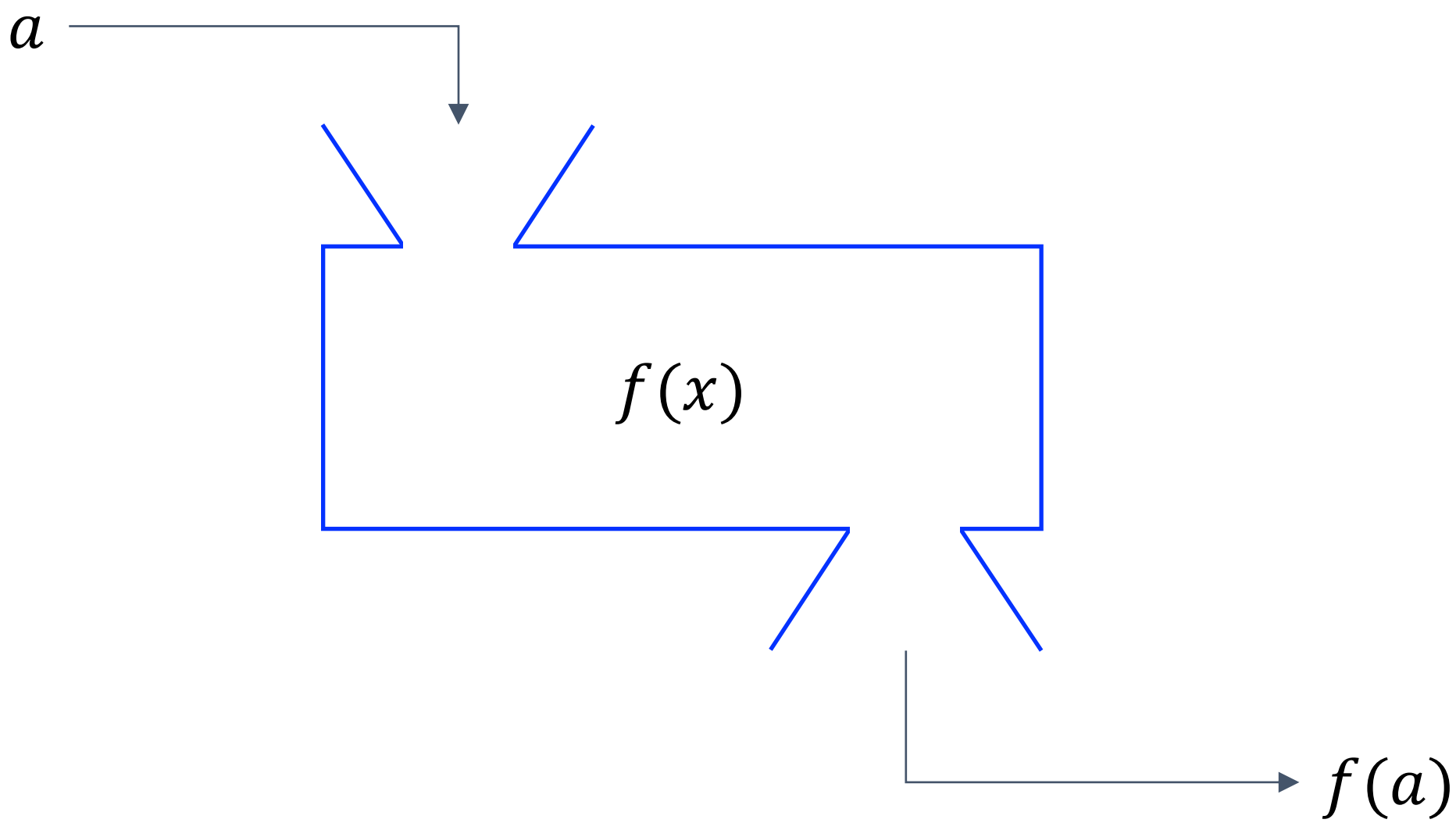
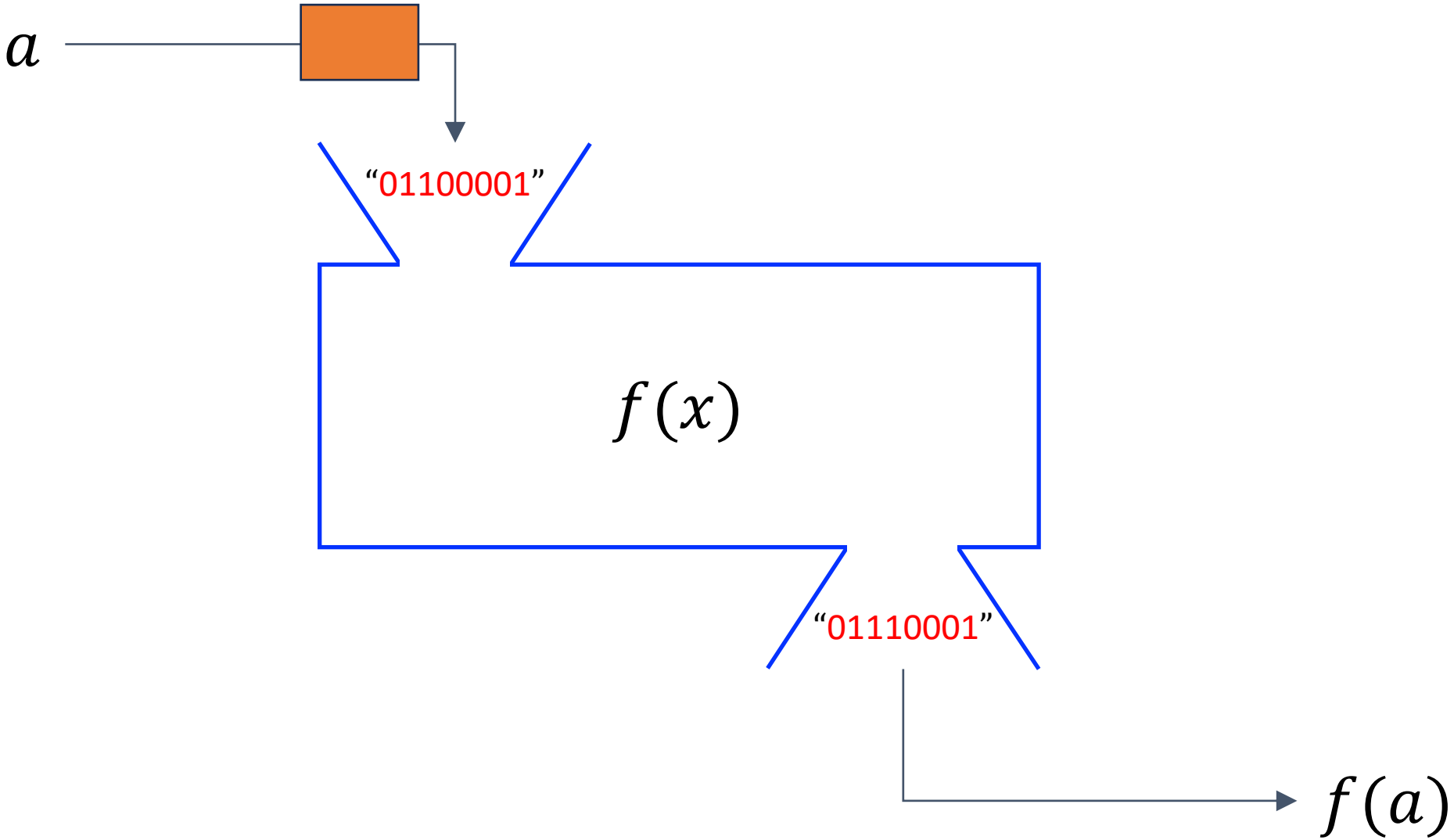
Quantum Data Embedding

2024.01.16

황용수 @ ETRI (양자컴퓨팅연구실)


a
$f(x)$
$f(a)$

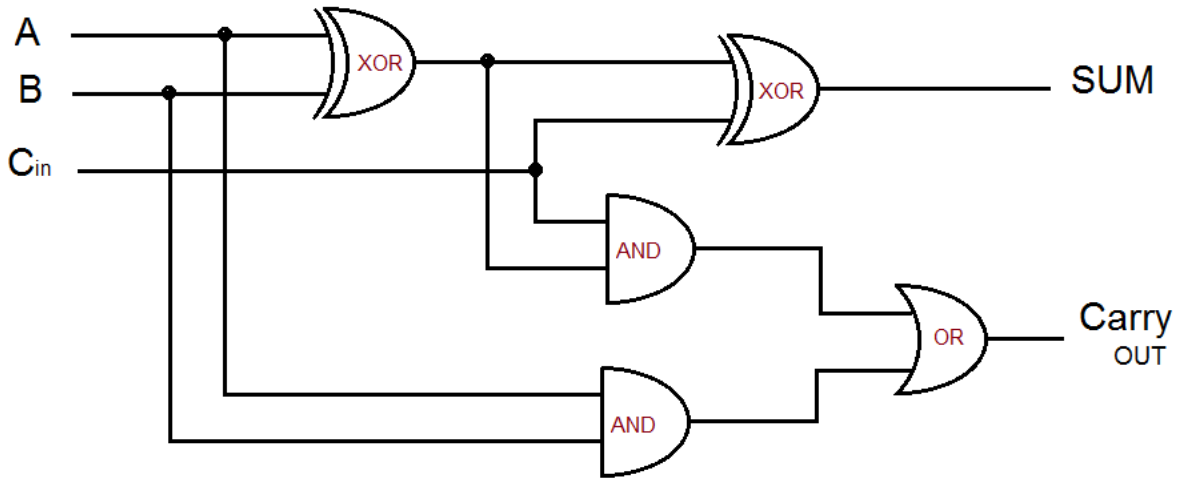

a
"01100001"
$f(x)$
"01110001"
$f(a)$

Outline

- Examples of (Quantum) Algorithm
- Preliminaries : Terminologies
- Quantum Data Embeddings
- Arbitrary Quantum State Preparation
- Robust Data Embedding

