Quantum Computing

Chapter 03: Quantum Algorithm and State Preparation

Prachya Boonkwan

Version: January 14, 2026

Sirindhorn International Institute of Technology
Thammasat University, Thailand

Who? Me?

- Nickname: Arm (P'/N' Arm, etc.)
- Born: Aug 1981
- Work
  - Researcher at NECTEC 2005-2024
  - Lecturer at SIIT, Thammasat University 2025-now
- Education
  - B.Eng & M.Eng in Computer Engineering, Kasetsart University, Thailand
  - Obtained Ministry of Science and Technology Scholarship of Thailand in early 2008
  - Did a PhD in Informatics (AI & Computational Linguistics) at University of Edinburgh, UK from 2008 to 2013

Table of Contents

1. Introduction

2. Clifford Set/1

3. Clifford Set/2

4. Clifford Set/3

5. Circuit Visualization

6. Conclusion

Introduction

Quantum Algorithm

- Quantum algorithm is a circuit of quantum logic gates for manipulating qubit-based information to achieve desired computation
  1. Preparation: convert input into quantum states, superpositions, and entanglement
  2. Computation: via state manipulation, conditions, loops, reflections, and phase estimation
  3. Measurement: measure the probability distribution for each quantum state

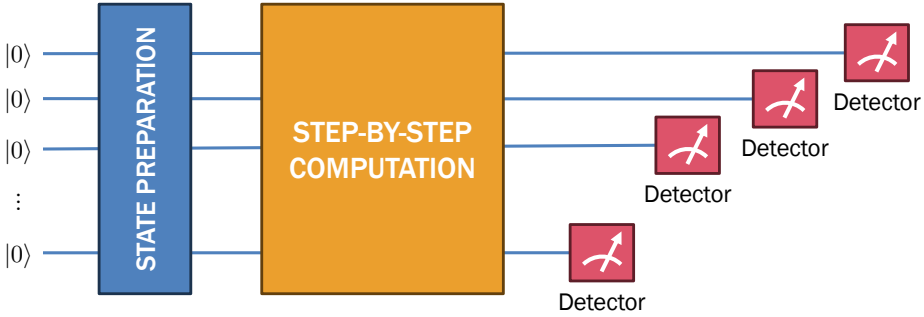


Figure 1: Overview of quantum algorithm

PennyLane

- Popular library for quantum computing in Python developed by Xanadu
- Features:
  - Easy to use with low learning curve
  - Support for CPU-based and GPU-based quantum simulation
  - Plugins for actual QPUs e.g. IBM Qiskit, Amazon Braket, and Xanadu
  - Support for automatic JIT compilation of quantum algorithms in the entire quantum-classical workflow
- The URL is `https://pennylane.ai/`

- Basic installation

```
pip install pennylane
```

- On CUDA-enabled machine

```
pip install custatevec_cu12
pip install pennylane-lightning-gpu
```

- On CUDA Tensor Networks

```
pip install cutensornet-cu12
pip install pennylane-lightning-tensor
```

- Docker

```
docker pull pennylaneai/pennylane:latest-
    lightning-qubit
```

Example Code

```
import pennylane as qml                # Quantum computing
from pennylane import numpy as np      # Classical computing

# Quantum device with two wires (i.e. variables)
dev = qml.device('default.qubit', wires=[0, 1])

# This is our simple quantum algorithm
@qml.qnode(dev)
def circuit(theta: float):

    # State preparation
    qml.H(wires=[0])             # Hadamard gate on qubit-0

    # Computation
    qml.RY(theta, wires=[0])     # Rotate qubit-0 on Axis-Y by theta
    qml.CNOT(wires=[0,1])        # Controlled NOT on qubit-1 with control qubit-0

    # Measurement
    return qml.state()

# Let's run our quantum algorithm
print(circuit(np.pi / 6))
```

Code Skeleton

Header

```
import pennylane as qml
from pennylane import numpy as np
```

Quantum device

```
dev = qml.device('default.qubit', wires=\ldots)
```

Several quantum devices are also available.

- `default.qubit`: Simple simulator
- `default.mixed`: Mixed-state simulator
- `lightning.qubit`: C++-based simulator
- `default.tensor`: Tensor Networks

Quantum algorithm = quantum node

- Preparation = variable initialization
- Computation = sequence of logic gates
- Measurement = collapsing quantum states to a bitstring

```
@qml.qnode(dev)
def circuit(args):
    # State preparation
    ......

    # Computation
    ......

    # Measurement
    return ......
```

Clifford Set/1

Clifford Set of Quantum Logic Gates

- A set of basic quantum logic gates
  - Fault-tolerant: Hardware implementation of these gates can significantly tolerate noise and quantum decoherence
  - Self-inversible: Applying a quantum operator twice on a mixed state results in the same mixed state
  - Efficiently simulatable: They can be emulated in polynomial time by probabilistic classical computers
- The main purpose of the Clifford set is state preparation
- A quantum algorithm consists of only Clifford gates is sometimes called stabilizer circuit due to its stability
- In this chapter, we will divide them into three levels

Level 1	Level 2	Level 3
NOT	Pauli X	Swap
Hadamard	Pauli Y	Controlled Z
Phase shift	Pauli Z	
Controlled NOT		

Table 1: Clifford set of quantum logic gates

Level 1: NOT Gate

- NOT gate converts $|0\rangle$ to $|1\rangle$ and vice versa, similar to logical NOT

$$\begin{aligned} \mathbf{X} &= |\overline{x}\rangle\langle x| \\ &= |1\rangle\langle 0| + |0\rangle\langle 1| \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

- $\mathbf{X}$ is self-inversible

$$\mathbf{X}\mathbf{X}|x\rangle = |x\rangle$$

- $\mathbf{X}$ is often used for preparing $|1\rangle$
- `qml.X(wires=…)`

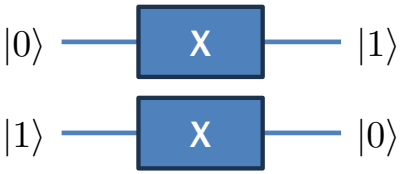


Figure 2: NOT gate

Level 1: Hadamard Gate

- Hadamard gate converts pure states to superpositions and vice versa

$$\begin{aligned} \mathbf{H} &= |+\rangle\langle 0| + |-\rangle\langle 1| \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{aligned}$$

