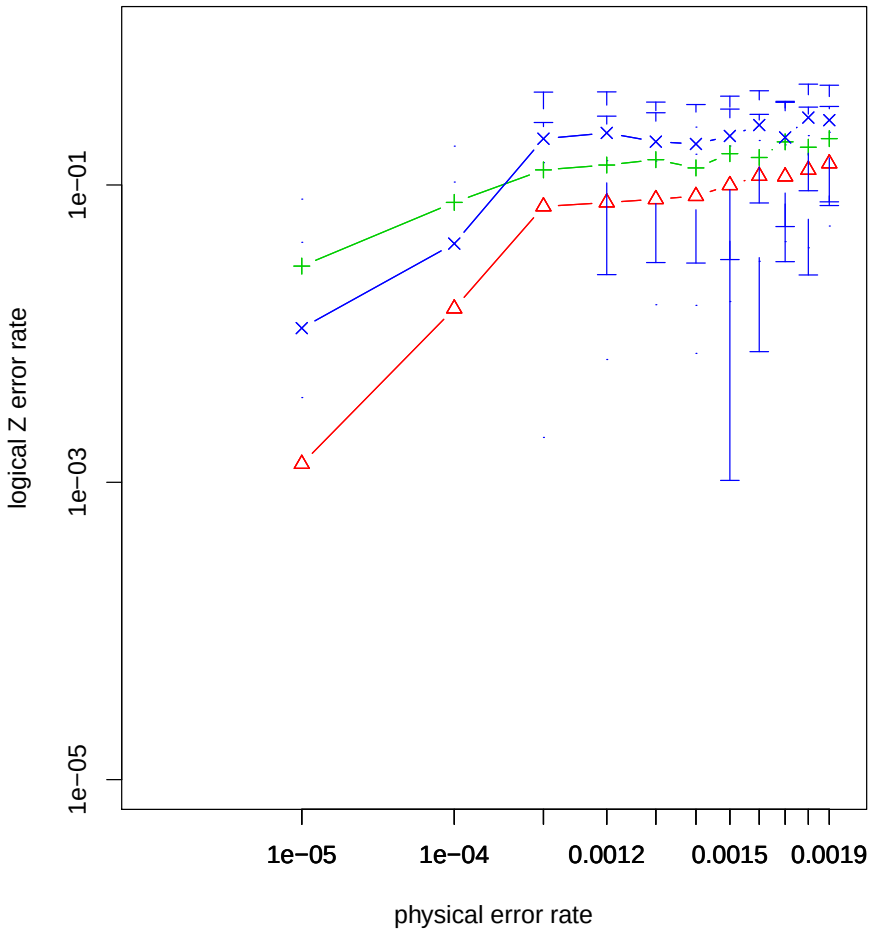

Figure 5.12: The graph of physical error rate versus logical Z error rate for yield 0.9. Blue points are of distance 9, green points are of distance 7, red points are of distance 5.

Chapter 6

Conclusion

6.1 Summary

In this thesis, I have investigated the influence of faulty devices, both data qubits and syndrome qubits. Faulty data qubits cause broken units (plaquettes) for error correction, a problem resolved by connecting broken plaquettes, ignoring the faulty data qubits. Faulty syndrome qubits cause irregular stabilizer circuits, solved by creating and using asynchronous circuits. I have also showed the validity of special plaquettes as the superplaquette and the donut plaquette.

I have numerically calculated two data.

The first shows the relationship between the code distance and the effective code distance, to indicate how the effectiveness of code distance will decline due to faulty devices, as a static simulation. It is fast but approximate. The static simulation implies that the effective code distance declines half for each 10% reduction in yield.

The second data set shows the accurate logical error rate of the surface code working on defective lattices, as a dynamic simulation. The purpose of the dynamic simulation is to calculate the threshold (the physical error rate where the code distance works effectively) The dynamic simulation simulates defective lattices and make complete stabilizer circuits for the lattices, and simulates error occurrence and emulates the error correction. Numerous simulations have been run for varying physical error rates. To accomplish the dynamic simulation, I have made a system which designs surface code circuits corresponding to arbitrary lattice situations and made an interface to have the library which simulates errors and emulates the error correction work along the circuits. The circuit maker composes stabilizer circuits for each error correction units on arbitrary lattice and schedules them. This approach to the dynamic simulation can be applied to the real system of the surface code quantum computer. The dynamic simulation indicates that the threshold of the physical error rate in yield 0.9 is smaller than 10^{-5}. This physical

error rate is too small to implement practically, however, there is still scope to improve the circuit maker and the error correction system, which may push the threshold higher.

As result, both simulation indicates that faulty devices can be tolerated at the expense of making the logical error rate worse. The dynamic analysis also confirms my approach works correctly.