Quantum/Classical Algorithms

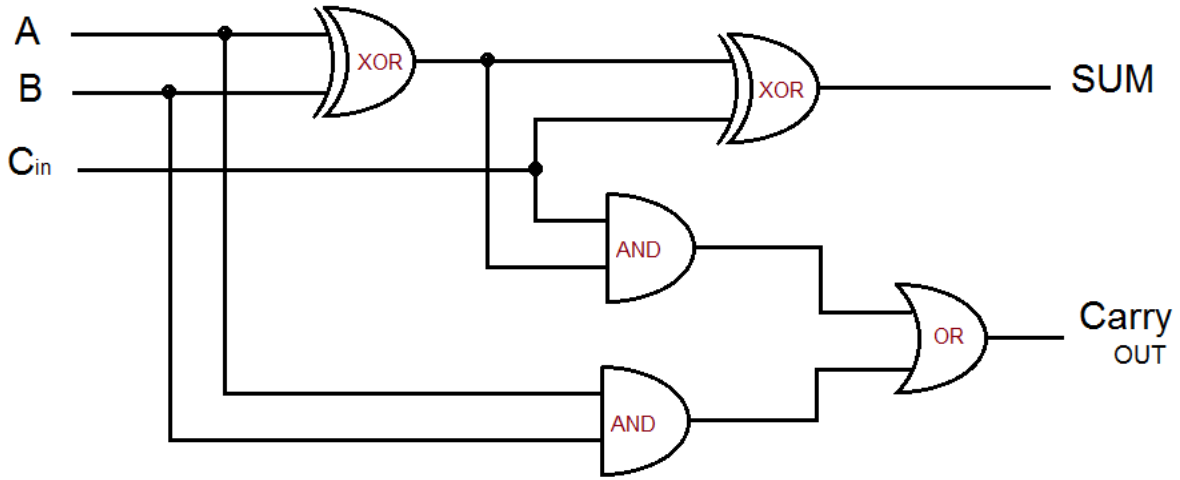
Classical Full Adder



A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

<Classical Full Adder : Circuit (L) and Truth table (R)>

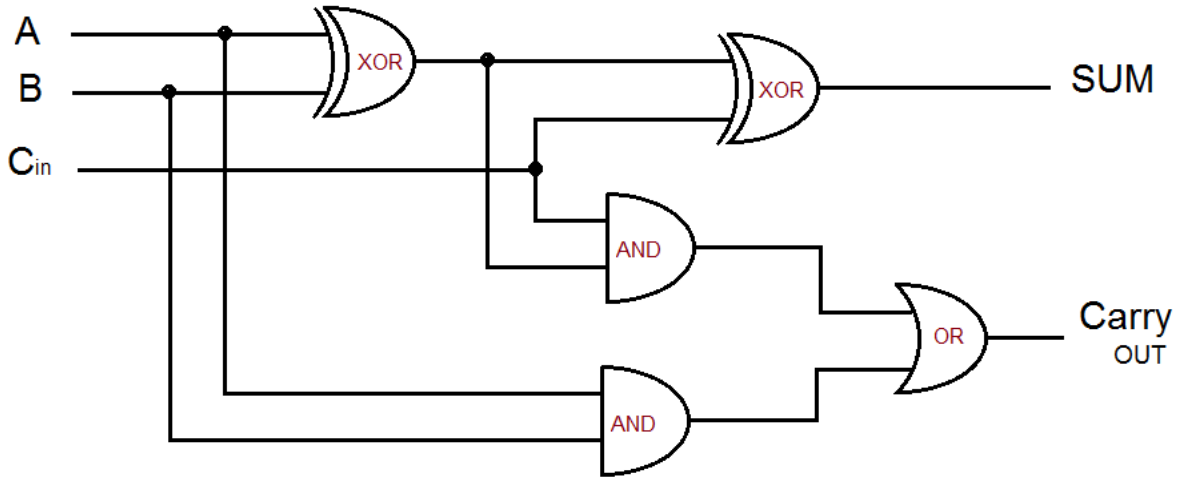
Classical Full Adder



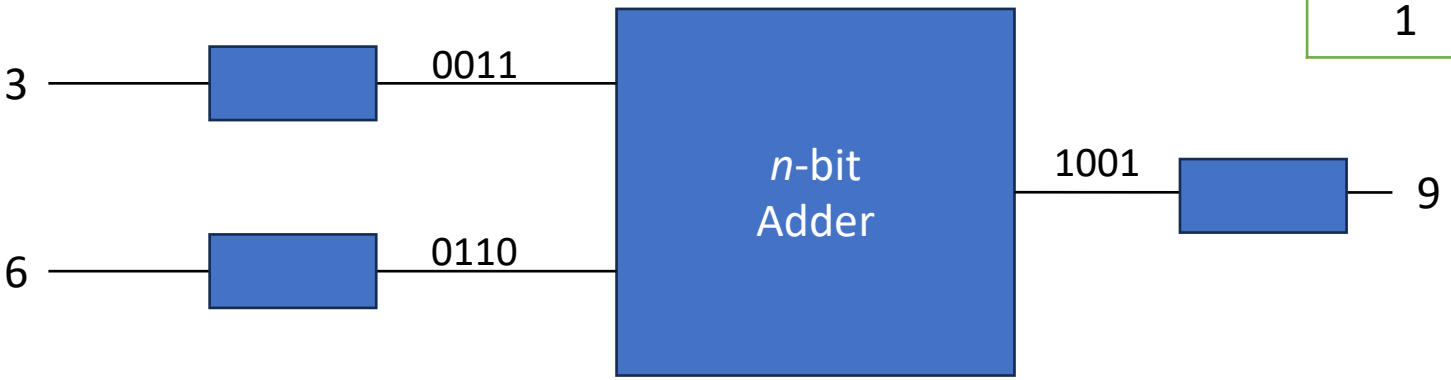
A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

3 + 6 = ?

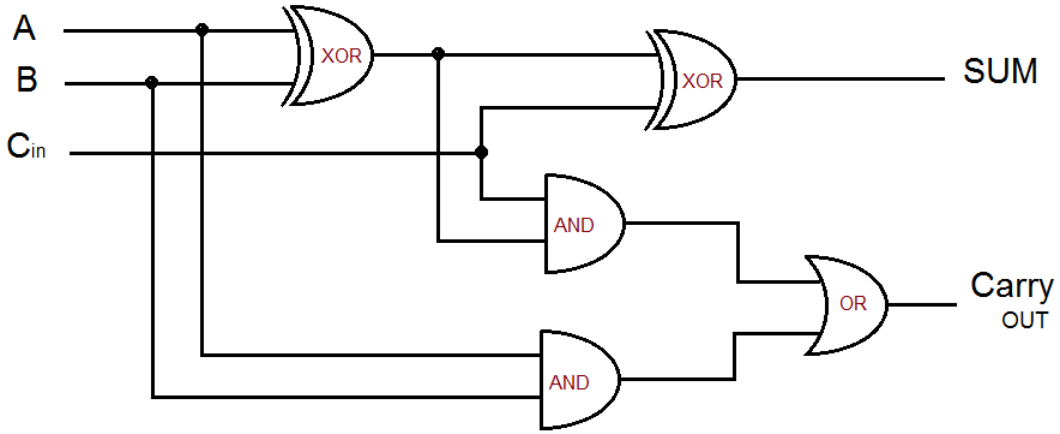
Classical Full Adder

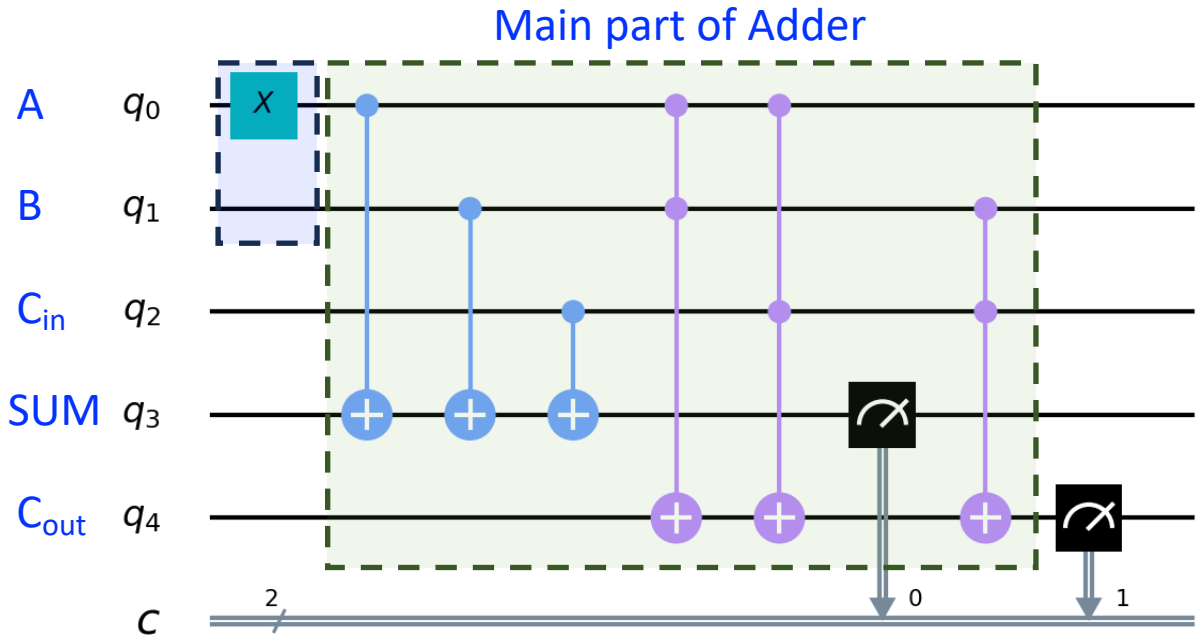


A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1



Quantum Adder





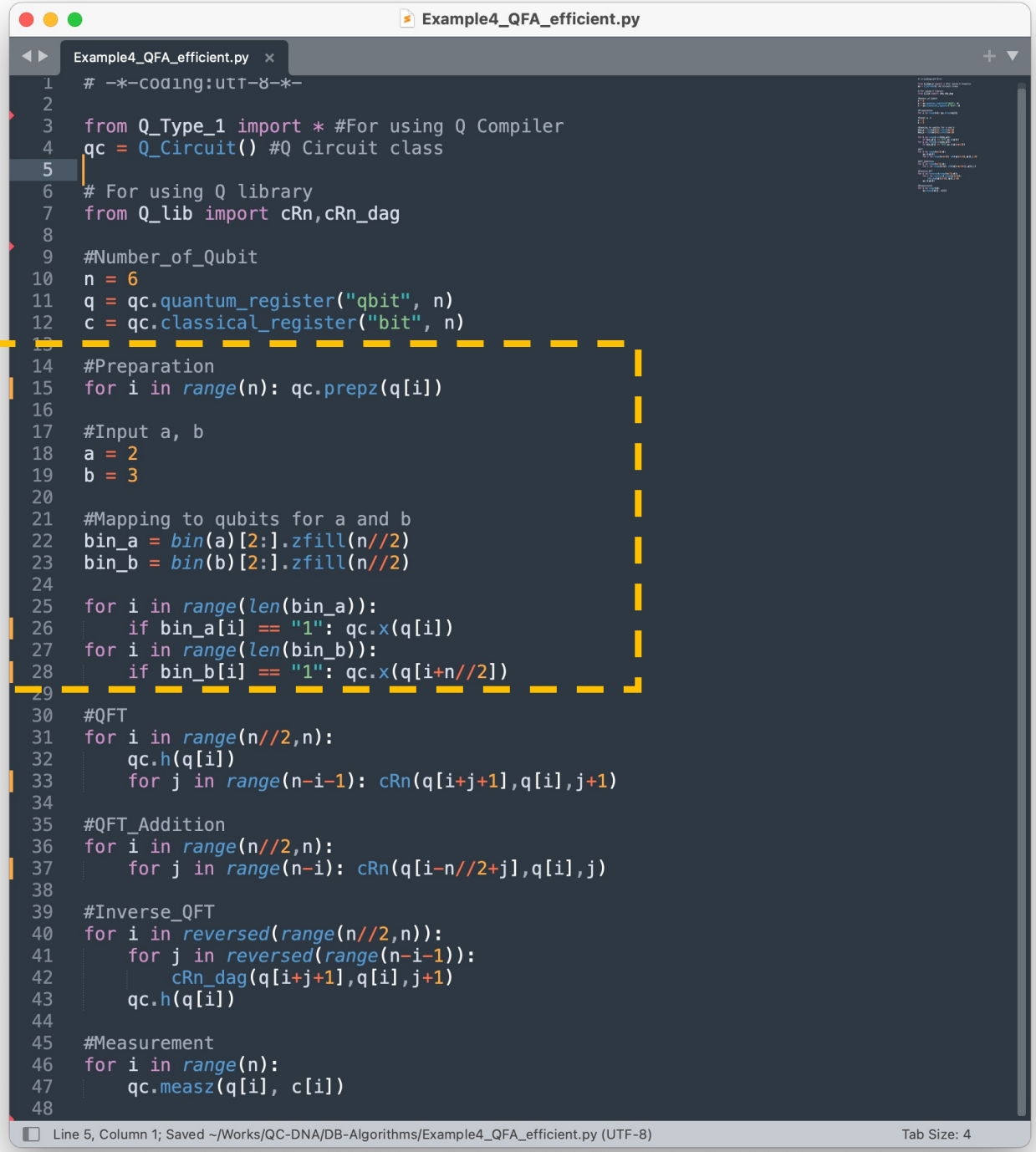
‘1’ + ‘0’ = ‘1’ $|100\rangle|00\rangle \longrightarrow |100\rangle|10\rangle$