- $\mathbf{H}$ is self-inversible

$$\mathbf{H}\mathbf{H}|x\rangle = |x\rangle$$

- Hadamard gate is often used for preparing a superposition
- `qml.H(wires=…)`

- Let's define the Hadamard bases

$$\begin{aligned} |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ |-\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

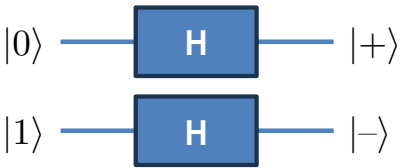


Figure 3: Hadamard gate

Level 1: Phase Shift Gate

- Phase shift gate converts real numbers to imaginary numbers and vice versa

$$\begin{aligned} \mathbf{S} &= |0\rangle\langle 0| + (i\,|1\rangle)\langle 1| \\ &= \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \end{aligned}$$

- **SS** is equivalent to sign flipping

$$\begin{aligned} \mathbf{SS}\,|0\rangle &= |0\rangle \\ \mathbf{SS}\,|1\rangle &= -|1\rangle \end{aligned}$$

- `qml.S(wires=…)`

- Let's define the Y bases

$$\begin{aligned} |+i\rangle &= \frac{|0\rangle + i\,|1\rangle}{\sqrt{2}} \\ |-i\rangle &= \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}} \end{aligned}$$

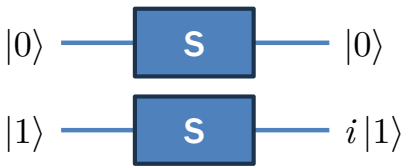


Figure 4: Phase shift gate

Level 1: Controlled NOT

- Controlled NOT is a condition tester, where the second qubit is bit-flipped if the first qubit is $|1\rangle$

$$\begin{aligned}\textbf{CNOT} &= |0,x\rangle\langle 0,x| + |1,\overline{x}\rangle\langle 1,x| \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}\end{aligned}$$

- **CNOT** is also self-inversible

$$\textbf{CNOT}\,\textbf{CNOT}\,|x,y\rangle = |x,y\rangle$$

- `qml.CNOT(wires=…)`

- **CNOT** is equivalent to logical XOR $\oplus$

$$\textbf{CNOT} = |x,(x \oplus y)\rangle\langle x,y|$$

- **CNOT** combined with **H** creates various forms of entanglement

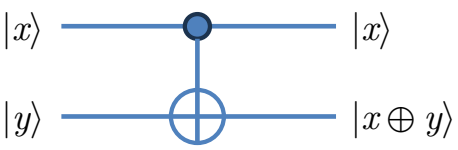


Figure 5: Controlled NOT gate

Summary of Level 1

Gates	Descriptions	Symbols = Formulae
NOT gate	Bit-flip the input qubit, similar to the logical NOT.	$\mathbf{X} = \|\overline{x}\rangle\langle x\|$
Hadamard gate	Convert pure $\leftrightarrow$ superposition on the input qubit.	$\mathbf{H} = \|+\rangle\langle 0\| + \|-\rangle\langle 1\|$
Phase shift gate	Convert real $\leftrightarrow$ imaginary if the input qubit is 1.	$\mathbf{S} = \|0\rangle\langle 0\| + (i\|1\rangle)\langle 1\|$
CNOT gate	Bit-flip the second qubit if the first one is 1. Create an entangled state when used with the Hadamard gate.	$\mathbf{CNOT} = \|0,x\rangle\langle 0,x\| + \|1,\overline{x}\rangle\langle 1,x\|$

Table 2: Basic Clifford gates

Special Unitary Group

- Special unitary group: $\mathbf{SU}(n)$ is the set of $n \times n$ complex matrices that are unitary and special
- Formal defintion:

$$\mathbf{SU}(n) = \{\mathbf{U} \in \mathbb{C}^{n \times n} | \mathbf{U}\mathbf{U}^* = \mathbf{I} \text{ and } \det(\mathbf{U}) = 1\}$$

- Why mentioning this?
  - Unitary $\mathbf{U}\mathbf{U}^* = \mathbf{I}$: All of these matrices are quantum operators
  - Special $\det(\mathbf{U}) = 1$: All of these matrices do not change the global phase

Quantum Gates	Groups
NOT gate	$\mathbf{SU}(2)$
Hadamard gate	$\mathbf{SU}(2)$
Phase shift gate	$\mathbf{SU}(2)$
CNOT gate	$\mathbf{SU}(4)$

Table 3: Clifford set and their SU groups

Quantum Algorithm: Example 1

- Step 0: The initial state is always $|0\rangle^{\otimes N}$.

$$\left|s^{(0)}\right\rangle = |0\rangle \otimes |0\rangle$$

- Step 1: Apply the Hadamard gate on the first qubit.

$$\begin{aligned}\left|s^{(1)}\right\rangle &= (\mathbf{H}|0\rangle) \otimes |0\rangle \\ &= |+\rangle \otimes |0\rangle\end{aligned}$$

- Step 2: Applying the NOT gate on the second qubit.

$$\begin{aligned}\left|s^{(2)}\right\rangle &= |+\rangle \otimes (\mathbf{X}|0\rangle) \\ &= |+\rangle \otimes |1\rangle\end{aligned}$$

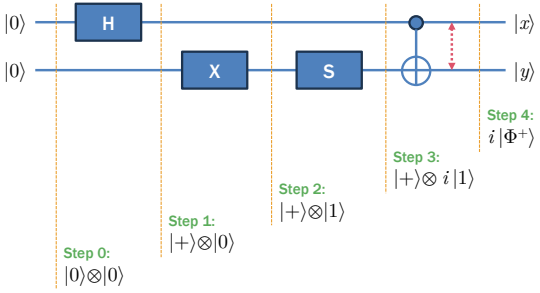


Figure 6: Just a quantum algorithm

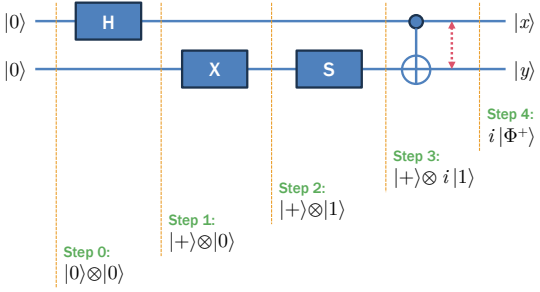
Quantum Algorithm: Example 1

- Step 3: Applying the phase shift gate on the second qubit $|1\rangle$ yields $i\,|1\rangle$.

