# Measurement Based Quantum Computing and Surface Codes

Shaurya Rawat
Mentored by Prof. Ramesh Pyndiah, IMT Atlantique

**Abstract**

We studied the measurement of quantum states (as qubits) and its application in implementing Quantum Algorithms by performing measurements on qubits, called 1-Way Quantum Computing. We characterize and identify the stabilizer theory for this computation. We then move on to the implementation of Quantum Error Correction as part of the study on Stabilizer Theory.

## 1 Introduction

Quantum computation offers a promising way of information processing, where the non-classical features of quantum mechanics like entanglement and superposition can be harnessed and exploited to our advantage and provide exponential speedup as compared to classical computing for some specific computations. Different models for quantum computation include :

1. **The Quantum circuit (or Gate based) model** - Unitary gates (unitary transformations) act as quantum channels for information processing. . Many proposals for the implementation of quantum computation are designed around this model, including physical prescriptions for implementing the elementary gates(Has a large number of well known algorithms, like Shor's, Grover Search, etc).(Most prevalent approach currently and the focus of many top tier companies like IBM (Qiskit), Google etc. And a number of SDKs and libraries already in place for computing and simulation, e.g. Qiskit, Pennylane, Q, Google circ, etc.

2. **Adiabatic quantum computation** - In this model, we make the quantum system evolve slowly from its initial state to the final state( state being defined by the problem at hand), and the evolution to final state is attained keeping the system in the ground state (adiabatic theorem). Shown to solve optimization problems like travelling salesman, max cut, etc. (Refer DWave Computing). Currently have the most number of working qubits in place. Their latest machine has 2000 qubits. Refer: DWave Quantum Annealers

3. **Quantum Turing Machine**- The quantum Turing machine (QTM) is the quantum analog of the classical Turing machine (TM). Shown to solve NP hard problems in polynomial time, thus it classifies as a Quantum Computing model.

4. **Measurement-based models (MCQ)** Creation of a cluster state and entanglement Any quantum computation via only measurement. Advantage : If hardware has good error threshold for measurements.
   Note : Measurements on entangled states play a key role in many quantum information protocols, such as quantum teleportation and entanglement-based quantum key distribution

   - Teleportation-based model [2] (Introduced by Gottesman and Chuang) - Using Teleportation Protocol as a building block and measurements in a rotated bell basis. (joint measurements of qubits using POVMs) (Highly complex and difficult to work with algorithms)
   - One-way quantum computer [5] (Introduced by Raussendorf and Briegel) - Approaches quantum computing in a different way using quantum measurements as a basis of computation We focus our study on this model of computing.

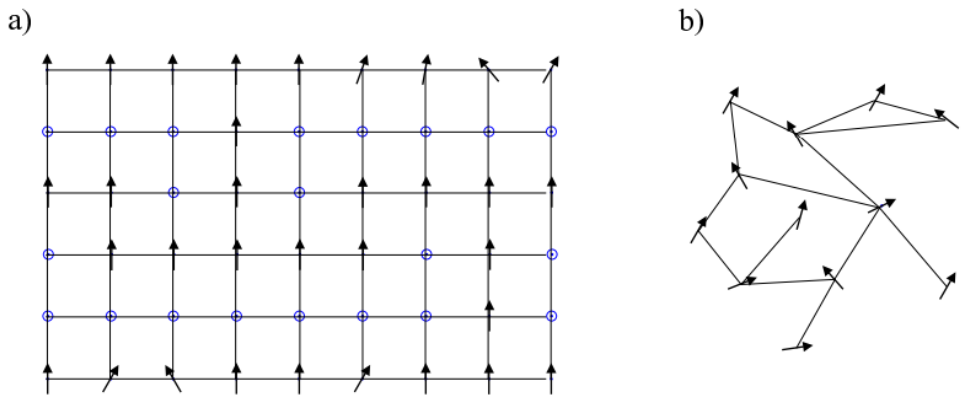



Figure 1: One way quantum computation consists of single-qubit measurements in certain bases and in a certain order on an entangled resource state. Cluster states have a square lattice structure (a) while the freedom of choosing specific general graph states such as illustrated in (b) can reduce the number of qubits needed for a given computation significantly.

## 1.1 One-way quantum computer

In One-way quantum computation, the first step is to prepare the system in a highly entangled quantum state, the 2D-cluster state or graph state (also called as the universal resource), independently of the quantum algorithm which is to be implemented. In the second step, the qubits in the system are measured individually, in a certain order and basis—and it is this measurement pattern which specifies our quantum algorithm.

Therefore, in one-way quantum computation, the quantum correlations in such highly entangled cluster states are exploited to allow universal quantum computation through single-qubit measurements alone

. Cluster states and a few other resource states have been created experimentally in various physical systems and the measurement-based approach offers a potential alternative to the standard circuit approach to realize a practical quantum computer. linear-optic quantum computation of Knill, Laflamme and Milburn

## 1.2 Cluster states and graph states

See, figure 1.

- With each state, we associate a graph, a set of vertices and edges connecting vertex pairs
- Each vertex on the graph corresponds to a qubit
- preparing every qubit in the state $|+\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle)$ to create our graph state.
- Apply a controlled $\sigma_z$ (CZ) operation $|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \sigma_z$ on every pair of qubits whose vertices are connected by a graph edge.
- This prepares our cluster state, which is a subset of graph state, and is highly entangled.
- Can be implemented as an $n-$dimensional square grid. The extra flexibility in the entanglement structure of graph states means that they often require far fewer qubits to implement the same one-way quantum computation.