Quantum Adder

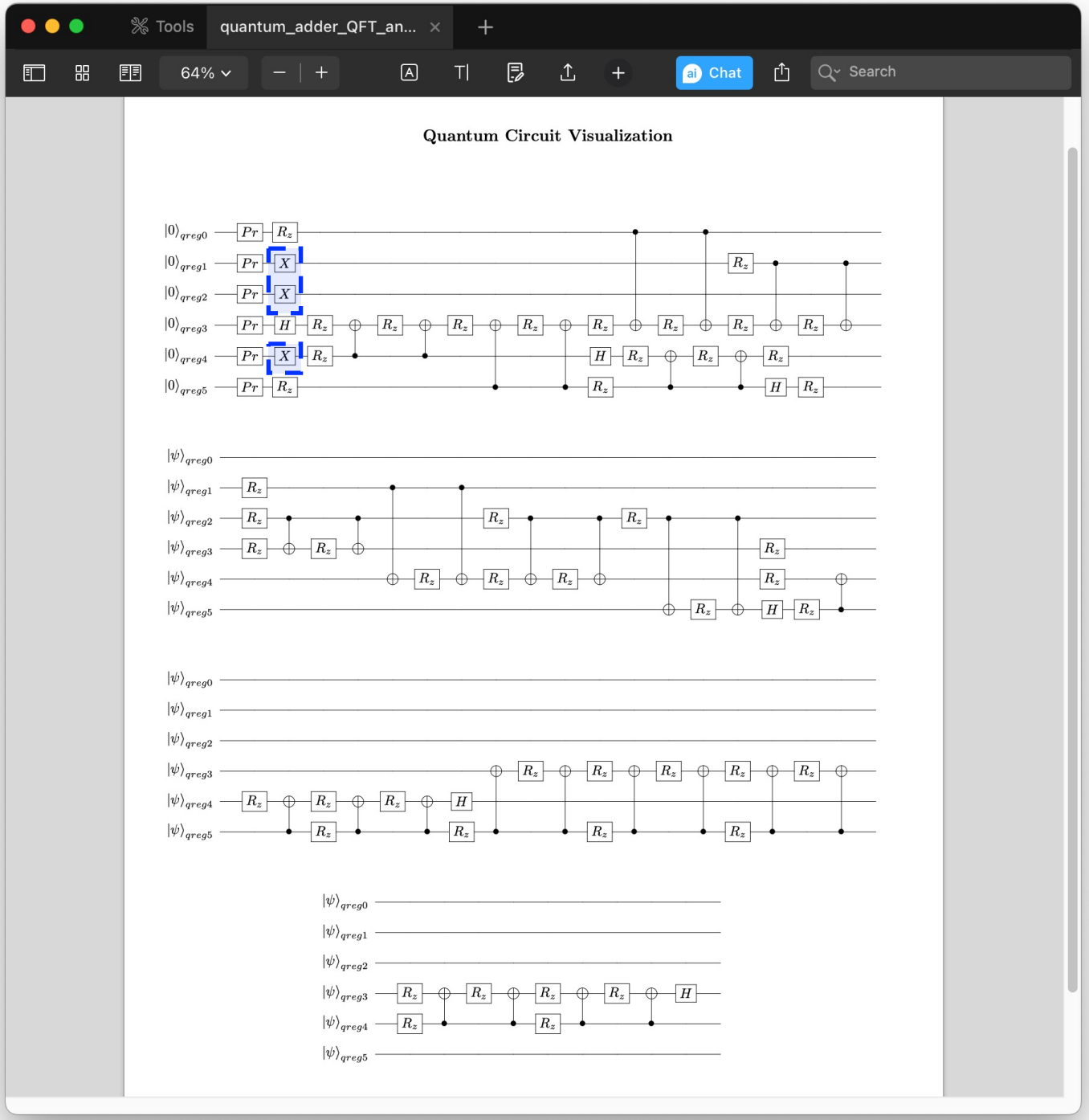
Part of processing input data

$2 \rightarrow 10 \rightarrow |10\rangle$

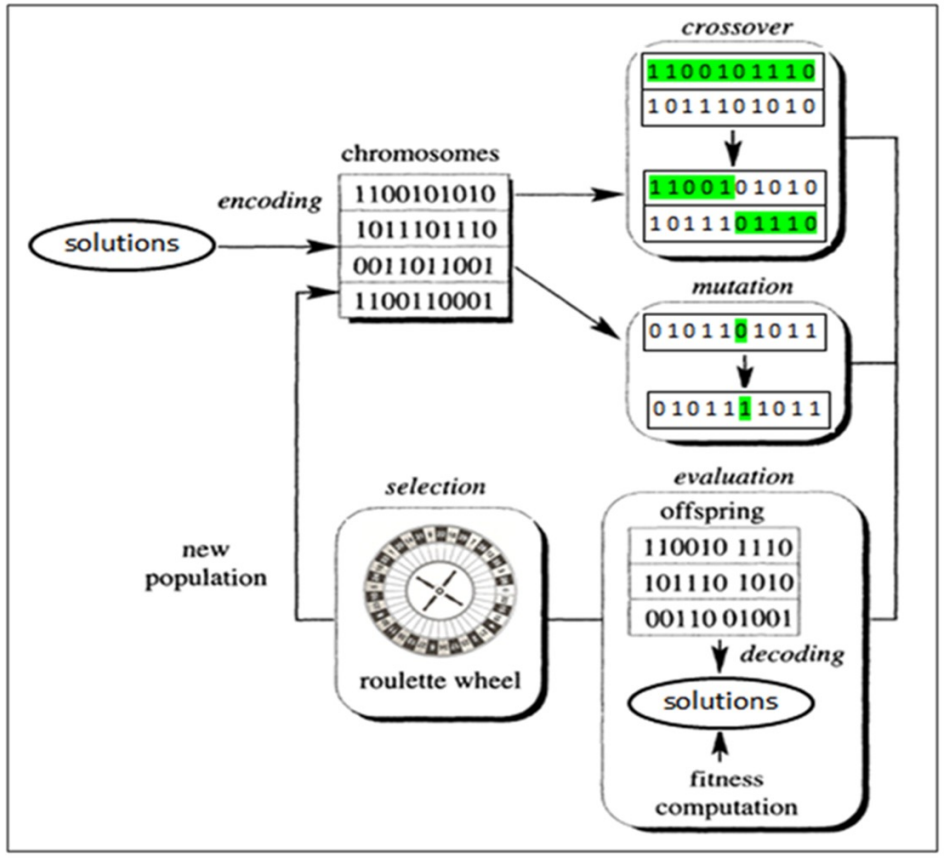
$3 \rightarrow 11 \rightarrow |11\rangle$



Quantum Adder



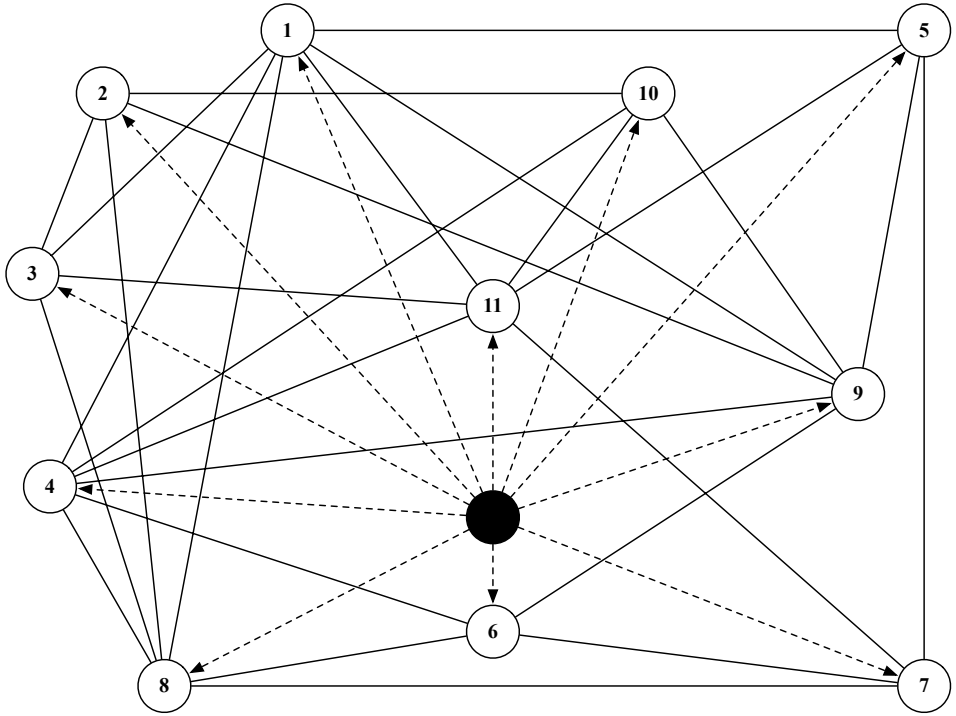
Classical Genetic Algorithm



Traditional structure of genetic algorithm (adapted from (Gen & Cheng 1997, p.3))

Graph for [[11, 1, 5]] Quantum Graph Code:

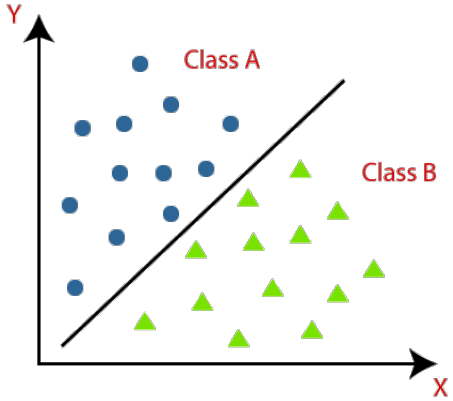
- ***Adjacency matrix*** of a simple mathematical graph is embedded into a **bit string sequence**



[YH et al., Graph for a quantum graph code, QIP 2016]

Classical Classifier

- **Classification problem** : A subclass of supervised learning problems in which a computer model is presented with labelled data and asked to learn some pattern.