$$\begin{aligned}\left|s^{(3)}\right\rangle &= |+\rangle \otimes (\mathbf{S}\,|1\rangle) \\ &= |+\rangle \otimes i\,|1\rangle\end{aligned}$$

- Step 4: To apply CNOT, we realize the tensor product of its inputs.

$$\begin{aligned}|+\rangle \otimes i\,|1\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes i\,|1\rangle \\ &= \frac{i}{\sqrt{2}}\left(|01\rangle + |11\rangle\right)\end{aligned}$$

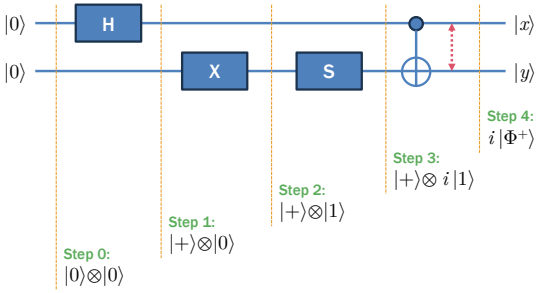


Quantum Algorithm: Example 1

- Step 4 (cont'd): Applying CNOT on $\frac{i}{\sqrt{2}}(|01\rangle + |11\rangle)$ yields

$$\begin{aligned} \left|s^{(4)}\right\rangle &= \text{CNOT}\left(\frac{i}{\sqrt{2}}(|01\rangle + |11\rangle)\right) \\ &= \frac{i}{\sqrt{2}}(|01\rangle + |10\rangle) \\ &= i\,|\Phi^+\rangle \end{aligned}$$

Note that the output now becomes an entangled state. The entanglement is denoted with a bidirectional arrow between the two qubits.



Example 1 in PennyLane

```
import pennylane as qml
from pennylane import numpy as np

dev = qml.device('default.qubit', wires=[0, 1])

@qml.qnode(dev)
def circuit():
    # State preparation
    qml.H(wires=[0])
    qml.X(wires=[1])
    qml.S(wires=[1])
    qml.CNOT(wires=[0, 1])
    # Measurement
    return qml.state()

print(circuit())
```

Result: $i\,|\Phi^+\rangle = \frac{i}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) = \begin{bmatrix} 0 & \frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \end{bmatrix}^\top$

```
[0.+0.j    0.+0.707j    0.+0.707j    0.+0.j]
```

Hadamard Gate and For-Loop

- Hadamard gate can be used as a for-loop

$$
\begin{aligned}
\left|s^{(0)}\right\rangle &= |000\rangle \\
\left|s^{(1)}\right\rangle &= (\mathbf{H}|0\rangle) \otimes |00\rangle \\
&= |+\rangle \otimes |00\rangle \\
\left|s^{(2)}\right\rangle &= |+\rangle \otimes (\mathbf{H}|0\rangle) \otimes |0\rangle \\
&= |+\rangle \otimes |+\rangle \otimes |0\rangle \\
\left|s^{(3)}\right\rangle &= |+\rangle \otimes |+\rangle \otimes (\mathbf{H}|0\rangle) \\
&= |+\rangle \otimes |+\rangle \otimes |+\rangle \\
\left|s^{(4)}\right\rangle &= \mathbf{U}\left(|+\rangle \otimes |+\rangle \otimes |+\rangle\right) \\
&= \sum_{k=0}^{7} \mathbf{U}\,|\mathbb{B}_3(k)\rangle
\end{aligned}
$$

- $\mathbf{H}^{\otimes N}|0\rangle^{\otimes N}$ is equivalent to 2^N iterations in classical computing

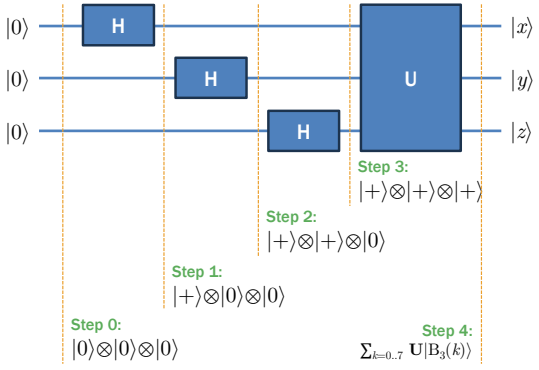


Figure 7: Paralellism via Hadamard gates

Exercise 3.1: State Preparation

Design quantum algorithms that prepare the following states. Develop PennyLane code to verify each of them.

1. $|+\rangle$
2. $|01\rangle$
3. $|100\rangle$
4. $i\,|1\rangle$
5. $-|1\rangle$
6. $|-\rangle \otimes |+\rangle \otimes |-\rangle$
7. $|\Psi^+\rangle$
8. $|\Psi^-\rangle$
9. $|1010\rangle$
10. $|10\rangle \otimes |+\rangle \otimes |01\rangle$
11. $|\Phi^+\rangle$
12. $|\Phi^-\rangle$
13. $-|0\rangle$
14. $|01\rangle \otimes |\Psi^+\rangle$
15. $|\Psi^+\rangle \otimes |\Phi^-\rangle$
16. $(-|0\rangle) \otimes |1\rangle$
17. $|00\rangle \otimes |+\rangle \otimes |-\rangle$
18. $-|+\rangle$
19. $-|-\rangle$
20. $|0\rangle \otimes (-i\,|1\rangle)$
21. $|+\rangle \otimes |\Psi^+\rangle$
22. $(-i\,|+\rangle) \otimes |-\rangle$

Operator Augmentation

- If we have a quantum operator

$$\mathbf{Q} = \sum_{k=1}^{K} |\phi_k\rangle\langle\psi_k|$$

we can augment $\mathbf{Q}$ with trivial qubits by

$$\mathbf{I}^M \otimes \mathbf{Q} = \sum_{k=1}^{K} |x_1, \ldots, x_M, \phi_k\rangle\langle x_1, \ldots, x_M, \psi_k|$$

$$\mathbf{Q} \otimes \mathbf{I}^M = \sum_{k=1}^{K} |\phi_k, x_1, \ldots, x_M\rangle\langle \psi_k, x_1, \ldots, x_M|$$

- This is useful for accommodating all entangled qubits, some of which not involved with the operator