## 1.3 Unitary Gates and Rotations

In Gate-based computing, we use unitary operations on our quantum state as building blocks of operations. Unitary operations correspond to rotations on the Bloch sphere and have the following form. A rotation around

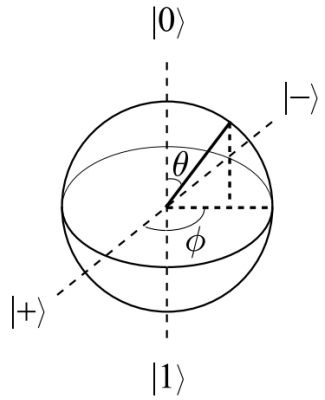



Figure 2: Single-qubit projective measurements will be represented by the pair of angles $(\theta, \phi)$ of the co-latitude $\theta$ and longitude $\phi$ of their $+1$ eigenstate on the Bloch sphere. This corresponds to a measurement of the observable $U_z(\phi+\pi/2)U_x(\theta)ZU_x(-\theta)U_z(-\phi-\pi/2)$.

the $k$ axis (where $k$ is $x$, $y$, or $z$) by angle $\phi$ can be written

$$U_k(\phi) = e^{-\frac{i\phi}{2}\sigma_k} \tag{1}$$

For clarity, we will use the notation $X \equiv \sigma_x$ , $Y \equiv \sigma_y$, etc. Note : Standard unitary operations for the eigenstates of $Z$ and $X$ are :

$$\begin{aligned} Z|0\rangle &= |0\rangle \\ -Z|1\rangle &= |1\rangle \\ X|+\rangle &= |+\rangle \equiv \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ -X|-\rangle &= |-\rangle \equiv \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2}$$

Note : Therefore, all unitary operations can be represented as rotations around the Bloch Sphere. For Bloch representation of unitary gates in Qiksit, refer here.

$$\begin{aligned} X|0\rangle &= |1\rangle \\ Z|+\rangle &= |-\rangle \equiv \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{3}$$

## 1.4 Single Qubit Measurements (Translating unitary operations to measurements)

Single-qubit measurements in a different bases play a key role in one-way quantum computation, as this measurement pattern is what defines our algorithm.

Using a Bloch sphere picture, every projective single-qubit measurement can be associated with a unit vector on the sphere, which corresponds to the $+1$ eigenstate of the measurement. We can then parameterize observables by the co-latitude $\theta$ and longitude $\phi$ of this vector (illustrated in figure 2), as a pair of angles $(\theta, \phi)$. A measurement with angles $(\theta, \phi)$ corresponds to a measurement of the observable

$$M(\theta, \phi) = U_z(\phi + \pi/2)U_x(\theta)ZU_x(-\theta)U_z(-\phi - \pi/2) \tag{4}$$

Here in 1.1, M$(\theta, \phi)$ is a measurement along the basis represented by $(\theta, \phi)$ on Bloch Sphere (Refer Fig 2). One way of implementing such a measurement is to apply the single-qubit unitary $U_x(-\theta)U_z(-\phi - \pi/2)$ to the qubit before measuring it in the computational basis. Therefore :

$$M(\theta, \phi) = M(z - basis) \ . \ U_x(-\theta)U_z(-\phi - \pi/2) \tag{5}$$

Note : z basis on Bloch Sphere ($\theta = 0$). Also the unitaries, $U_x$ and $U_z$ refer to :

$$U_z(\phi) = e^{-\frac{i\phi}{2}\sigma_z} \quad ; \quad U_x(\phi) = e^{-\frac{i\phi}{2}\sigma_x} \tag{6}$$

## 1.5 Trivial One Way Computation via 2 qubits (Using Projective Measurement)

We begin with a basic example where we have an unknown input state in first qubit.[6] $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. and another qubit in the $|+\rangle$ state $= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. A Controlled-Z (CZ) operation is applied on the two qubits (for creating our maximally entangled cluster state). The state of the two qubits is then

$$\frac{1}{\sqrt{2}}\left(\alpha|0\rangle|+\rangle + \beta|1\rangle|-\rangle\right) . \tag{7}$$

The first qubit is now measured in the M$(\pi/2, \phi)$ basis $\{(1/\sqrt{2})(|0\rangle \pm e^{i\phi}|1\rangle)\}$, where $\phi$ is a real parameter. This corresponds, in the the Bloch sphere picture, to a measurement along the basis vector in the $x$-$y$ plane at angle $\phi$ to the $x$ axis.

Since the two qubits are entangled, the measurement of first qubit projects the second qubit into a state specified below. There are two possible outcomes to the measurement, which occur with equal probability. If the measurement returns the $\pm 1$ eigenvalue, the second qubit will be projected into the state (Projective Measurement)

$$\alpha|+\rangle \pm e^{i\phi}\beta|-\rangle . \tag{8}$$

This result can be verified by using the concept of POVM. (Also, the concept behind teleportation protocol)

Introduce the binary digit $m \in \{0, 1\}$ to represent a measurement outcome as $(-1)^m$. The state of qubit two can then be written, up to a global phase,

$$X^m H U_z(\phi)|\psi\rangle . \tag{9}$$

We see that the unknown input state which was prepared on the first qubit has been transferred to qubit two without any loss of information and coherence. In addition to this it has undergone a unitary transformation: $X^m H U_z(\phi)$.

Here, the angle of the rotation $U_z(\phi)$ is set in the choice of measurement basis. By altering the measurement basis M$(\phi)$, we can perform any unitary operation to the input qubit, with the output appearing on the second qubit.

Note : The unitary transformation $HU_z(\phi)$ is accompanied by an additional Pauli transformation $(X)$ when the measurement outcome is $-1$. This is a typical feature of one-way quantum computation; due to the randomness of the measurement outcomes, any desired unitary can be implemented only up to random but known Pauli transformations. Since these Pauli operators are an undesired by-product of implementing the unitary in the one-way model, they can be treated as "by-product operators" , and can be ignored for our analysis.

Graphical Notation : In figure 3 (a), this protocol is represented using a graphical notation that we will use throughout this chapter. The input qubit is represented by a square, and the output qubit by a lozenge, a smaller square tilted at $45°$. The CZ operation applied to the two qubits is represented by a line between them.