- For binary classification, the input is a set of labelled feature vectors $\{(x_i, y_i)\}_{i=1}^{M}$ where $\mathcal{X}$ is a arbitrary set, $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \{0, 1\}$ is a binary label.
- Then, given this data, the goal of the learner is to output a rule $f: \mathcal{X} \to \{0, 1\}$ which accurately classifies the data and can be used to make predictions on new data

Quantum Classifier

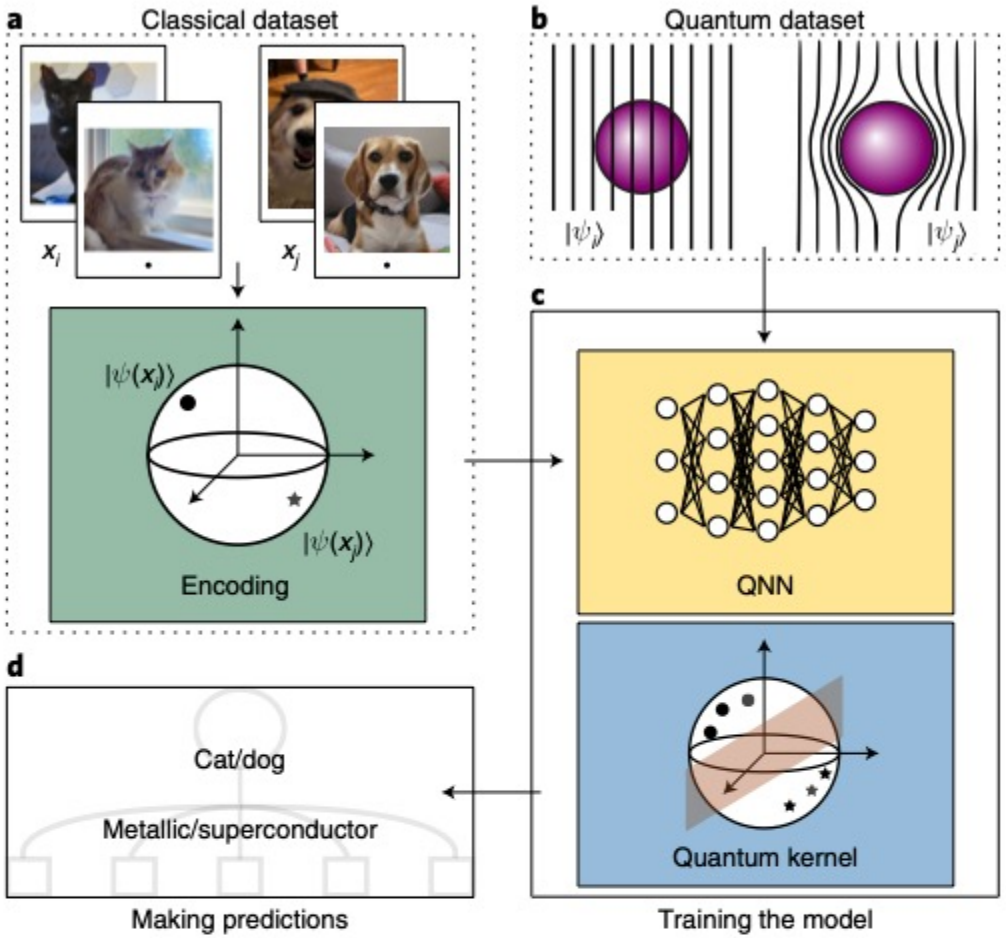


Fig. 3 | Classification with QML. a, The classical data x, that is, images of cats and images of dogs, is encoded into a Hilbert space via some map $x \rightarrow |\psi(x)\rangle$. Ideally, data from different classes (here represented by dots and stars) are mapped to different regions of the Hilbert space. **b**, Quantum data $|\psi\rangle$ can be directly analyzed on a quantum device. Here the dataset is composed of states representing metallic or superconducting systems. **c**, The dataset is used to train a QML model. Two common paradigms in QML are QNNs and quantum kernels, both of which allow for classification of either classical or quantum data. In kernel methods we fit a decision hyperplane that separates the classes. **d**, Once the model is trained, it can be used to make predictions.

Preliminaries : Terms

Quantum Register

- **Quantum register** :
  - An *n*-qubit quantum register is a quantum system comprising *n* pure qubits.

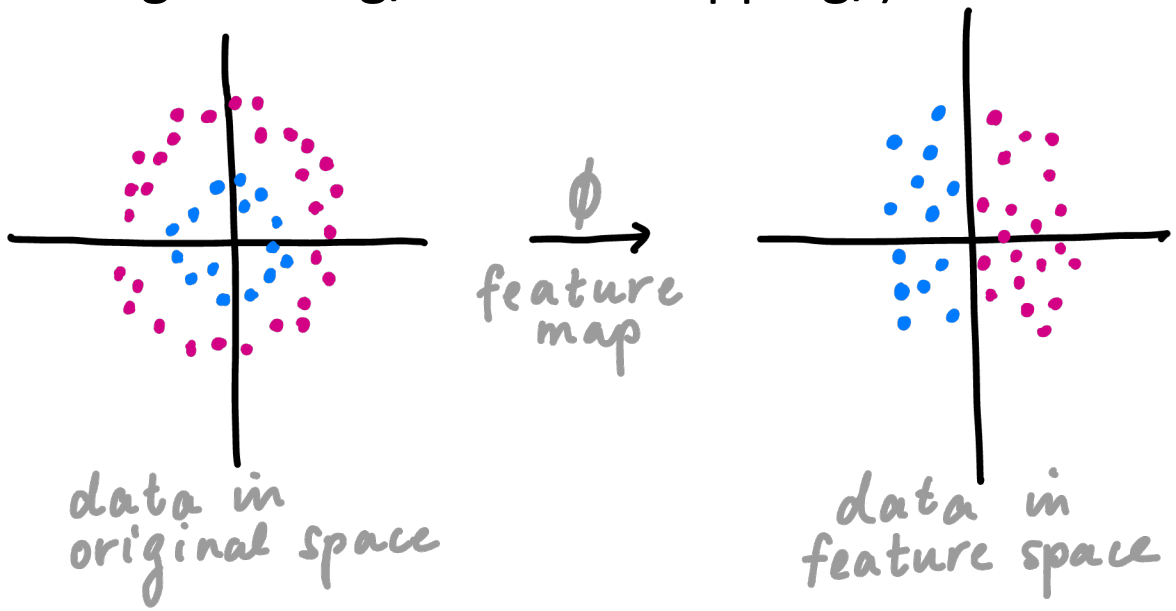
$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i|i\rangle = a_0|0\rangle + a_1|1\rangle + \cdots + a_{2^n-1}|2^n-1\rangle$$

  - If each quantum system (qubit) *s* is prepared in the state $|\psi_s\rangle$, the composite system can be described as a tensor product, $|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle$.
  - If the state is not separable, it cannot be represented as a tensor product), it is in an entangled state.

Quantum Data Embeddings

Feature Map

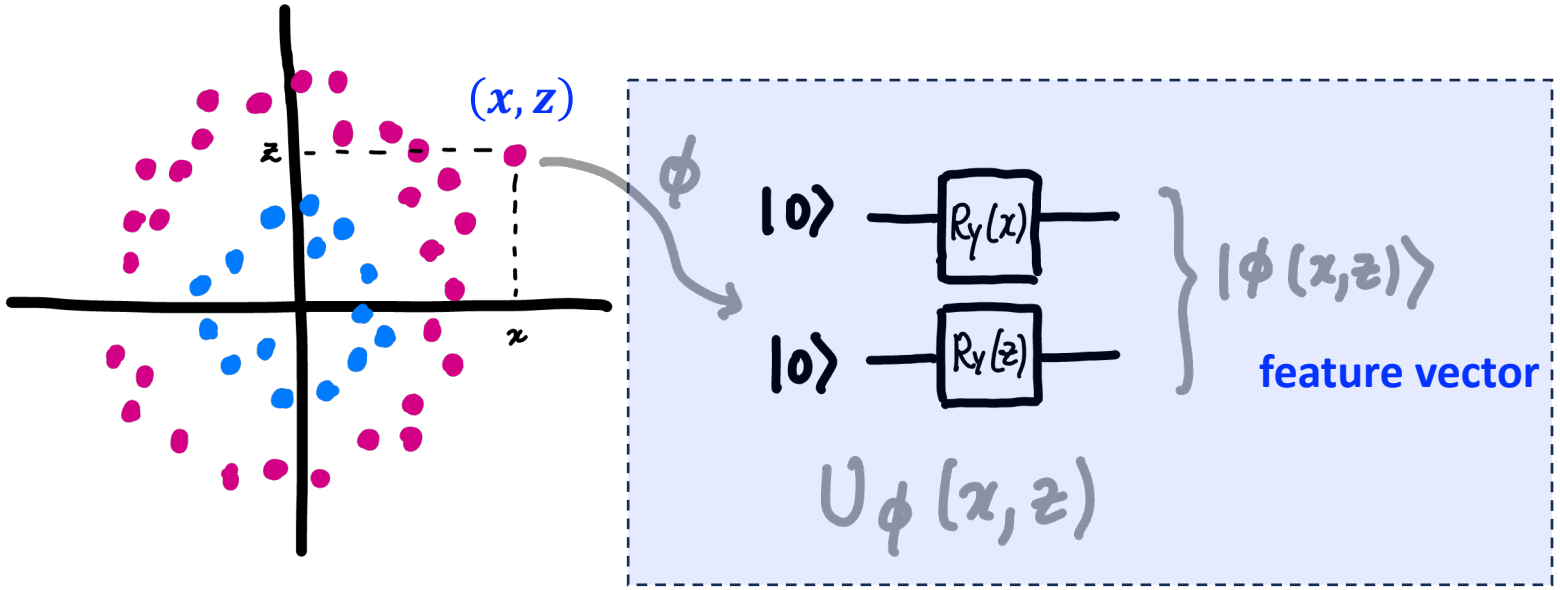
- **Feature Mapping** is a technique used in data analysis and machine learning to *transform* input data *from a lower-dimensional space* to *a higher-dimensional space*, where it can be more easily analyzed or classfied. (https://www.geeksforgeeks.org/feature-mapping/)



$\mathcal{X}$ be a set of input data. A **feature map** $\phi: \mathcal{X} \mapsto \mathcal{F}$ where $\mathcal{F}$ is the **feature space**. The output of the map $\phi(x)$ for all $x \in \mathcal{X}$ are called **feature vectors**.

Quantum Feature Map

- A **quantum feature map** $\phi: \mathcal{X} \mapsto \mathcal{F}$ is a feature map where the vector space $\mathcal{F}$ is a Hilbert space and the feature vectors are quantum states.
  The map transforms $x \mapsto |\phi(x)\rangle$ by way of a unitary transformation $U_\phi(x)$, which is typically a **mapping circuit** whose parameters depend on the input data.



Mapping circuit depends on *parameters* and *encoding method*.