- Examples:

$$\begin{aligned} \mathbf{I} \otimes \mathbf{X} &= |x, \overline{y}\rangle\langle x, y| \\ \mathbf{X} \otimes \mathbf{I} &= |\overline{x}, y\rangle\langle x, y| \end{aligned}$$

$$\begin{aligned} \mathbf{I} \otimes \mathbf{CNOT} &= |x, y, y \oplus z\rangle\langle x, y, z| \\ \mathbf{CNOT} \otimes \mathbf{I} &= |x, x \oplus y, z\rangle\langle x, y, z| \end{aligned}$$

$$\begin{aligned} \mathbf{I}^2 \otimes \mathbf{H} &= |x, y, +\rangle\langle x, y, 0| \\ &\quad + |x, y, -\rangle\langle x, y, 1| \\ \mathbf{I} \otimes \mathbf{H} \otimes \mathbf{I} &= |x, +, y\rangle\langle x, 0, y| \\ &\quad + |x, -, y\rangle\langle x, 1, y| \end{aligned}$$

Operator Augmentation and Tensor Product

- Important property of tensor product

$$\left(\bigotimes_{k=1}^{N} \mathrm{A}_k\right)\left(\bigotimes_{k=1}^{N} \mathrm{B}_k\right) = \bigotimes_{k=1}^{N} \mathrm{A}_k\mathrm{B}_k$$

- Examples:

$$\begin{aligned}(\mathrm{X}\otimes\mathrm{Y})(\mathrm{I}\otimes\mathrm{Z}) &= \mathrm{XI}\otimes\mathrm{YZ}\\ (\mathrm{H}\otimes\mathrm{H})(\mathrm{X}\otimes\mathrm{Y})(\mathrm{Y}\otimes\mathrm{Z}) &= \mathrm{HXY}\otimes\mathrm{HYZ}\\ (\mathrm{H}\otimes\mathrm{H}\otimes\mathrm{X})(\mathrm{H}\otimes\mathrm{X}\otimes\mathrm{Y})(\mathrm{H}\otimes\mathrm{X}\otimes\mathrm{Z}) &= \mathrm{H}^3\otimes\mathrm{HX}^2\otimes\mathrm{XYZ}\end{aligned}$$

- This property is useful for Pauli decomposition (decomposing a large matrix into Pauli axes: X, Y, Z) and computing the Hamiltonian matrix (taught in Chapter 7 afterwards)

Quantum Algorithm: Example 2

- Step 0: The initial state is $|0\rangle^{\otimes N}$.

$$\left|s^{(0)}\right\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle$$

- Step 1: Apply the Hadamard gate on the first qubit.

$$\begin{aligned}\left|s^{(1)}\right\rangle &= (\mathbf{H}|0\rangle) \otimes |0\rangle \otimes |0\rangle \\ &= |+\rangle \otimes |0\rangle \otimes |0\rangle\end{aligned}$$

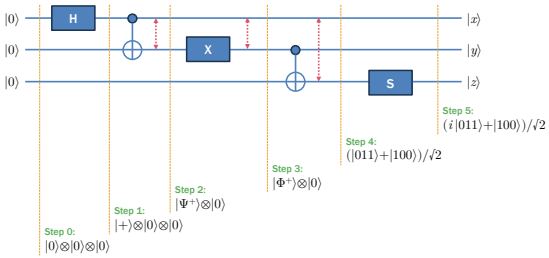


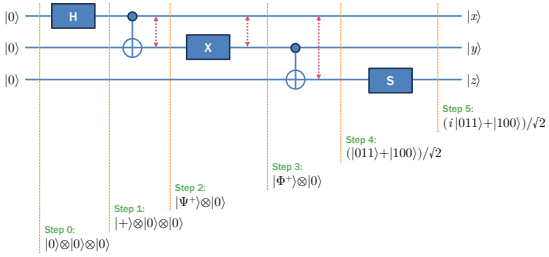
Figure 8: Another quantum algorithm

Quantum Algorithm: Example 2

- Step 2: Apply the CNOT gate on the first and second qubits.

$$\begin{aligned}\left|s^{(2)}\right\rangle &= (\mathrm{CNOT}\,(|+\rangle \otimes |0\rangle)) \otimes |0\rangle \\ &= \left(\mathrm{CNOT}\,\frac{|00\rangle + |10\rangle}{\sqrt{2}}\right) \otimes |0\rangle \\ &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \otimes |0\rangle\end{aligned}$$

The result is an entangled state, so a bidirectional arrow is shown.



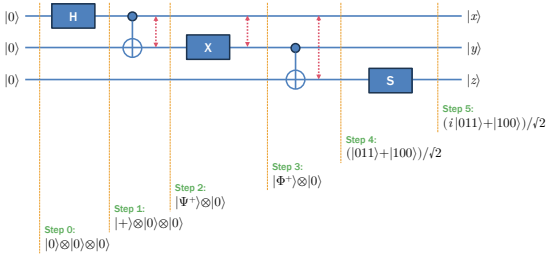
Quantum Algorithm: Example 2

- Step 3: Apply the NOT gate on the second qubit. Since it is entangled with the first qubit, we have to augment the NOT gate.

$$\mathrm{I} \otimes \mathrm{X} \quad = \quad |x, \overline{y}\rangle \langle x, y|$$

Note that the first qubit is preserved while the second one is flipped.

$$\begin{aligned} \left|s^{(3)}\right\rangle &= \left((\mathrm{I} \otimes \mathrm{X}) \frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) \otimes |0\rangle \\ &= \frac{|01\rangle + |10\rangle}{\sqrt{2}} \otimes |0\rangle \end{aligned}$$



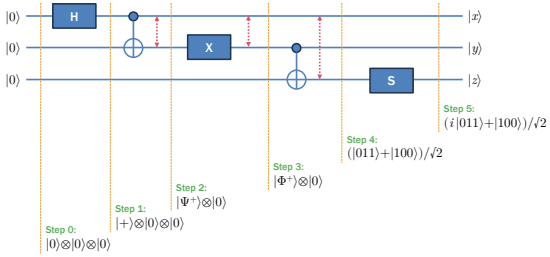
Quantum Algorithm: Example 2

- Step 4: Apply the CNOT gate on the second and third qubits. Since the second qubit is entangled with the first one, we augment the CNOT gate.

$$\mathrm{I} \otimes \mathrm{CNOT} \quad = \quad |x, y, y \oplus z\rangle \langle x, y, z|$$

Note that the first qubit is preserved.

$$\begin{aligned} \left|s^{(4)}\right\rangle &= (\mathrm{I} \otimes \mathrm{CNOT}) \left(\frac{|01\rangle + |10\rangle}{\sqrt{2}} \otimes |0\rangle \right) \\ &= (\mathrm{I} \otimes \mathrm{CNOT}) \frac{|010\rangle + |100\rangle}{\sqrt{2}} \\ &= \frac{|011\rangle + |100\rangle}{\sqrt{2}} \end{aligned}$$



Note that it results in another entangled state. That is why we mark it with a bidirectional arrow across the three qubits.