## 1.6 Connecting one-way qubits - via consecutive single-qubit measurements

Euler's rotation theorem states : any single-qubit SU(2) rotation can be decomposed as a product of three rotations $U_z(\gamma)U_x(\beta)U_z(\alpha)$. Thus, by repeating the simple two-qubit protocol (above) three times, any arbitrary single-qubit rotation may be obtained (up to an extra Hadamard, which can be accounted for via the binary m). Two one-way patterns are combined as one would expect, the output qubit(s) of one pattern become the input qubit(s) of the next. The main issue in connecting patterns together is to track the effect of the Pauli by-product operators which have accumulated due to the previous measurements.

Concatenating the two-qubit protocol three times, with different angles $\phi_1$, $\phi_2$ and $\phi_3$ gives the one-way pattern illustrated in figure 3 (b).

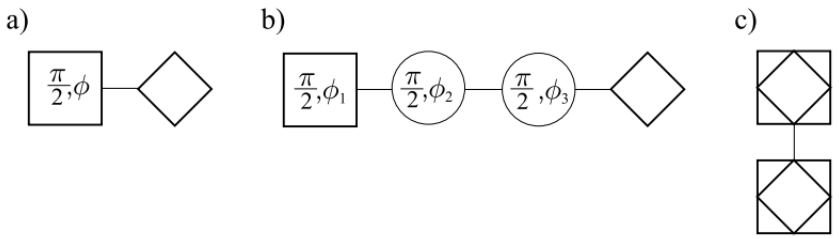



Figure 3: The one-way graph and measurement patterns for a) the single-qubit operation $HU_z(\phi)$ and b) an arbitrary single-qubit operation, $U_z(\gamma)U_x(\beta)U_z(\alpha)$, when the measurement angles are set $\phi_1 = \alpha$, $\phi_2 = (-1)^{m_1}\beta$ and $\phi_3 = (-1)^{m_2}\gamma$, and $m_a$ is the binary measurement outcome of the measurement on qubit $a$. Note that this imposes an ordering in the measurements of this pattern. This second pattern is made by composing three copies of the pattern (a) with differing measurement angles as described in the text. Pattern (c) implements a CZ operation. Input and output qubits coincide for this pattern.

Here in Fig 3, 1.5 : The square figures denote our input qubit, and slanted sqaures (lozenge) denote our output qubit, with the angles inside refering to the measurement basis for that particular qubit. Note, Fig 3(c) denoting a C-Z has both a 2 qubit input and output cluster state, highlighting the entanglement.

To see the effect of the by-product operators from each measurement, let us label the binary outcome from each $m_a$. The unitary implemented by the combined pattern is therefore

$$U = HZ^{m_3}U_z(\phi_3)HZ^{m_2}U_z(\phi_2)HZ^{m_1}U_z(\phi_1). \tag{10}$$

Since $HZH = X$ and $HU_z(\phi)H = U_x(\phi)$ this can be rewritten

$$HZ^{m_3}U_z(\phi_3)X^{m_2}U_x(\phi_2)Z^{m_1}U_z(\phi_1). \tag{11}$$

We can rewrite this further using the identities $XU_z(\phi) = U_z(-\phi)X$ and $ZU_x(\phi) = U_x(-\phi)Z$,

$$X^{m_3}Z^{m_2}X^{m_1}HU_z((-1)^{m_2}\phi_3)U_x((-1)^{m_1}\phi_2)U_z(\phi_1). \tag{12}$$

Now we have split up the operation in the same way as the two-qubit example, a unitary plus a known Pauli correction. In this case, however, this unitary is not deterministic – the sign of two of the rotations depends on two of the measurements. Nevertheless, if we perform the measurements sequentially and choose measurement angles $\phi_1 = \alpha$, $\phi_2 = (-1)^{m_1}\beta$ and $\phi_3 = (-1)^{m_2}\gamma$, we obtain deterministically the desired single-qubit unitary.

The dependency of measurement bases on the outcome of previous measurements is a generic feature of one-way quantum computation. This dependency means that there is a minimum number of time-steps in which any one-way quantum computation can be implemented. (Hardware limitation)

The Pauli corrections remaining at the end of the implemented one-way quantum computation are unimportant and never need to be physically applied; they can always be accounted for in the interpretation of the final measurement outcome. For example, if the final state is to be read out in the computational basis any extra $Z$ operations commute with the measurements and have no effect on their outcome. Any $X$ operations simply flip the measurement result, and thus can be corrected via classical post-processing.

## 2 Cluster State 1-Way Quantum Computing (Large Scale)

We move on to define one-way quantum computing, on a larger scale than just connected linear qubits. For this, we introduce the concept of graph states and 2-qubit measurements [1]. We will then see the stabilizer formalism which will establish the basis of our 1-Way quantum computation model, physically implementable as square lattice cluster states, which can be generated more efficiently. Note : A number of measurement patterns for quantum gates designed specifically for two-dimensional square lattice cluster states can be found here

## 2.1 Graph states as a resource

An important observation is that, given a one-way pattern of such connected qubits as described above, all of the measurements can be made after all the CZ operations have been implemented. Therefore, quantum algorithms can begin by initializing qubits to a fiducial starting state. This state is usually $|+\rangle$ on each qubit. When the input qubits of a one-way graph measurement pattern are prepared in $|+\rangle$, then the entangled state (maximally) generated by the CZ gates is our graph state. Thus the graph state can be considered a resource for this quantum computation. This graph state description fits well with physical implementations of qubits, such as in linear optics.