Quantum Data Embedding (or Encoding)

- Quantum Data Embedding:
  - Process to *load* classical data onto a quantum system

$$x \mapsto |\psi(x)\rangle$$

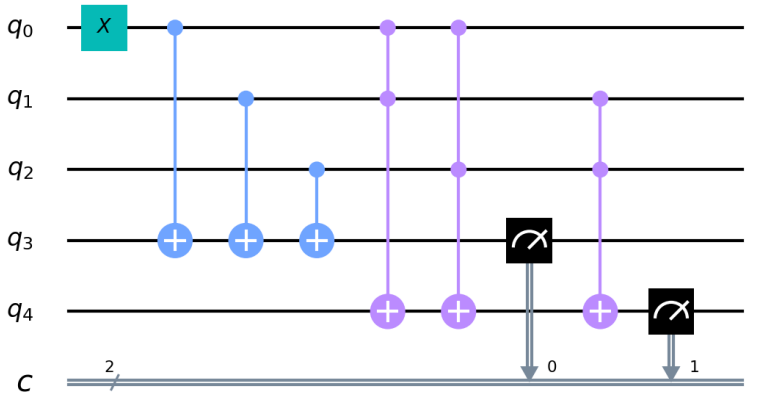
  - Two categories:
    - Digital : classical data → qubit strings, $\frac{1}{\sqrt{M}}\sum_{m=1}^{M}|x(m)\rangle$
    - Analog : classical data → amplitude of a quantum state, $\sum_{i=1}^{N} x_i|i\rangle$ with $\sum|x_i|^2 = 1$

Quantum Data Embedding

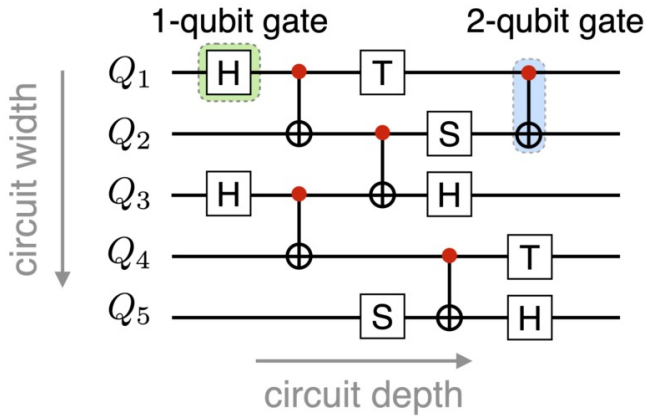
- Guide to select an embedding method:
  1) Data must be represented properly for main computations
     - Digital for arithmetic computation
     - Analog for machine learning tasks

2) #qubits should be minimal
3) #parallel operations should be minimal (to reduce the circuit width)

Constraints of current HW



<Circuit of quantum adder>



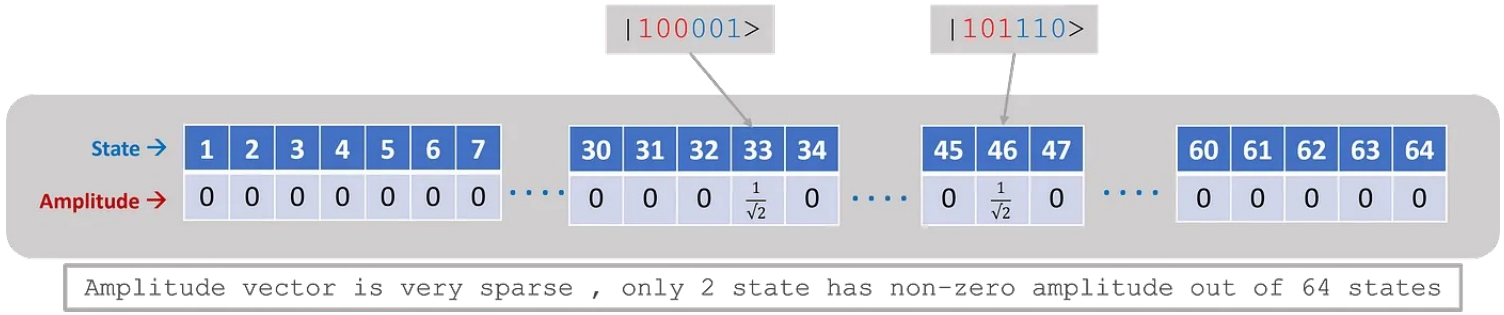
<source: cisco.com>

Basis Encoding

- Basis encoding transforms a bit string into a quantum state on a computational basis. It is primarily used when real numbers have to be arithmetically manipulated in a quantum algorithm.

Input Data			Pre-processing to convert to Binary form		Basis encoded state	
Input Sample	Feature x1	Feature x2	Binary(x1)	Binary(x2)	Basis encoded Quantum state	Amplitude vector
X^1	5	6	101	110	|101110>	$\frac{1}{\sqrt{2}}$|101110>
X^2	4	1	100	001	|100001>	$\frac{1}{\sqrt{2}}$|100001>

6 qubit = 2^6 = 64 States



Basis Encoding

PennyLane Example

Example

Basis embedding encodes the binary feature vector into a basis state.

```
dev = qml.device('default.qubit', wires=3)

@qml.qnode(dev)
def circuit(feature_vector):
    qml.BasisEmbedding(features=feature_vector, wires=range(3))
    return qml.state()

X = [1,1,1]
```

The resulting circuit is:

```
>>> print(qml.draw(circuit, expansion_strategy="device")(X))
0: ──X─┤  State
1: ──X─┤  State
2: ──X─┤  State
```

And, the output state is:

```
>>> print(circuit(X))
    [0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 1.+0.j]
```

Thus, `[1,1,1]` is mapped to $|111\rangle$.

Basis Encoding

Fixed-Point Encoding of Real Number

Definition 3.1 (Fixed-point encoding of real numbers (Rebentrost et al., 2021)). Let c_1, c_2 be positive integers, and $a \in \{0,1\}^{c_1}$, $b \in \{0,1\}^{c_2}$, and $s \in \{0,1\}$ be bit strings. Define the rational number as:

$$\mathcal{Q}(a,b,s) := (-1)^s \left(2^{c_1-1}a_{c_1} + \cdots + 2a_2 + a_1 + \frac{1}{2}b_1 + \cdots + \frac{1}{2^{c_2}}b_{c_2}\right) \in [-R, R], \tag{3.2}$$

where $R := 2^{c_1} - 2^{-c_2}$. If c_1, c_2 are clear from the context, we can use the shorthand notation for a number $z := (a,b,s)$ and write $\mathcal{Q}(z)$ instead of $\mathcal{Q}(a,b,s)$. Given an n-dimensional vector $v \in (\{0,1\}^{c_1} \times \{0,1\}^{c_2} \times \{0,1\})^n$ the notation $\mathcal{Q}(v)$ means an n-dimensional vector whose j-th component is $\mathcal{Q}(v_j)$, for $j \in [n]$.

$$00010.110_2$$
$$= 1\times2^{\mathbf{1}} + 1\times2^{\mathbf{-1}} + 1\times2^{\mathbf{-2}}$$
$$= 2 + 0.5 + 0.25$$
$$= 2.75$$

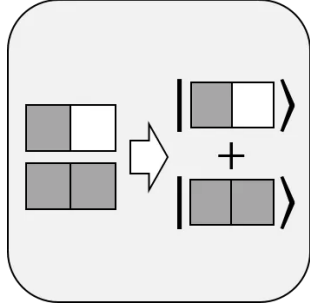
$$000.10110_2$$
$$= 1\times2^{\mathbf{-1}} + 1\times2^{\mathbf{-3}} + 1\times2^{\mathbf{-4}}$$
$$= 0.5 + 0.125 + 0.0625$$
$$= 0.6875$$

Negative number based on 2's complement (2의 보수):

For 4-bit system:

- 0000 ~ 0111 : 0 ~ 7
- 1000 ~ 1111 : -8 ~ -1

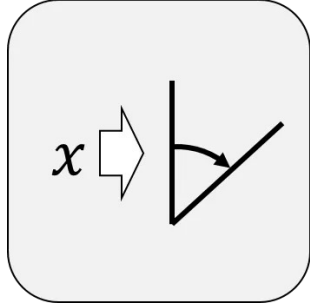
QuAM (Q-Associative Memory)



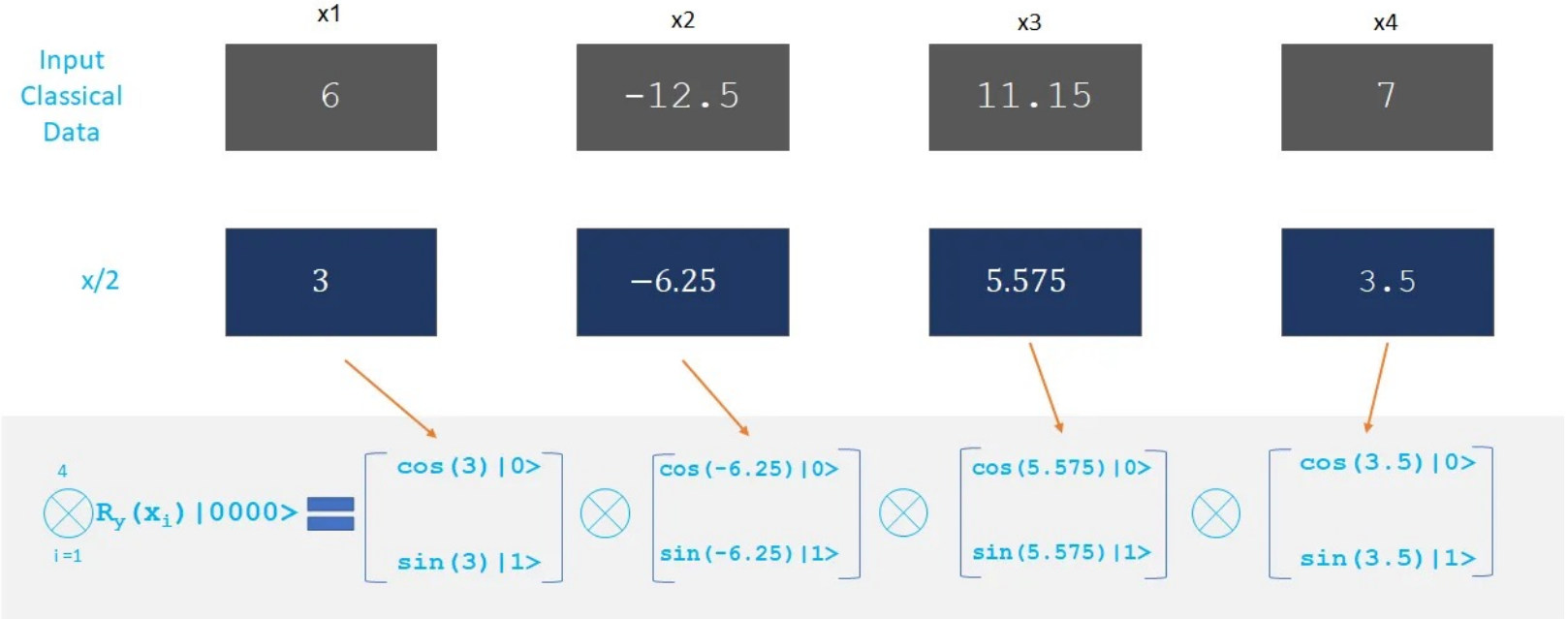
- QuAM encoding is based on superposition to encode a set of data in a qubit register of the same length. It requires a binary representation of all equally long values, or we need to pad the register with zeros.

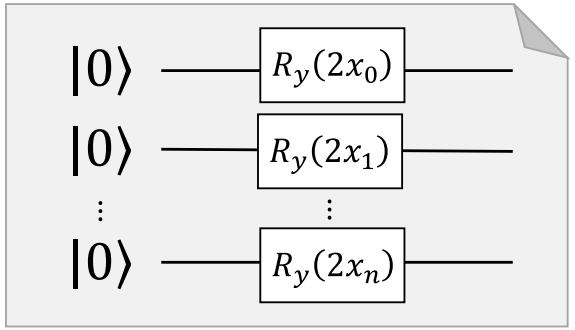
Input variable	Input Classical Data	Binary Number	Basis encoded Quantum Data	QUAM encoded value
X1	10	1010	|1010>	$\frac{1}{\sqrt{3}}\|1010\rangle + \frac{1}{\sqrt{3}}\|1111\rangle + \frac{1}{\sqrt{3}}\|1000\rangle$
X2	15	1111	|1111>	
X3	8	1000	|1000>	

Angle Encoding (Tensor Product Encoding)



- The *n* classical features are encoded into the rotation angle of the *n* qubits.
- This encoding requires *n* qubits to represent n-dimensional data, but is cheaper to prepare in complexity (constant circuit depth): it requires one rotation on each qubit $R_x(v)$ or $R_y(v)$ for the value *v* to encode.