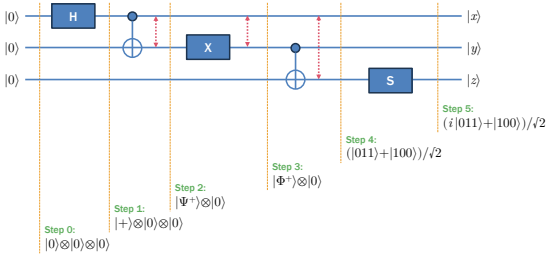
Quantum Algorithm: Example 2

- Step 5: Apply the phase shift gate on the third qubit. Since it is entangled with the first and second ones, we have to augment the gate.

$$\begin{aligned} \mathrm{I}^2 \otimes \mathrm{S} &= |x,y,0\rangle\langle x,y,0| \\ &\quad + i\,|x,y,1\rangle\langle x,y,1| \end{aligned}$$

Note that the first and second qubits are preserved.

$$\begin{aligned} \left|s^{(5)}\right\rangle &= (\mathrm{I}^2 \otimes \mathrm{S}) \left(\frac{|011\rangle + |100\rangle}{\sqrt{2}}\right) \\ &= \frac{i\,|011\rangle + |100\rangle}{\sqrt{2}} \end{aligned}$$



Example 2 in PennyLane

```
import pennylane as qml
from pennylane import numpy as np

dev = qml.device('default.qubit', wires=[0,1,2])

@qml.qnode(dev)
def circuit():
    # State preparation
    qml.H(wires=[0])
    qml.CNOT(wires=[0,1])
    qml.X(wires=[1])
    qml.CNOT(wires=[1,2])
    qml.S(wires=[2])

    # Measurement
    return qml.state()

print(circuit())
```

Result:

```
[0.   0.   0.   0.707j   0.707   0.   0.   0.]
```

Creating GHZ State

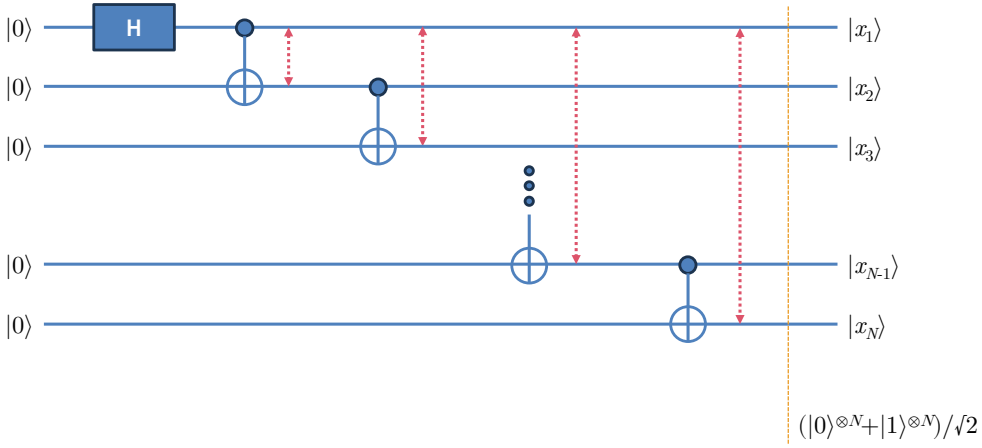


Figure 9: Hadamard gate and a series of CNOT can be used to create the GHZ state.

Exercise 3.2: Entanglement Preparation

Design quantum algorithms that prepare the following states. Develop PennyLane code to verify each of them.

1. $|\Psi^+\rangle$
2. $|\Phi^+\rangle$
3. $\frac{1}{\sqrt{2}}\left(|000\rangle + |111\rangle\right)$
4. $\frac{1}{\sqrt{2}}\left(|100\rangle + |011\rangle\right)$
5. $\frac{1}{\sqrt{2}}\left(|0011\rangle + |1100\rangle\right)$
6. $\frac{1}{\sqrt{2}}\left(|100\rangle - |011\rangle\right)$
7. $\frac{1}{\sqrt{2}}\left(|100\rangle - i\,|011\rangle\right)$
8. $\frac{1}{\sqrt{2}}\left(i\,|100\rangle - |011\rangle\right)$
9. $\frac{1}{\sqrt{2}}\left(i\,|101\rangle + |010\rangle\right) \otimes |\Psi^-\rangle$

Answer the following questions.

10. Let's define the adjoint shift gate

$$\mathbf{S}^* \quad = \quad |0\rangle\langle 0| - (i\,|1\rangle)\langle 1|$$

Show that $\mathbf{SSS}\,|x\rangle = \mathbf{S}^*\,|x\rangle$ for any quantum state $|x\rangle$.

11. Show that

$$\begin{aligned} \mathbf{I}^M \otimes \mathbf{U} &= \mathbf{I}_M \otimes \mathbf{U} \\ \mathbf{U} \otimes \mathbf{I}^M &= \mathbf{U} \otimes \mathbf{I}_M \end{aligned}$$

where $\mathbf{I}_M$ is the $M \times M$ identity matrix.