## 2.2 Two-qubit gates

So far we have seen how an arbitrary single-qubit operation could be achieved in one-way quantum computation in a simple linear one-way pattern. However, for universal quantum computation, entangling two-qubit gates are necessary. One such gate is a CZ gate. There is a particularly simple way in which the CZ can be implemented within the one-way framework. This is simply to use the CZ represented by a single graph-state edge to implement the CZ directly. This leads to the one-way pattern illustrated in figure 3 (c). Notice that here the input qubits are also the output qubits. This is indicated by the superimposed squares and lozenges.

Now, The main tool we shall use in our investigation of these properties is the *stabilizer formalism*.

## 2.3 Stabilizer formalism (Basic Theory)

The stabilizer formalism[3] is a powerful tool for understanding the properties of graph states and one-way quantum computation. Stabilizer formalism is a framework whereby states and sub-spaces over multiple qubits are described and characterized in a compact way in terms of operators under which they are invariant. In standard quantum mechanics one uses complete sets of commuting observables in a similar fashion, such as in the description of atomic states by "quantum numbers". Stabilizer operators acting on our qubits, projects our qubit state to the +1 eigenbasis of the qubit states. Thus, we can use this projective scheme to manipulate our qubits in a way we have seen before.

An operator $K$ *stabilizes* a subspace $\mathcal{S}$ when, for all states $|\psi\rangle \in \mathcal{S}$,

$$K|\psi\rangle = |\psi\rangle. \tag{13}$$

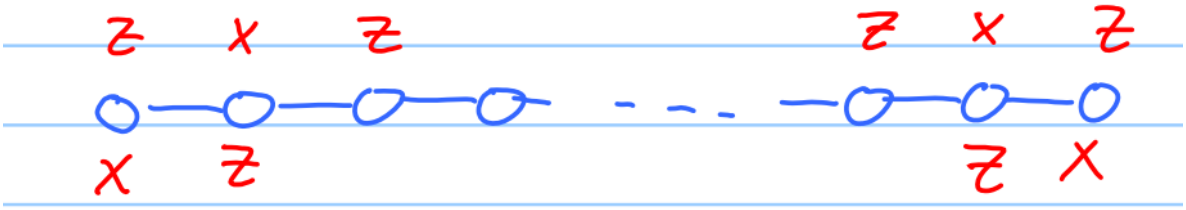
In other words, $|\psi\rangle$ is an eigenstate of $K$ with eigenvalue $+1$.

In the stabilizer formalism one focuses on operators which, in addition to this stabilizing property, are Hermitian members of the Pauli group, i.e. tensor products of Pauli and identity operators. The key principle of the stabilizer formalism is to identify a set of such stabilizing operators which uniquely defines a given state or sub-space - i.e. there is no state outside the sub-space (for a specified system) which the same set of operators also jointly stabilizes. The sub-spaces (and states) which can be defined uniquely using stabilizing operators from the Pauli group are called *stabilizer sub-spaces* (or stabilizer states).

Stabilizer states and sub-spaces occur widely in quantum information science and include Bell states, GHZ states, many error-correcting codes, and, of course, graph states and cluster states. Note that there are other joint eigenstates of the stabilizing operators with some $-1$ eigenvalues. However, only states with $+1$ eigenvalue are "stabilizcan, by definition. This set of operators then embodies all the properties of the state and can allow an easier analysis, for example, of how the state transforms under measurement and unitary evolution. Since the product of two stabilizing operators is itself stabilizing, the set of operators which stabilize a sub-space has a group structure. It is called the *stabilizer group* or simply the *stabilizer* of the sub-space. The group can be compactly expressed by identifying a set of generators. For a $k$-qubit sub-space in an $n$ qubit system, $n-k$ generators are required.

A simple example of a stabilizer state is the state $|+\rangle$. Its stabilizer group is generated by $X$ alone. The stabilizer for the tensor product state $|+\rangle^{\otimes n}$ is then generated by $n$ operators $K_a = X_a$ acting on each qubit $a$. From this we can derive the stabilizer generators for graph states. Consider a stabilizer state transformed by the unitary transformation $V$. The stabilizers of the transformed state are then given by $VK_aV^\dagger$. Since the CZ gate transforms $X \otimes \mathbb{1}$ to $X \otimes Z$ under conjugation, we find that the stabilizer generators for graph states have the form

$$K_a = X_a \prod_{b \in N(a)} Z_b \tag{14}$$




for every qubit $a$ in the graph. $N(a)$ is the neighbourhood of $a$, i.e. the set of qubits sharing edges with $a$ on the graph (this corresponds to nearest neighbours in a cluster state).

## 2.4 Key discussions over applying the stabilizer formalism

$$K_a = X_a \prod_{b \in N(a)} Z_b \tag{15}$$

The above stabilizer generators, create the graph state for our model, which, upon measurement in different basis, can perform our 1-way computation as seen in linearly connected model. We created the cluster states for 2,3,4 and more qubits, using the formulation above, the circuit for which is attached below. We also analyzed the universal gates using this model, which therefore theoretically allows us to perform any computation, as all computations can be broken down to Single qubit Pauli unitaries and H gate and the two qubit CNOT Gate. To create more complex algorithms, would require higher order cluster states (possibly 3-D) which is currently beyond the scope of current technology. (Our model is sensitive to measurement based errors). Advantage can be when we can establish much better measurement techniques, although work is being done in using quantum optics as the preferred mode.

In the end, we see that the stabilizer model forms the key component of 1-way computation. This stabilizer model is also the basis for error correction as we work on the logical qubit space (codespace) of the given data qubits, for error-free computation.