6.2 Future work

The dynamic simulation in this thesis still has room to be improved in applying it to the practical quantum computation in the future.

I have assumed that there are memory errors, physical gate errors and measurement errors. I've not assumed qubits' dynamic loss during the computation. With it, more realistic numerical results can be calculated for certain loss-prone systems.

Additionally, I've considered encoding one qubit on a lattice. Of course we needs multiple qubits for interesting quantum computations and have to consider the influence to logical operations.

In this thesis, the implementation of the surface code is analyzed more realistically than before. However, more research is required to determine the system size of the real implementation of the surface code quantum computer.

Acknowledgement

I gratefully acknowledge my advioer, Professor Jun Murai, Associate Professor Rodney D. Van Meter and Project Assistant Professor Clare Horsman, for their professional advice, guidance and encouragement. I would like to thank professors, associate professors and assistant professors in Keio University, Hideyuki Tokuda, Hiroyuki Kusumoto, Osamu Nakamura, Jin Mitsugi, Keisuke Uehara, Jin Nakazawa, Keiji Takeda, Hitoshi Asaeda, Hideaki Yoshifuji, Kenji Saito, Masaaki Sato, Masafumi Nakane, Achmad Husni Thamrin and Kotaro Kataoka. I particularly thank Dr. Austin G. Fowler for his dedicated and genial guidance. I thank Dr. Hajime Tazaki, Dr. Masayoshi Mizutani and MAUI members. I also thank Toshiaki Hatano, Yuki Nakajima, Kunihiko Shigematsu, Shuhei Yamaguchi, Yutaka Karatsu and Kenji Yonekawa, Asuka Nakajima, Kaori Ishizaki, Pham Tien Trung, Midori Kato, Masakazu Fujimori, Shoichiro Fukuyama, Kouji Murata, Iori Mizutani.

Finally, I acknowledge the member of Jun Murai Laboratory, WIDE project and FIRST Quantum Information Processing Project members.

References

[1] John L. Gustafson. Reevaluating Amdahl's Law. *Communications of the ACM*, 31(5):532–533, May 1988.

[2] Doug Burger, Stephen W. Keckler, Kathryn S. McKinley, Michael Dahlin, Lizy Kurian John, Calvin Lin, Charles R. Moore, James Burrill, Robert G. McDonald, and William Yode. Scaling to the end of silicon with EDGE architectures. *IEEE Computer*, 37(7):44–55, 2004.

[3] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Symposium on Foundations of Computer Science*, pages 124–134, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[4] P. W. Shor. Fault-tolerant quantum computation. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 56–, Washington, DC, USA, 1996. IEEE Computer Society.

[5] Rodney Van Meter, Thaddeus D. Ladd, Austin G. Fowler, and Yoshihisa Yamamoto. Distributed quantum computation architecture using semiconductor nanophotonics. *International Journal of Quantum Information*, 8:295–323, 2010. preprint available as arXiv:0906.2686v2 [quant-ph].

[6] Robert Raussendorf and Jim Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Phys. Rev. Lett.*, 98(19):190504, May 2007.

[7] Robert Raussendorf, Jim Harrington, and Kovid Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9:199, 2007.

[8] Austin G. Fowler, Ashley M. Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80(5):052312, Nov 2009.

[9] N Cody Jones, Rodney Van Meter, Austin G Fowler, Peter L McMahon, Jungsang Kim, Thaddeus D Ladd, and Yoshihisa Yamamoto. A layered architecture for quantum computing using quantum dots. *arxiv*, page 39, 2010.

[10] Netanel H. Lindner and Terry Rudolph. Proposal for pulsed on-demand sources of photonic cluster state strings. *Phys. Rev. Lett.*, 103(11):113602, Sep 2009.

[11] Thomas M. Stace, Sean D. Barrett, and Andrew C. Doherty. Thresholds for topological codes in the presence of loss. *Physical Review Letters*, 102(20):200501, 2009.

[12] Sean D. Barrett and Thomas M. Stace. Fault tolerant quantum computation with very high threshold for loss errors. *Phys. Rev. Lett.*, 105(20):200502, Nov 2010.

[13] D. Schlingemann. Cluster states, algorithms and graphs. *Arxiv preprint quant-ph/0305170*, 2003.

[14] Vladimir Kolmogorov. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*.

[15] A.Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

[16] SB Bravyi and A.Y. Kitaev. Quantum codes on a lattice with boundary. *Arxiv preprint quant-ph/9811052*, 1998.

[17] Austin G. Fowler, Adam C. Whiteside, and Lloyd C. L. Hollenberg. Towards practical classical processing for the surface code. 2011.

[18] Christian D. Lorenz and Robert M. Ziff. Precise determination of the bond percolation thresholds and finite-size scaling corrections for the sc, fcc, and bcc lattices. *Phys. Rev. E*, 57:230–236, Jan 1998.