Clifford Set/2

Level 2: Pauli-X Gate

- Pauli gate rotates the qubit for phase π on one the axes: X, Y, and Z
- Pauli X flips the qubit

$$|0\rangle \mapsto |1\rangle \qquad |1\rangle \mapsto |0\rangle$$

which is equivalent to rotation on Axis X

$$\begin{aligned} \mathbf{X} &= |1\rangle\langle 0| + |0\rangle\langle 1| \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

- $\mathbf{X}$ is self-inversible: $\mathbf{X}\mathbf{X}|x\rangle = |x\rangle$
- `qml.X(wires=…)`

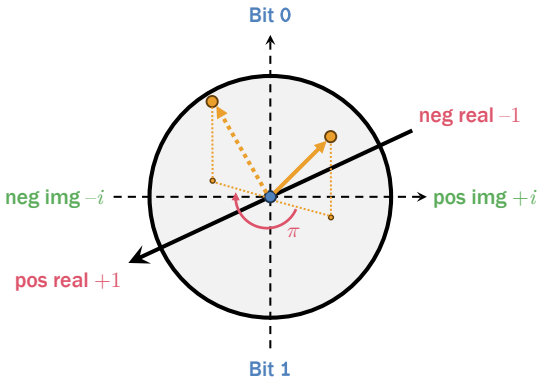


Figure 10: Pauli X gate

Level 2: Pauli-Y Gate

- Pauli Y flips the qubit, flips the sign, and flips the amplitude

$$|0\rangle \mapsto i\,|1\rangle \qquad |1\rangle \mapsto -i\,|0\rangle$$

which is equivalent to rotation on Axis Y

$$\begin{aligned} \mathbf{Y} &= (i\,|1\rangle)\langle 0| + (-i\,|0\rangle)\langle 1| \\ &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \end{aligned}$$

- $\mathbf{Y}$ is self-inverse: $\mathbf{Y}\mathbf{Y}|x\rangle = |x\rangle$
- `qml.Y(wires=…)`

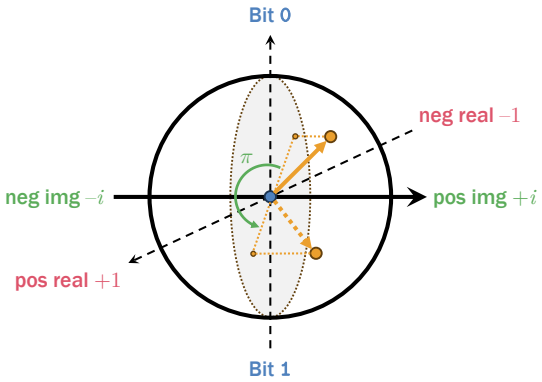


Figure 11: Pauli Y gate

Level 2: Pauli-Z Gate

- Pauli Z keeps the qubit and flips the sign

$$|0\rangle \mapsto |0\rangle \qquad |1\rangle \mapsto -|1\rangle$$

while is equivalent to rotation on Axis Z

$$\begin{aligned} \mathbf{Z} &= |0\rangle\langle 0| + (-|1\rangle)\langle 1| \\ &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned}$$

- $\mathbf{Z}$ is self-inversible: $\mathbf{Z}\mathbf{Z}|x\rangle = |x\rangle$
- `qml.Z(wires=…)`

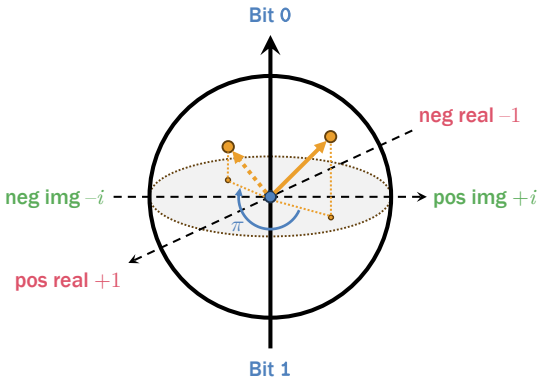


Figure 12: Pauli Z gate

Summary of Level 2

Gates	$\|0\rangle$	$\|1\rangle$	Flip qubit?	Flip sign?	Flip amplitude?
$\mathbf{X}$	$\|1\rangle$	$\|0\rangle$	Y	N	N
$\mathbf{Y}$	$i\|1\rangle$	$-i\|0\rangle$	Y	Y	Y
$\mathbf{Z}$	$\|0\rangle$	$-\|1\rangle$	N	Y	N

Table 4: Pauli gates. All of them belong to the $\mathbf{SU}(2)$ group.

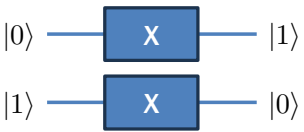


Figure 13: Pauli X

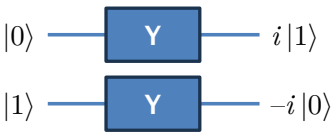


Figure 14: Pauli Y

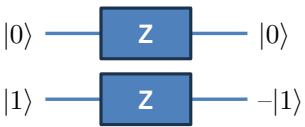


Figure 15: Pauli Z

Generalized Operator Augmentation

- For any quantum operator $\mathbf{Q}$, we can augment it with trivial qubits by

$$\mathbf{Q}^{(N)}_{p_1,\ldots,p_M} = \sum_{k=1}^{K} \left|x'_1,\ldots,x'_N\right\rangle \left\langle x_1,\ldots,x_N\right|$$

where the output mixed state is specified by the indices $p_1,\ldots,p_M$, i.e.

$$\left|x'_{p_1},\ldots,x'_{p_M}\right\rangle = \mathbf{Q}\left|x_{p_1},\ldots,x_{p_M}\right\rangle$$

while the remaining input qubits are unchanged, i.e. $x'_{p_j} = x_{p_j}$ for all indices $j \notin \{p_1,\ldots,p_M\}$

- Examples:

$$\begin{aligned}
\mathbf{X}^{(3)}_3 &= |x,y,1\rangle\langle x,y,0| \\
&\quad + |x,y,0\rangle\langle x,y,1| \\
\mathbf{Z}^{(3)}_2 &= |x,0,z\rangle\langle x,0,z| \\
&\quad + (-1)|x,1,z\rangle\langle x,1,z| \\
\mathbf{H}^{(3)}_1 &= |+,y,z\rangle\langle 0,y,z| \\
&\quad + |-,y,z\rangle\langle 1,y,z| \\
\mathbf{CNOT}^{(4)}_{1,3} &= |0,x,y,z\rangle\langle 0,x,y,z| \\
&\quad + |1,x,\overline{y},z\rangle\langle 1,x,y,z|
\end{aligned}$$

Quantum Algorithm: Example 3

- Step 0: The initial state is $|0\rangle^{\otimes N}$.

$$\left|s^{(0)}\right\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$$

- Step 1: We recognize two entangled states caused by the Hadamard and CNOT gates.

$$\left|s^{(1)}\right\rangle = \frac{1}{2}\left(|00\rangle + |11\rangle\right) \otimes \left(|00\rangle + |11\rangle\right)$$

- Step 2: We apply the X and Z gates on the first and latter pairs of qubits.