## 2.5 Implementation in IBM Q

$$\begin{aligned} Z|0\rangle &= |0\rangle \\ Z|1\rangle &= -|1\rangle \\ Z|+\rangle &= |-\rangle \\ Z|-\rangle &= |+\rangle \end{aligned} \tag{16}$$

### 2.5.1 Stabilizer generation in linear array (and 2D) of qubits

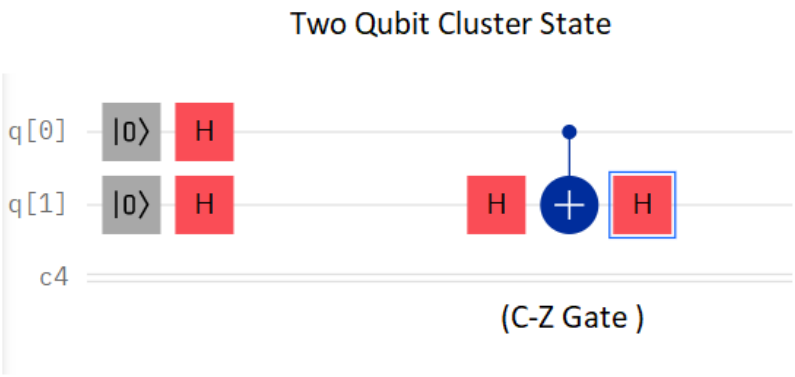
We prepare the qubits in $|++....+\rangle$ state. And apply CZ operation. A cluster state is a stabilizer state, a simultaneous eigenstate with eigenvalue 1 of a set of commuting Pauli operators.

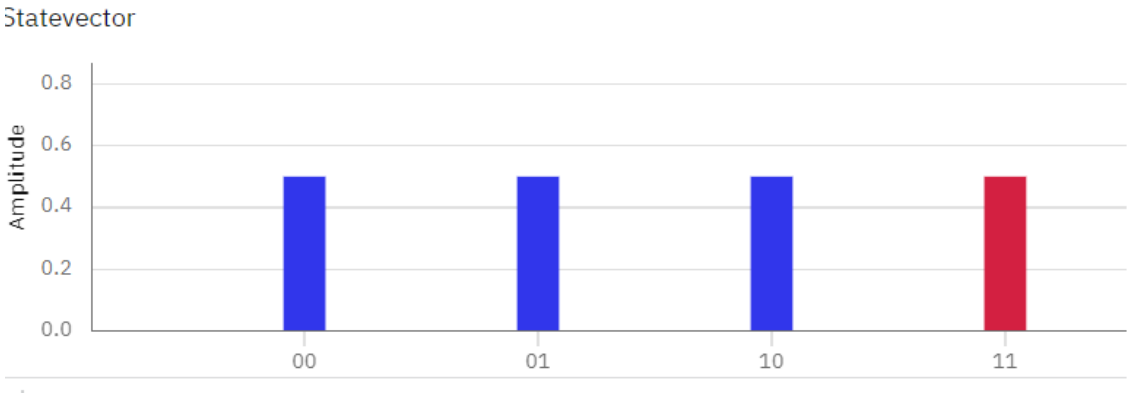
$$K_a = X_a \otimes Z_b \quad is\ our\ generator. \tag{17}$$

In the one dimensional lattice, there is a stabilizer generator associated with the $i_{th}$ qubit, namely $Z \otimes X \otimes Z$ acting on qubits $i_{-1}, i, i_{+1}$, unless the qubit is at the end of the chain. For the first qubit the stabilizer generator is XZ acting on qubits 1, 2, and for the nth (last) qubit is is ZX acting on qubits n-1 and n.

The controlled-Z gates transform the stabilizer IXI acting on three successive qubits to the stabilizer ZXZ of the cluster state. We can apply the same construction to any graph: Starting with $|+\rangle$ at each vertex of the graph, we apply controlled-Z to each pair of qubits connected by an edge. The corresponding state is called a graph state. The term cluster state is used when the graph is a regular lattice, like the 1D chain, or a 2D square lattice.

Figure 1: 2 Qubit Cluster State




Attached are the 2,3,4 qubit linear implementation of cluster states. These are given by :

$$
\begin{aligned}
|\phi\rangle_{C2} &= \frac{|0\rangle|+\rangle + |1\rangle|-\rangle}{\sqrt{2}} \\
|\phi\rangle_{C3} &= \frac{|+\rangle|0\rangle|+\rangle + |-\rangle|1\rangle|-\rangle}{\sqrt{2}} \\
|\phi\rangle_{C4} &= \frac{|+\rangle|0\rangle|+\rangle|0\rangle + |+\rangle|0\rangle|-\rangle|1\rangle + |-\rangle|1\rangle|-\rangle|0\rangle + |-\rangle|1\rangle|+\rangle|1\rangle}{2}
\end{aligned}
\tag{18}
$$