Angle Encoding (Tensor Product Encoding)

PennyLane Example

Example

Angle embedding encodes the features by using the specified rotation operation.

```
dev = qml.device('default.qubit', wires=3)

@qml.qnode(dev)
def circuit(feature_vector):
    qml.AngleEmbedding(features=feature_vector, wires=range(3), rotation='Z')
    qml.Hadamard(0)
    return qml.probs(wires=range(3))

X = [1,2,3]
```

Here, we have also used rotation angles RZ. If not specified, RX is used as default. The resulting circuit is:

```
>>> print(qml.draw(circuit, expansion_strategy="device")(X))
0: ──RZ(1.00)──H─┤  ╭Probs
1: ──RZ(2.00)────┤  ├Probs
2: ──RZ(3.00)────┤  ╰Probs
```

QRAM (Q-Random Access Memory)

- QRAM is used to ***load*** a superposition of data values at once.
- A classical RAM that receives an address with a memory index loads the data at the address into an output register.
- QRAM provides the same functionality, but the address and the output register are quantum registers that can be the superposition of multiple values.

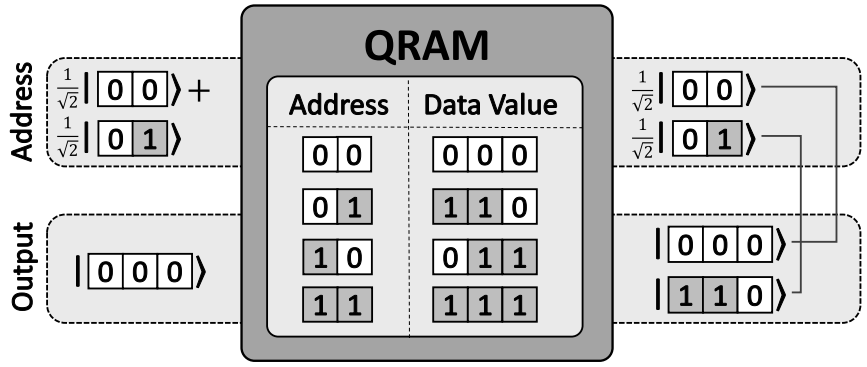


FIGURE 6 Basic functionality of a quantum random access memory (QRAM) is based on [8]. Given an address register that is in a superposition of addresses ($|00\rangle$ and $|01\rangle$), QRAM creates a superposition of addresses and their data values: $\frac{1}{\sqrt{2}}|00\rangle|000\rangle + \frac{1}{\sqrt{2}}|01\rangle|110\rangle$

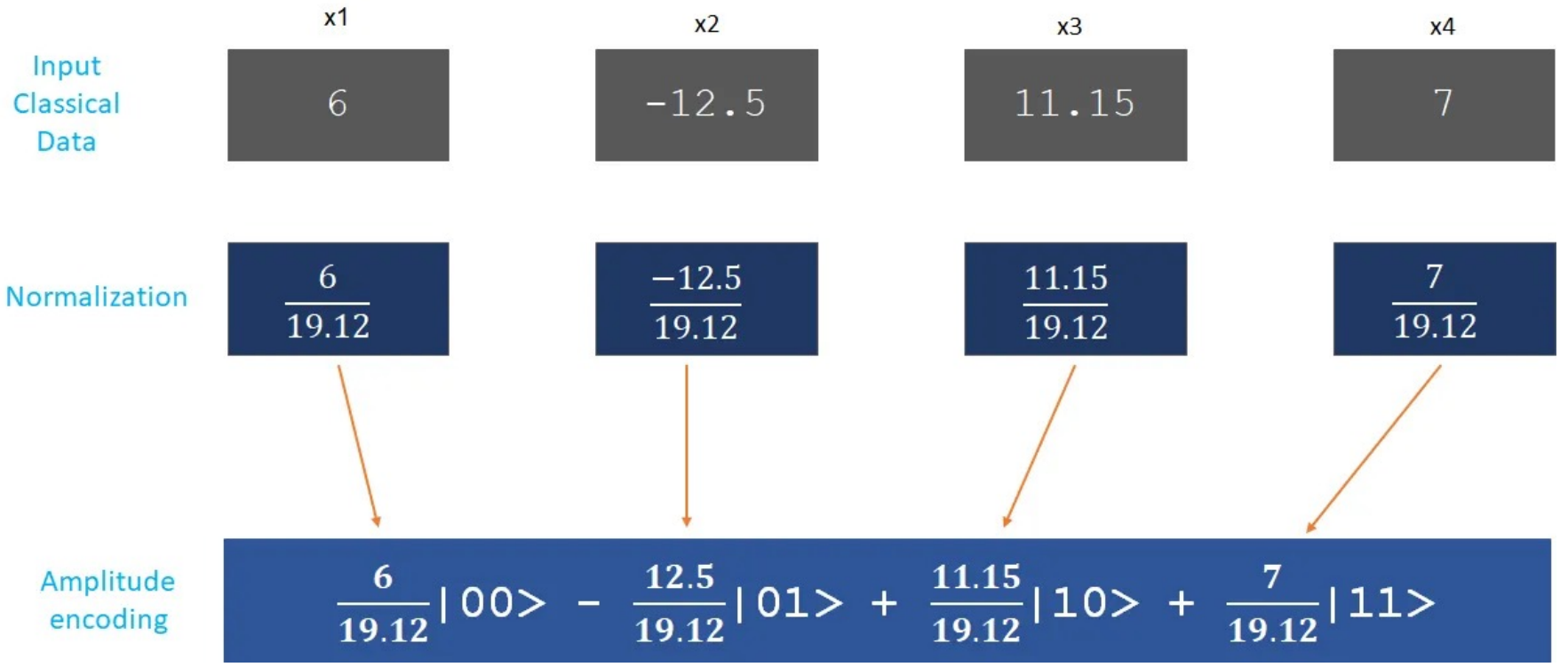
address register

$$\frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a_i\rangle|0\rangle \mapsto \frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a_i\rangle|x_{a_i}\rangle$$

data register

Amplitude Encoding

- Data is encoded into the amplitudes of a quantum state. This encoding requires $\log_2(n)$ qubits to represent an n-dimensional data.



norm factor: $\sqrt{6^2 + (-12.5)^2 + 11.15^2 + 7^2}$

Amplitude Encoding

PennyLane Example

Example

Amplitude embedding encodes a normalized 2^n-dimensional feature vector into the state of n qubits:

```
import pennylane as qml

dev = qml.device('default.qubit', wires=2)

@qml.qnode(dev)
def circuit(f=None):
    qml.AmplitudeEmbedding(features=f, wires=range(2))
    return qml.expval(qml.PauliZ(0)), qml.state()

res, state = circuit(f=[1/2, 1/2, 1/2, 1/2])
```

The final state of the device is - up to a global phase - equivalent to the input passed to the circuit:

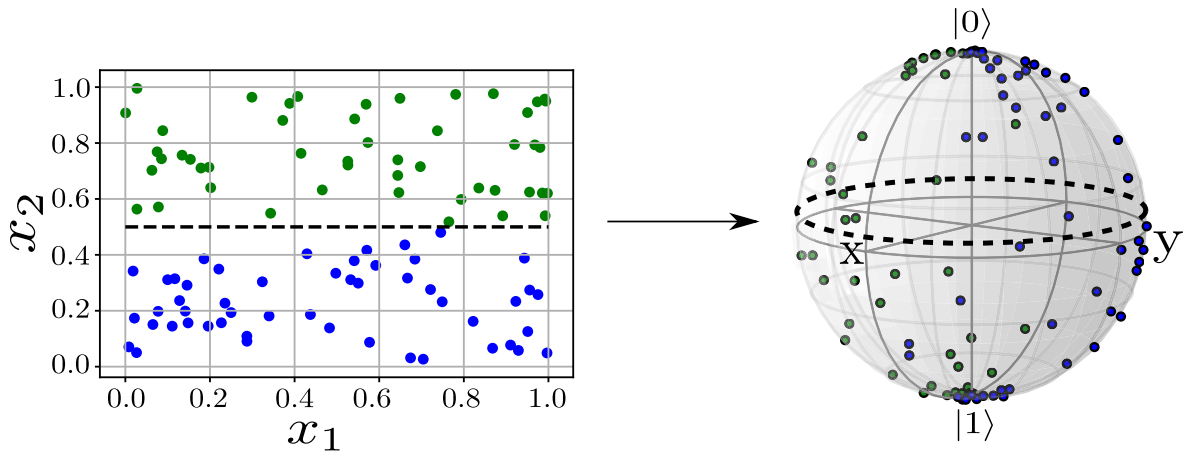
```
>>> state
tensor([0.5+0.j, 0.5+0.j, 0.5+0.j, 0.5+0.j], requires_grad=True)
```

Dense Angle Encoding (PRA 102, 032420)

- Given a feature vector $x = [x_1, x_2, \cdots, x_N]^T \in \mathbb{R}^N$, the dense angle encoding maps $x \mapsto E(x)$ given by

$$|x\rangle = \bigotimes_{i=1}^{\lceil N/2 \rceil} \cos(\pi x_{2i-1}) |0\rangle + e^{2\pi i x_{2i}} \sin(\pi x_{2i-1}) |1\rangle$$

This encoding exploits an additional property of qubits (relative phase) to use the only *n*/2 qubits to encode *n* data points



<Visual representation of data encoding for a single qubit, PRA 102, 032420>

General Qubit Encoding (PRA 102, 032420)

- Given a feature vector $x = [x_1, x_2, \cdots, x_N]^T \in \mathbb{R}^N$,
  the general qubit encoding maps $x \mapsto E(x)$ given by

$$|x\rangle = \bigotimes_{i=1}^{\lceil N/2 \rceil} \boldsymbol{f_i(x_{2i-1}, x_{2i})}|0\rangle + \boldsymbol{g_i(x_{2i-1}, x_{2i})}|1\rangle,$$

  where $f, g: \mathbb{R}\times\mathbb{R} \to \mathbb{C}$ are such that $|f_i|^2 + |g_i|^2 = 1$.

Qsample Encoding

- **Qsample encoding** is a method to connect a concept of probability distribution of a discrete random variable to a quantum state.
- Assume there is a PMF of random variable X as $\Pr(X = i) = p_i$. Any discrete random variable could be represented like it by indexing the events. For them, we define a quantum state $|\psi\rangle = \sum_{i=1}^{N} p_i |i\rangle$.
- Joint probability of random variables:
  - $|x\rangle = \sum_{i=1}^{N} \sqrt{\Pr(X = i)}|i\rangle$ and $|y\rangle = \sum_{i=1}^{N} \sqrt{\Pr(Y = j)}|j\rangle$
  - $|x, y\rangle = \sum_{i,j=1}^{N} \sqrt{\Pr(X = i)\Pr(Y = j)}|i\rangle|j\rangle$ if two variables are independent.
  - $|x, y\rangle = \sum_{i,j=1}^{N} \sqrt{\Pr(X = i, Y = j)}\,|i\rangle|j\rangle$ if two variables are dependent each other.

PMF(Probability Mass Function) : 확률질량함수, 이산확률변수에서 특정 값에 대한 확률을 나타내는 함수

Hamiltonian Encoding (Matrix Encoding)

- This encoding encodes the data into the operator.
  A matrix A has to be represented as an operation on a quantum computer.
- If A is not Hermitian, encode $H = \begin{pmatrix} O & A \\ A^\dagger & O \end{pmatrix}$ instead of A.
- To make the matrix as a unitary matrix, we use a matrix exponential as e^{-iAt}.
- Since any unitary matrix A can be decomposed with Pauli gates such as $A = \alpha I + \beta X + \delta Y + \gamma Z$ based on a real vector $(\alpha, \beta, \delta, \gamma)$.
- Then, we can develop the unitary evolution as follows,
  $e^{iAt} = e^{i(\alpha I + \beta X + \delta Y + \gamma Z)t} = e^{i\alpha t I} \cdot e^{i\beta t X} \cdot e^{i\delta t Y} \cdot e^{i\gamma t Y} = R_x(\beta t) R_Y(\delta t) R_Z(\gamma t).$
- We can express any unitary evolution by three rotation operators.