$$\begin{aligned}\left|s^{(2)}\right\rangle &= \left[\mathrm{I}|\mathrm{X}|\mathrm{I}|\mathrm{Z}\right]\left|s^{(1)}\right\rangle \\ &= \frac{1}{2}\left(|01\rangle + |11\rangle\right) \otimes \left(|00\rangle - |11\rangle\right)\end{aligned}$$

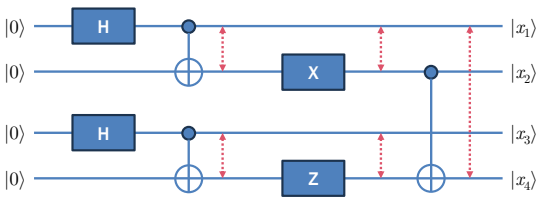


Figure 16: Yet another quantum algorithm

Quantum Algorithm: Example 3

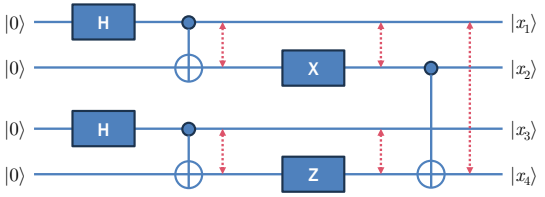
- Step 2 (cont'd): We expand the tensor product of $\left|s^{(2)}\right\rangle$

$$\left|s^{(2)}\right\rangle = \frac{1}{2}(|0100\rangle - |0111\rangle + |1100\rangle - |1111\rangle)$$

- Step 3: The CNOT gate for the 2nd and 4th qubits is simply a CNOT gate with the 1st and 3th qubits augmented to it.

$$\mathbf{CNOT}_{2,4}^{(4)} = |v, x, y, x \oplus z\rangle\langle v, x, y, z|$$

Observe that v and y are introduced as dummy variables.



- Step 3 (cont'd): Applying this gate on $\left|s^{(2)}\right\rangle$ yields

$$\mathbf{CNOT}_{2,4}^{(4)}\left|s^{(2)}\right\rangle = \frac{1}{2}(|0101\rangle - |0110\rangle + |1101\rangle - |1110\rangle)$$

which is still entangled.

Exercise 3.3: Pauli Gates

Design quantum algorithms that prepare the following states. Develop PennyLane code to verify each of them.

1. $|\Phi^+\rangle$
2. $\frac{i}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)$
3. $|\Psi^-\rangle$
4. $\frac{i}{\sqrt{2}}\left(-|00\rangle + |11\rangle\right)$ [Use $\mathbf{Y}$.]
5. $\frac{1}{\sqrt{2}}\left(|011\rangle + |100\rangle\right)$ [Use $\mathbf{Z}$.]
6. $\frac{1}{\sqrt{2}}\left(-|010\rangle - i\,|101\rangle\right)$ [Use $\mathbf{Y}$.]
7. $\frac{i}{\sqrt{2}}\left(|0100\rangle - |1011\rangle\right)$ [Use $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$.]
8. $\frac{i}{2}\left(|0101\rangle - |0110\rangle - |1000\rangle + |1011\rangle\right)$ [Use $\mathbf{Y}$ and $\mathbf{Z}$.]

Answer the following questions.

9. Show that $\mathbf{Z}|x\rangle = \mathbf{HXH}|x\rangle$. It means that $\mathbf{Z}$ can be emulated by the Hadamard gate and the NOT gate.
10. Show that $\mathbf{Y}|x\rangle = i\mathbf{XZ}|x\rangle$. It means that $\mathbf{Y}$ can be emulated by the Hadamard gate and the NOT gate.
11. Show that $-i\mathbf{XYZ} = \mathbf{I}$.
12. Show that for any single-qubit quantum operator $\mathbf{Q}$,

$$\mathbf{Q}_j^{(N)} \quad = \quad \mathbf{I}^{j-1} \otimes \mathbf{Q} \otimes \mathbf{I}^{N-j}$$

Clifford Set/3

Level 3: SWAP Gate

- Swap: We can trivially swap places of two qubits for ease of processing

$$\begin{aligned}\mathbf{SWAP} &= |y,x\rangle\langle x,y| \\ &= |00\rangle\langle 00| + |10\rangle\langle 01| \\ &\quad + |01\rangle\langle 10| + |11\rangle\langle 11| \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

- This is useful for rearranging the qubits before passing them to a quantum gate
- `qml.SWAP(wires=…)`

- Example:

$$\mathbf{SWAP}\left(\frac{|01\rangle - |10\rangle}{\sqrt{2}}\right) = \frac{|10\rangle - |01\rangle}{\sqrt{2}}$$

- **SWAP** is self-inversible

$$\mathbf{SWAP}\,\mathbf{SWAP}\,|x,y\rangle = |x,y\rangle$$

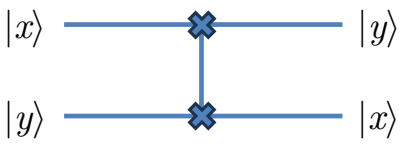


Figure 17: Swap gate

Level 3: Controlled Z Gate

- Controlled Z gate keeps the qubit and flips the sign if the control qubit is $|1\rangle$

$$\begin{aligned}\mathbf{CZ} &= |00\rangle\langle 00| + |01\rangle\langle 01| \\ &\quad + |10\rangle\langle 10| + (-1)|11\rangle\langle 11| \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}\end{aligned}$$

- $\mathbf{CZ}$ is self-inversible: $\mathbf{CZ}\,\mathbf{CZ}\,|x,y\rangle = |x,y\rangle$
- `qml.CZ(wires=…)`

- Example:

$$\mathbf{CZ}\left(\frac{|01\rangle - |10\rangle}{\sqrt{2}}\right) = \frac{|01\rangle - |11\rangle}{\sqrt{2}}$$

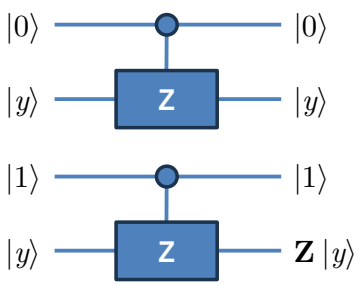


Figure 18: Controlled Z gate

Exercise 3.4: Swap and Controlled Z

Answer the following questions.