### 2.5.2 Implementation of general SU(2) gates, H-gate, CNOT gate using cluster states

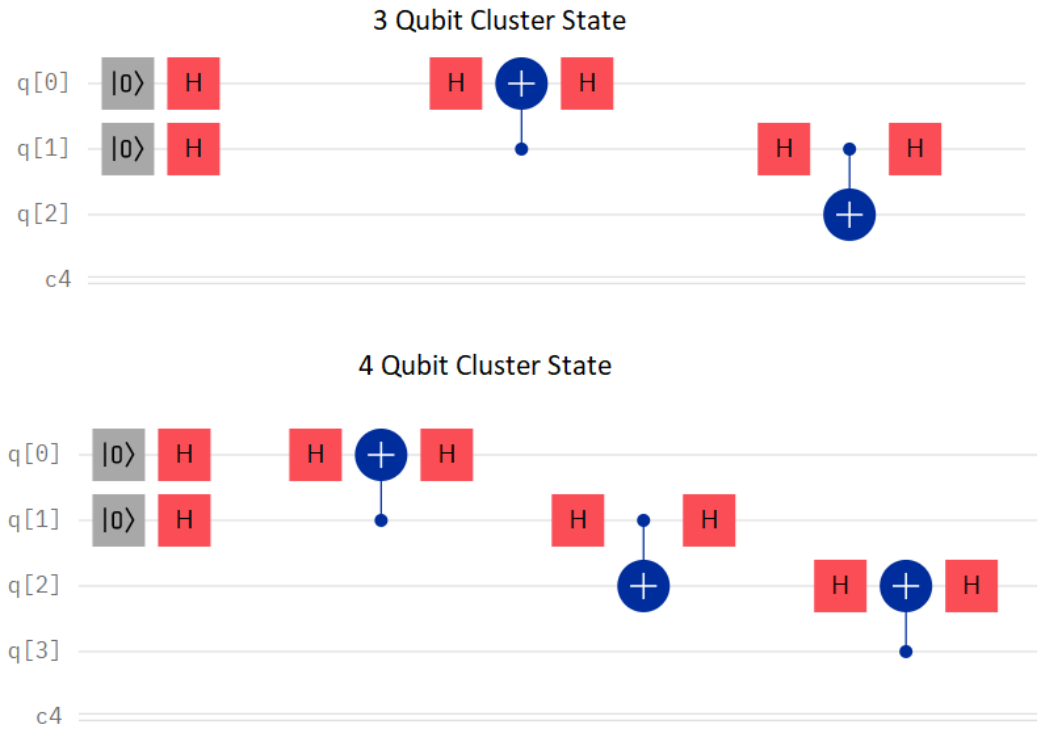
We need 5 qubits for a general SU(2) rotation using measurement based computing. Realization scheme showed in Figure attached. Steps :

- Prepare the qubit state $|\psi\rangle_C 5 = |\psi\rangle_{in} \otimes |+\rangle_i$ (i runs from 2 to 5)
- Entangle the five qubits of the cluster $C_5$ via the unitary operation S(CNOTs) (as in circuit figure) to prepare our cluster state.
- Measure the qubits in the basis as described in the figure to obtain the required unitary operation.
- Implemented the Hadamard gate on IBMQ. (in figure)

### 2.5.3 Measurement Based Computing and Error Correcting codes

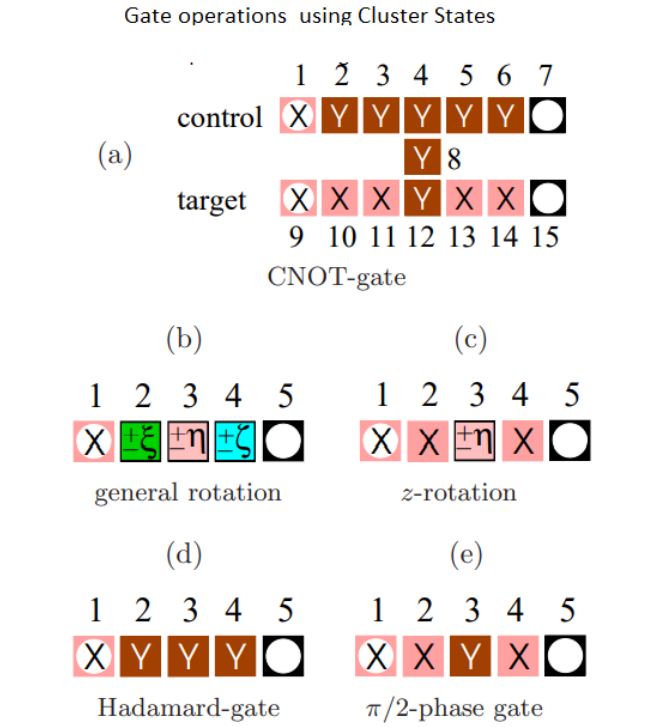
Thus by implementing the single Pauli gates, H-Gate and the two qubit CNOT, we obtain the complete set of universal gates for computation. We can see the analogy between measurement-based computing and the error correcting codes via the stabilizer formalism. How we can project the combined qubit state into a subspace of

Figure 2: 3 Qubit Cluster State



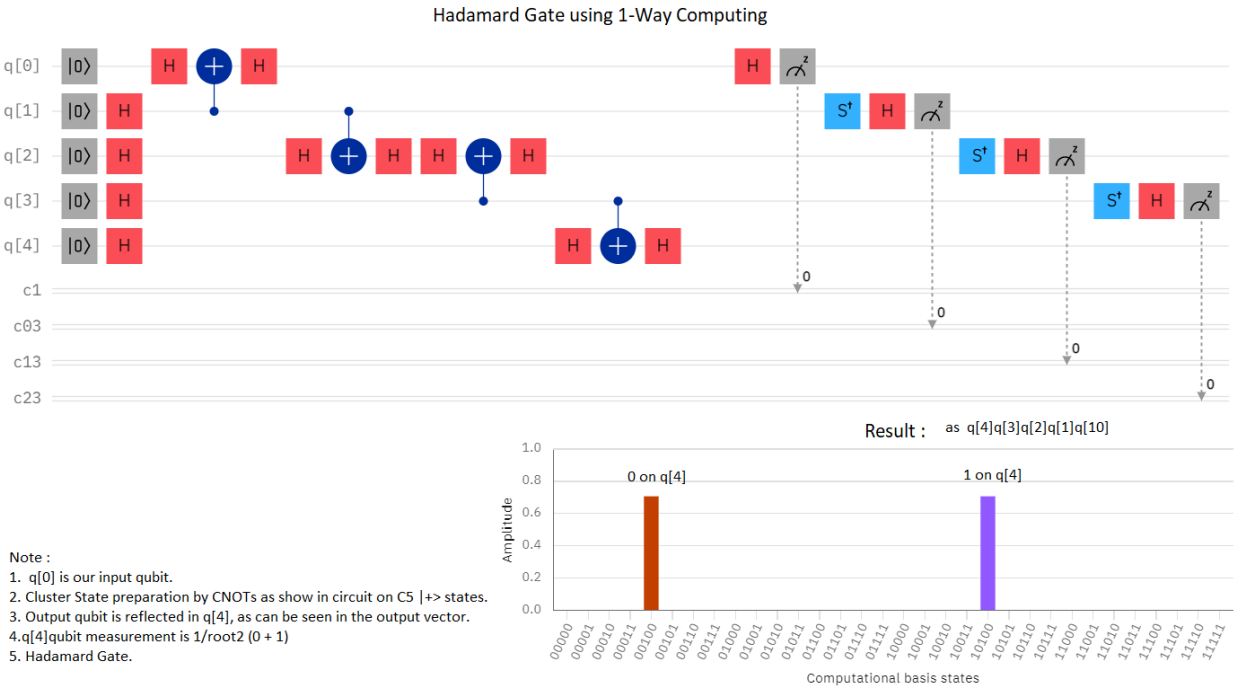

the initial state of all the qubits and perform measurements to do desired computation. In error correction, say [n,k,d] code, we encode k logical qubits into a subspace of n data qubits.