$e^{iAx} = \cos(x)\, I + i \sin(x)\, A$ when $x\ is\ real\ and\ A^2 = I$

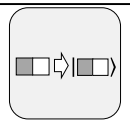
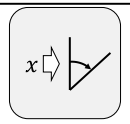
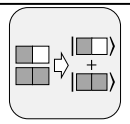
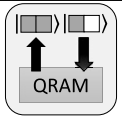
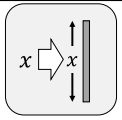
Hamiltonian Encoding (Matrix Encoding)

- However, in reality, the circuit identity cannot be established because the commutativity rule does not hold for matrices, That is, $AB \neq BA$ so $e^{A+B} \neq e^{A}e^{B}$.
- This can be overcome by applying Trotter Suzuki formula,
  For large enough r, the equation holds $e^{-i(H_1+H_2)} \approx \left(e^{-iH_A t/r} e^{-iH_B t/r}\right)^r$.

- The hamiltonian encoding method is densely connected with Physics. Therefore, various physical algorithms including VQE and QAOA are developed with hamiltonian encoding.

Review

TABLE 1 Comparison of data encoding patterns. For the QUANTUM RANDOM ACCESS MEMORY (QRAM) ENCODING, we assume that all n data points are loaded

Encoding pattern		Encoding	Req. qubits
	BASIS ENCODING [13]	$x_i \approx \sum_{i=-k}^{m} b_i 2^i \mapsto \lvert b_m \dots b_{-k} \rangle$	$l = k+m$ per data-point
	**ANGLE ENCODING**	$x_i \mapsto cos(x_i) \lvert 0 \rangle + sin(x_i) \lvert 1 \rangle$	1 per data-point
	QUAM ENCODING [13]	$X \mapsto \sum_{i=0}^{n-1} \frac{1}{\sqrt{n}} \lvert x_i \rangle$	l
	**QRAM ENCODING**	$X \mapsto \sum_{n=0}^{n-1} \frac{1}{\sqrt{n}} \lvert i \rangle \lvert x_i \rangle$	$\lceil \log n \rceil + l$
	AMPLITUDE ENCODING [13]	$X \mapsto \sum_{i=0}^{n-1} x_i \lvert i \rangle$	$\lceil \log n \rceil$

Quantum Analog-Digital Conversion

PHYSICAL REVIEW A **99**, 012301 (2019)

Quantum analog-digital conversion

Kosuke Mitarai,[1,*] Masahiro Kitagawa,[1,2] and Keisuke Fujii[3,4,†]
[1]*Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*
[2]*Quantum Information and Quantum Biology Division, Institute for Open and Transdisciplinary Research Initiatives, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*
[3]*Graduate School of Science, Kyoto University, Yoshida-Ushinomiya-cho, Sakyo-ku, Kyoto 606-8302, Japan*
[4]*JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan*

(Received 21 June 2018; published 2 January 2019)

Many quantum algorithms, such as the Harrow-Hassidim-Lloyd (HHL) algorithm, depend on oracles that efficiently encode classical data into a quantum state. The encoding of the data can be categorized into two types: analog encoding, where the data are stored as amplitudes of a state, and digital encoding, where they are stored as qubit strings. The former has been utilized to process classical data in an exponentially large space of a quantum system, whereas the latter is required to perform arithmetics on a quantum computer. Quantum algorithms such as HHL achieve quantum speedups with a sophisticated use of these two encodings. In this work, we present algorithms that convert these two encodings to one another. While quantum digital-to-analog conversions have implicitly been used in existing quantum algorithms, we reformulate it and give a generalized protocol that works probabilistically. On the other hand, we propose a deterministic algorithm that performs a quantum analog-to-digital conversion. These algorithms can be utilized to realize high-level quantum algorithms such as a nonlinear transformation of amplitudes of a quantum state. As an example, we construct a "quantum amplitude perceptron," a quantum version of the neural network that hence has a possible application in the area of quantum machine learning.

DOI: 10.1103/PhysRevA.99.012301

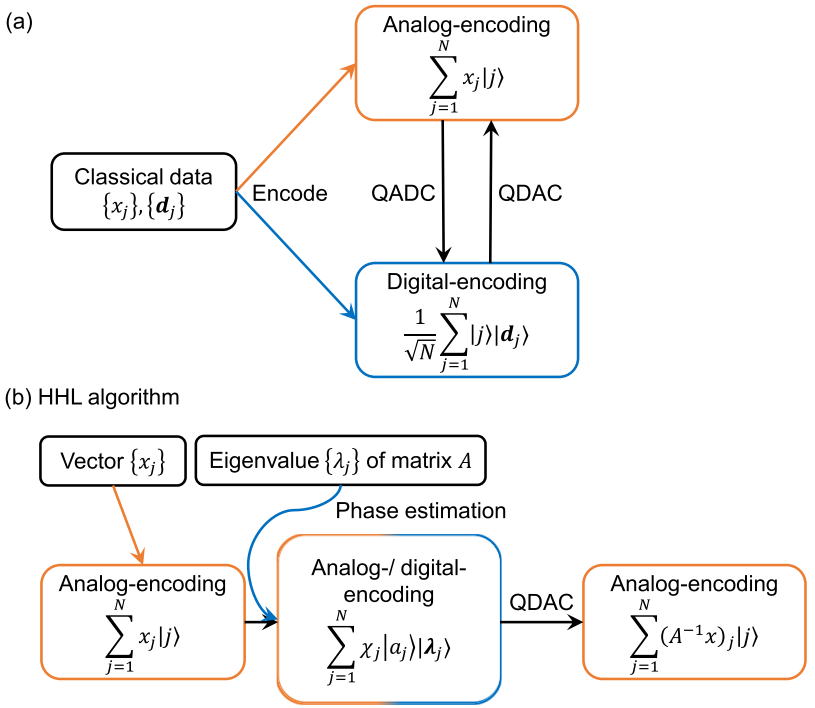


FIG. 1. (a) Schematic sketch of analog encoding and digital encoding. QDAC and QADC mediate these two encodings. (b) A brief flowchart of the HHL algorithm [4]. $\{|a_j\rangle\}$ denote eigenvectors of a Hermitian matrix A, each corresponding to eigenvalues $\{\lambda_j\}$. χ_j are complex numbers such that $\sum_{j=1}^{N} x_j|j\rangle = \sum_{j=1}^{N} \chi_j|a_j\rangle$.

Use Case : Solve HHL Algorithm

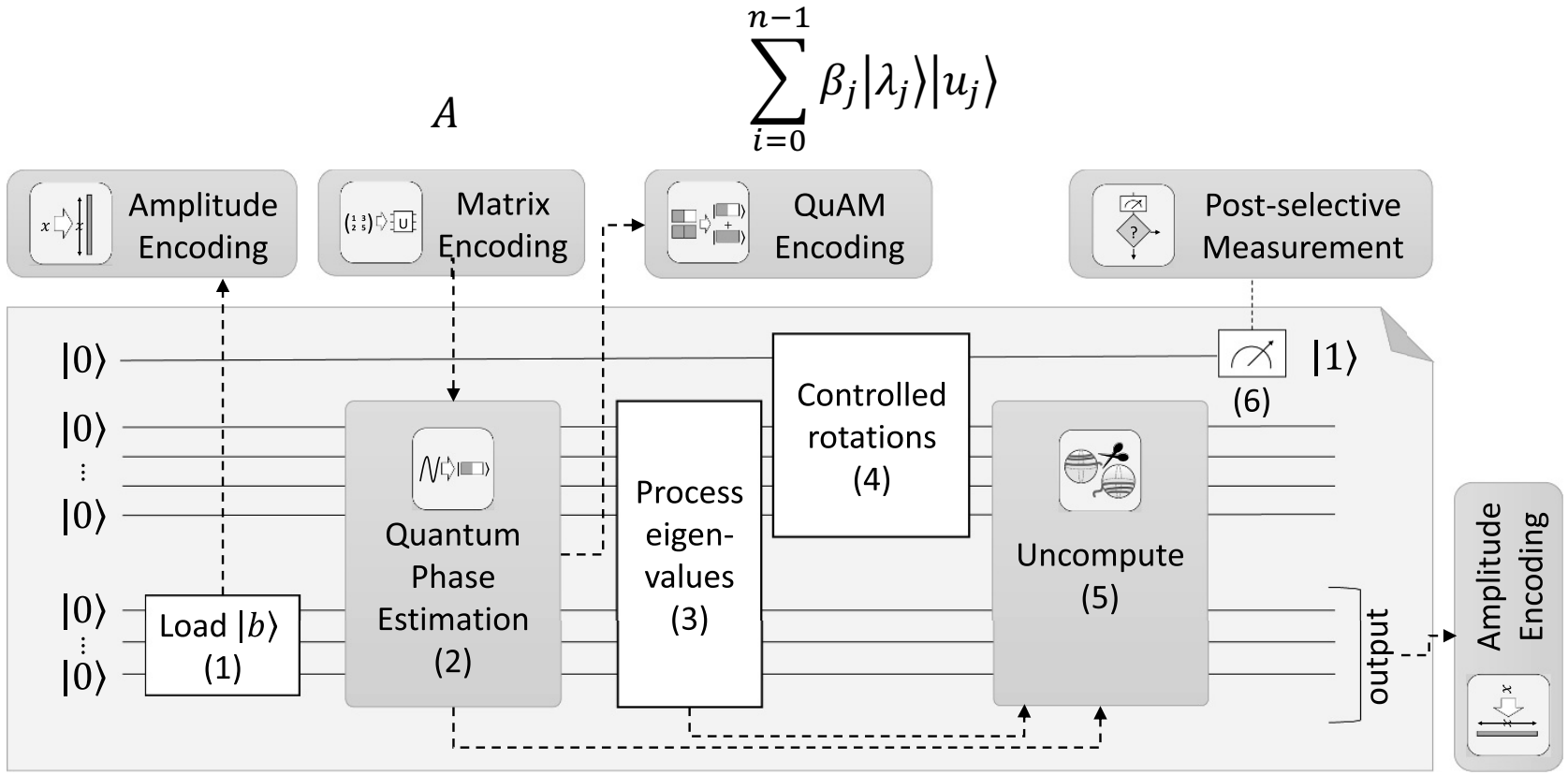
HHL (Harrow-Hassidim-Lloyd)

- A system of linear equations: $Ax = b$
  Given $A \in \mathbb{C}^{n\times n}$ which fulfils certain requirements* and a vector $|b\rangle \in \mathbb{C}^n$, find $|x\rangle \in \mathbb{C}^n$ such that $A|x\rangle = |b\rangle$. An equivalent formulation of this equation is $|x\rangle = A^{-1}|b\rangle$.
- Due to the spectral theorem, $|b\rangle$ can be written in terms of the eigenvectors $\{u_i\}$ of A, $|b\rangle = \sum_{j=0}^{n-1} \beta_j \left|u_j\right\rangle$. Then, the equation is reformulated as

$$|x\rangle = \sum_{j=0}^{n-1} \lambda_j^{-1} \beta_j \left|u_j\right\rangle.$$

The condition number of A is low and A is a sparse matrix.