1. Let's define the Controlled Y gate $\mathbf{CY}$ from the following state mapping.

$$|0x\rangle \mapsto |0x\rangle \qquad |10\rangle \mapsto i\,|10\rangle \qquad |11\rangle \mapsto -i\,|11\rangle$$

   Explain why $\mathbf{CY}$ is not self-inversible.

2. Compute $\mathbf{SWAP}^{(4)}_{2,3}|s\rangle$, where

$$|s\rangle \;=\; \frac{1}{2}\left(|00\rangle + |11\rangle\right) \otimes \left(|01\rangle + |10\rangle\right)$$

3. Create $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes N} + (-1)^{N+1}|1\rangle^{\otimes N})$, where $N \geq 1$, using only $\mathbf{H}$ and $\mathbf{CZ}$.

4. Design a quantum operator $\mathbf{L}$ that rotates a mixed state of N qubits to the left for one bit. E.g. $\mathbf{L}|101100\rangle = |011001\rangle$. Then convert $\mathbf{L}$ to a quantum algorithm.

Circuit Visualization

Circuit Visualization

- We can visualize a quantum circuit in two formats: text and graphics
- Text format: `qml.draw(circuit)(params)`
- This instruction displays the given circuit after applying the given parameters on it

```
@qml.qnode(dev)
def circuit(theta: float):
    qml.H(wires=[0])
    qml.RY(theta, wires=[0])
    qml.CNOT(wires=[0,1])
    return qml.state()

print(qml.draw(circuit)(np.pi / 6))
```

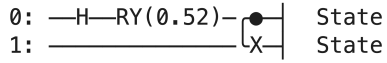


Figure 19: String representation of a quantum circuit

Circuit Visualization

- Graphical format: `qml.draw_mpl(circuit)(params)`

```
@qml.qnode(dev)
    def circuit(theta: float):
    qml.H(wires=[0])
    qml.RY(theta, wires=[0])
    qml.CNOT(wires=[0,1])
    return qml.state()

fig, ax = qml.draw_mpl(circuit)(np.pi / 6)
fig.savefig('circuit.png')
```

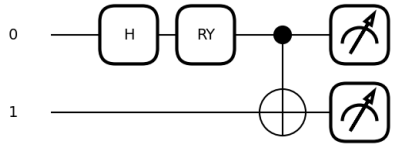


Figure 20: Graphical representation of a quantum circuit

Conclusion

Clifford Set of Quantum Gates

Gates	Symbols	From $\|0\rangle$	From $\|1\rangle$
NOT	$\mathbf{X}$	$\|1\rangle$	$\|0\rangle$
Hadamard	$\mathbf{H}$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$	$\frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$
Phase shift	$\mathbf{S}$	$\|0\rangle$	$i\|1\rangle$
Pauli-X	$\mathbf{X}$	$\|1\rangle$	$\|0\rangle$
Pauli-Y	$\mathbf{Y}$	$i\|1\rangle$	$-i\|0\rangle$
Pauli-Z	$\mathbf{Z}$	$\|0\rangle$	$-\|1\rangle$

Table 5: Single-qubit gates

Gates	Symbols	Descriptions
CNOT	$\mathbf{CNOT}$	$\|0,x\rangle \mapsto \|0,x\rangle$
		$\|1,x\rangle \mapsto \|1,\overline{x}\rangle$
Swap	$\mathbf{SWAP}$	$\|x,y\rangle \mapsto \|y,x\rangle$
CZ	$\mathbf{CZ}$	$\|0,x\rangle \mapsto \|0,x\rangle$
		$\|1,x\rangle \mapsto \|1,\mathbf{Z}\,x\rangle$

Figure 21: Two-qubit gates

Operator Augmentation

- If we have a quantum operator

$$\mathbf{Q} = \sum_{k=1}^{K} |\phi_k\rangle \langle \psi_k|$$

we can augment $\mathbf{Q}$ with trivial qubits by

$$\mathbf{I}^M \otimes \mathbf{Q} = \sum_{k=1}^{K} |x_1, \ldots, x_M, \phi_k\rangle \langle x_1, \ldots, x_M, \psi_k|$$

$$\mathbf{Q} \otimes \mathbf{I}^M = \sum_{k=1}^{K} |\phi_k, x_1, \ldots, x_M\rangle \langle \psi_k, x_1, \ldots, x_M|$$

- This is useful for accommodating all entangled qubits, some of which not involved with the operator

- For any quantum operator $\mathbf{Q}$, we can augment it with trivial qubits by

$$\mathbf{Q}^{(N)}_{p_1,\ldots,p_M} = \sum_{k=1}^{K} \left| x'_1, \ldots, x'_N \right\rangle \langle x_1, \ldots, x_N|$$

where the output mixed state is specified by the indices $p_1, \ldots, p_M$, i.e.

$$\left| x'_{p_1}, \ldots, x'_{p_M} \right\rangle = \mathbf{Q} \left| x_{p_1}, \ldots, x_{p_M} \right\rangle$$

while the remaining input qubits are unchanged, i.e. $x'_{p_j} = x_{p_j}$ for all indices $j \notin \{p_1, \ldots, p_M\}$

Questions?

Target Learning Outcomes/1

- Conceptual Understanding of Quantum Circuits
  1. Define the three primary stages of a quantum algorithm: state preparation, computation through gate manipulation, and measurement.
  2. Explain the characteristics of the Clifford gate set, including fault-tolerance, self-inversibility, and efficient classical simulability.
- Quantum Gate Mastery
  1. Apply Level 1 and 2 gates (Hadamard, Phase Shift, and Pauli X,Y,Z) to transform pure states into superpositions or modify quantum phases.
  2. Differentiate between the effects of Pauli X,Y, and Z gates in terms of qubit flipping, sign flipping, and amplitude changes.
  3. Utilize multi-qubit gates (CNOT, SWAP, CZ) to create entanglement or rearrange qubit order for processing.

Target Learning Outcomes/2

- Mathematical and Algebraic Application
  1. Calculate the state of a quantum system after a series of gate applications using tensor products and matrix multiplication.
  2. Perform operator augmentation to describe the action of a gate on specific indices within an N-qubit state.
  3. Prove gate equivalencies through algebraic manipulation.
- Computational Implementation
  1. Construct functional quantum circuits using the PennyLane library, including device initialization and QNode definition.
  2. Visualize quantum circuits in both text and graphical formats to verify algorithm structure.
  3. Develop algorithms to prepare specific quantum states, such as Bell states or GHZ states.