Figure 3: Gate Operation using 1-Way Computing (Figure Shows measurement basis)




Realization of elementary quantum gates using 1-Way Computation. Each square represents a lattice qubit. The squares in the extreme left column marked with white circles denote the input qubits, those in the right-most column denote the output qubits.

Figure 4: Hadamard Gate using 1-Way Computing




Note :
1. q[0] is our input qubit.
2. Cluster State preparation by CNOTs as show in circuit on C5 |+> states.
3. Output qubit is reflected in q[4], as can be seen in the output vector.
4.q[4]qubit measurement is 1/root2 (0 + 1)
5. Hadamard Gate.

# 3 Error Correction

## 3.1 Depolarising error Channel

Added the error model and code below for error rate(gate, measurement)= (0.1,0.1)

## 3.2 Stabilizer Formalism for Error Correcting Code

Stabilizer codes are a natural quantum extension of certain classical codes. As such, they are inherently suited for dealing with bit and phase flip errors, e.g. the Pauli X, Y , or Z on some qubits. In a quantum computer, an infinite amount of possible single-qubit errors, corresponding to any rotation on the Bloch sphere, could in principle happen, making such a translation seem inappropriate. It turns out, however, that when measuring the presence of one of these errors, one essentially "projects" the continuous error into discrete ones. In general, if a QECC can correct against a set of errors, then it can also protect against any linear combination of these. The Pauli matrices, together with the identity I, form a basis for the space of complex 2 × 2 matrices, and are thus sufficient to represent all unitary single qubit errors. Stabilizer codes are uniquely determined by their so-called stabilizer S, which is the group of operators that leave the elements of the codespace unaffected. The codepace C(S) is then defined as the eigenspace of S with eigenvalue +1

C(S) = $|\psi\rangle : \forall g \in S, \quad g|\psi\rangle = |\psi\rangle$. Here g denotes the elements of stabilizer group. Valid ones are abelian subgroups of the n-fold Pauli group $G_n$. For a code with $n$ physical qubits, encoding $k$ logical ones, there will be $n_k$ elements. That is, S = $< g_1, g_2, ...., g_{n-k} >$. Errors are identified by measuring the eigenvalues $g_l|\psi\rangle = \beta_l|\psi\rangle$ , where $\beta_l = \pm$ 1, which gives a corresponding 'syndrome' : ${\beta_l}_{l=0}^{n-k}$.
A trivial syndrome, $\beta_l$ = 1 for all l, means that the state is still stabilized by S and thus has no errors.

## 3.3 Depolarising Error Model and Correction (Steane Code and Shor Code)

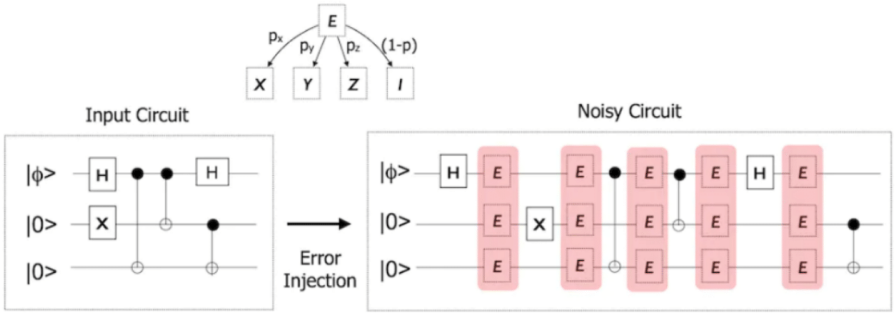
## 3.4 QECSIM Library simulations

QECSIM [4] is a useful tool for checking the validity of error correcting codes for different error rates and different error models, whether or not, our code can correct for these errors. We cross-verify with the stabilizer formalism to check the states produced. Each code will have its own set of stabilizer group. In essence, from measurement of ancilla qubits, we can deduce, if in +1 eigenstate, no error occured).

# Depolarising error Model

Selecting the depolarizing channel error model introduces a random error in between each operation on a qubit. These errors are injected in a form of bit-flips (x error), phase-flips (z error) or both at the same time (y error), using equal probabilities for x, y and z errors.

The error injection process is depicted in the following figure which illustrate how errors are injected in a perfect circuit to produce a noisy circuit




The depolarizing channel is defined as:

$$E(\rho) = (1-\lambda)\rho + \lambda \mathrm{Tr}[\rho]\frac{I}{2^n}$$

with $0 \leq \lambda \leq 4^n/(4^n-1)$

CPTP map with Kraus representation as :

$$\Delta_\lambda(\rho) = \sum_{i=0}^{3} K_i \rho K_i^\dagger,$$

We can express I,X,Y,Z as $K_i$ which when operated on $\rho$ sum up to I. (Therefore valid map)

In [1]:
```
from qiskit.providers.aer.noise import NoiseModel
from qiskit.providers.aer.noise.errors import pauli_error, depolarizing_error
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit, Aer, transpile, assemble
from qiskit.visualization import plot_histogram
aer_sim = Aer.get_backend('aer_simulator')
```

In [2]:
```
def get_noise(p_meas,p_gate):
    error_meas = pauli_error([('X',p_meas), ('I', 1 - p_meas)])
    error_gate1 = depolarizing_error(p_gate, 1)
    error_gate2 = error_gate1.tensor(error_gate1)

    noise_model = NoiseModel()
    noise_model.add_all_qubit_quantum_error(error_meas, "measure") # measurement error is appli
    noise_model.add_all_qubit_quantum_error(error_gate1, ["x"]) # single qubit gate error is ap
    noise_model.add_all_qubit_quantum_error(error_gate2, ["cx"]) # two qubit gate error is appl

    return noise_model
```

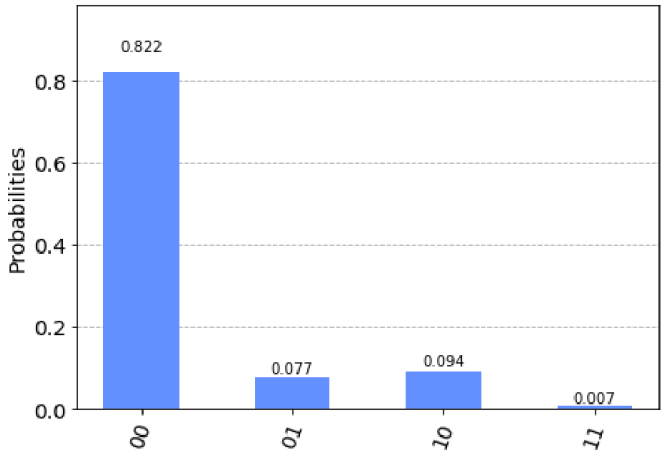
In [4]:
```
noise_model = get_noise(0.1,0.1)
#Gate based and Measurement based error.
```

In [5]:
```
qc0 = QuantumCircuit(2) # initialize circuit with three qubits in the 0 state
qc0.measure_all() # measure the qubits

# run the circuit with the noise model and extract the counts
qobj = assemble(qc0)
counts = aer_sim.run(qobj, noise_model=noise_model).result().get_counts()

plot_histogram(counts)
```

Out[5]:




In [ ]:

In [ ]:

## QECSIM Documentation and Examples

### QEC SIM :

Python 3 package to simulate quantum error correction using stabilizer codes. It is lightweight, modular and extensible( allowing additional codes to be added), and we can also modify or even plug in own error models and decoders(for syndrome measurements)

**Three key abstract classes:**

1. *qecsim.model.StabilizerCode*:Contains syntax for working with various stabilizer codes and logical subspaces
2. *qecsim.model.ErrorModel* : Encodes a error model via a probability distribution function. (modular)
3. *qecsim.model.Decoder* : Class contains objects for syndrome measurements.

Important syntax aside from classes :

- Error_probability : Input feeded to our probability distribution function. Error probability to be float in [0, 1].
- qecsim.app.run_once(code, error_model, decoder, error_probability, rng=None) : Run the final circuit with all classes

- Binary Symplectic Form $\boldsymbol{A} \circ \boldsymbol{B}$ is given by A.I.B ; Here I = Tensor with (Identity on diagonal elements) Pauli Operators (Stabilizers, logical operators and recovery operators) are represented in Binary Symplectic form It is means of representing elements of the Pauli group (neglecting global phases) using binary vectors a and b such that an element P of the Pauli group acting on n qubits is $\boldsymbol{X}^{a}\boldsymbol{Z}^{b} = \boldsymbol{X}^{a_1}\boldsymbol{Z}^{b_1} \otimes \ldots \otimes \boldsymbol{X}^{a_n}\boldsymbol{Z}^{b_n}$. E.g.

1. numpy.array([0, 0, 1, 1, 0, 1, 1, 0]) = [0 0 1 1 I 0 1 1 0] = IZYX
2. numpy.array([0, 1, 0, 1, 1, 1, 0, 0]) = [0 1 0 1 I 1 1 0 0] = ZYIX

Symplectic form denoting action of Pauli operators on 4 qubits. Library contained in *qecsim.paulitools*

**Flowchart of Simulation :**

- We generate random Pauli error by passing code and *error_probability* to *qecsim.model.ErrorModel.generate()*.
- Run the required stabilizer code by *qecsim.model.StabilizerCode*, which contains built in stabilizer codes.
- $\boldsymbol{A} \circ \boldsymbol{B}$
- evaluate syndrome as $error \odot code.stabilizers^{T}$
- generate step_measurement_errors[t] as syndrome bit-flips.
- Finally calculate, syndrome[t] as step_measurement_errors[t-1] $\oplus$ step_syndromes[t] $\oplus$ step_measurement_errors[t].
- evaluate error as $\oplus$ step_errors (all t values)
- All this is internaly resolved decoding by passing code, time_steps and syndrome to qecsim.model.DecoderFTP.decode_ftp().
- Finally, define recovered as decoding $\oplus$ error

# References


[1] Dan Browne and Hans Briegel. "One-way Quantum Computation". In: *Quantum information: From foundations to quantum technology applications* (2016), pp. 449–473.

[2] Daniel Gottesman and Isaac L Chuang. "Quantum teleportation is a universal computational primitive". In: *arXiv preprint quant-ph/9908010* (1999).

[3] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[4] *QECSIM Library Package Documentation*. https://qecsim.github.io/overview.html. Accessed: 2022-07-02.

[5] Robert Raussendorf and Hans J Briegel. "A one-way quantum computer". In: *Physical review letters* 86.22 (2001), p. 5188.

[6] Robert Raussendorf, Daniel E Browne, and Hans J Briegel. "Measurement-based quantum computation on cluster states". In: *Physical review A* 68.2 (2003), p. 022312.