HHL (Harrow-Hassidim-Lloyd)



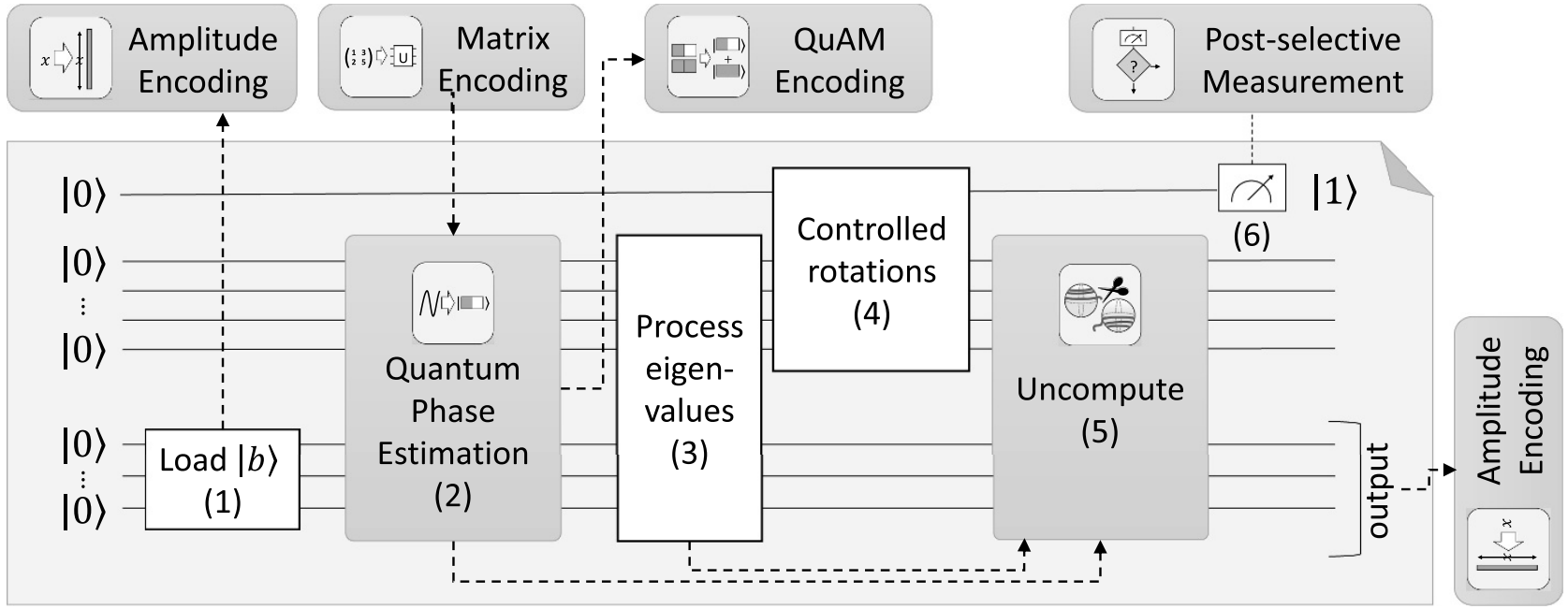
$$|b\rangle = \sum_{i=0}^{n-1} b_i |i\rangle$$

HHL (Harrow-Hassidim-Lloyd)

restrictions on A

A

$$\sum_{i=0}^{n-1} \beta_j |\lambda_j\rangle |u_j\rangle$$



$$|b\rangle = \sum_{i=0}^{n-1} b_i |i\rangle$$

repetitions of readout

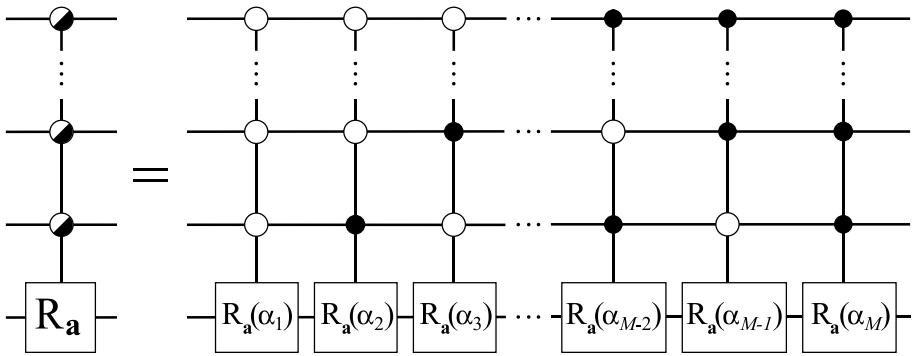
exponential time

Preparation of Arbitrary Quantum State

Preparation of Arbitrary Quantum State

- Suppose that a quantum state $|s\rangle$ has to be prepared on an empty qubit register. If the state preparation method is *not known* that exploits the structure of the state to prepare it efficiently, we have to use a method for creating an arbitrary state instead.

#1 Uniformly Controlled Rotation Based (Mottonen, QIC Vol. 5, No. 6, 467)



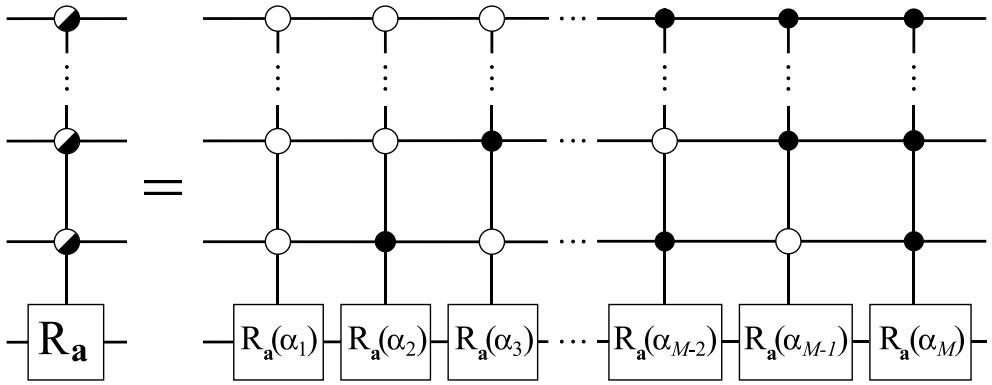
k-fold Uniformly Controlled Rotation Gate, $F_m^k(a,\alpha)$

FIG. 1: Definition of the k-fold uniformly controlled rotation $F_m^k(\mathbf{a},\boldsymbol{\alpha})$ of qubit m about the axis $\mathbf{a}$. The left hand side defines the gate symbol used for the uniformly controlled rotation. The enumeration of the qubits is arbitrary with the exception that the target qubit is the m^{th} one. The black control bits stand for value 1 and the white for 0. Above, $M = 2^k$.

$$|\psi\rangle = \alpha_0|000\rangle|t\rangle + \alpha_1|001\rangle|t\rangle + \alpha_2|010\rangle|t\rangle + \alpha_3|011\rangle|t\rangle + \alpha_4|100\rangle|t\rangle + \alpha_5|101\rangle|t\rangle + \alpha_6|110\rangle|t\rangle + \alpha_7|111\rangle|t\rangle$$

$$F_4^3(a,\alpha)|\psi\rangle = \alpha'_0|000\rangle|t\rangle + \alpha'_1|001\rangle|t\rangle + \alpha'_2|010\rangle|t\rangle + \alpha'_3|011\rangle|t\rangle + \alpha'_4|100\rangle|t\rangle + \alpha'_5|101\rangle|t\rangle + \alpha'_6|110\rangle|t\rangle + \alpha'_7|111\rangle|t\rangle$$

#1 Uniformly Controlled Rotation Based

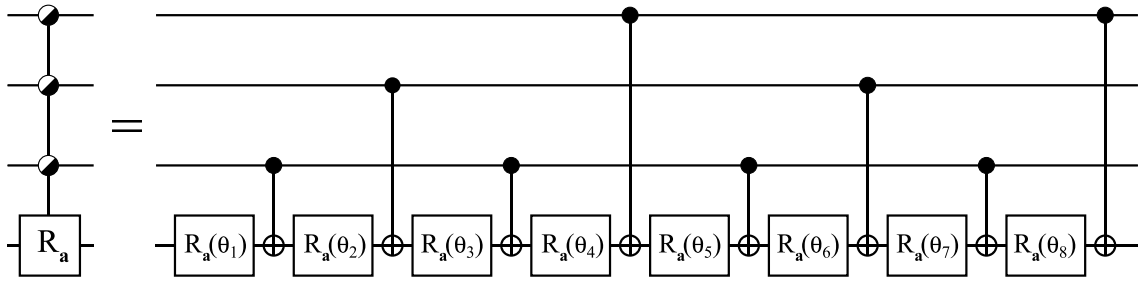


k-fold Uniformly Controlled Rotation Gate, $F_m^k(a, \alpha)$

$$\begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{2^k} \end{pmatrix} = M \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2^k} \end{pmatrix}$$

$$M_{ij} = 2^{-k}(-1)^{b_{j-1} \cdot g_{i-1}}$$

- b_m : *binary* code
- g_m : *gray* code for integer m



Single- and Two-qubit gate decomposition of *k*-fold Uniformly Controlled Rotation Gate

2^k CNOT + 2^k R_a

#1 Uniformly Controlled Rotation Based

- **Gray code** : 자료를 표현하는 방식 중 하나. I/O 장치, A/D 변환기, 주변장치 등에서 숫자를 표현할 때 사용함. 수의 크기가 변할 때 인접한 수 사이에 한자리만 (1 bit) 변하게 만들어진 코드이다.

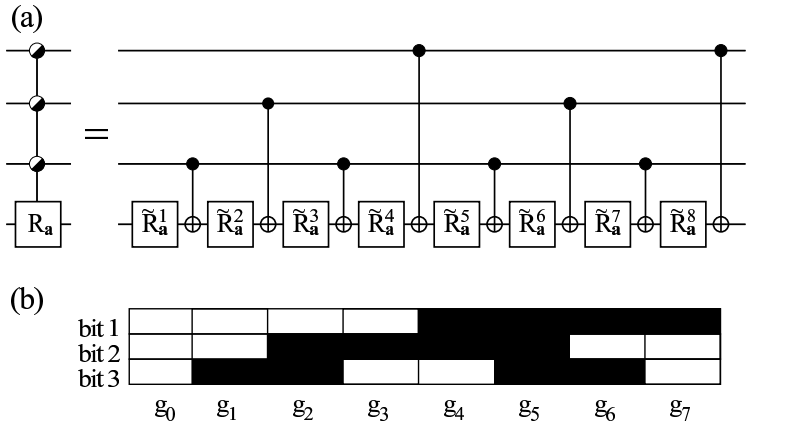


Figure 2: (a) Quantum circuit realizing the gate $F_4^3(R_\mathbf{a})$, where $\mathbf{a}$ is perpendicular to the x-axis. Here we have used a notation $\tilde{R}_\mathbf{a}^j = R_\mathbf{a}(\theta_j)$. (b) Binary reflected 3-bit Gray code used to define the positions of the control nodes. The black and white rectangles denote bit values one and zero, respectively.

Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
..	..	..
510	01 1111 1110	01 0000 0001
511	01 1111 1111	01 0000 0000

The difference in the vector element indicates the control qubit of CNOT

#1 Uniformly Controlled Rotation Based

- Find a gate sequence corresponding to a matrix U such that $U|a\rangle = |b\rangle$ for given $|a\rangle$ and $|b\rangle$.
- For convenience, we transform $|a\rangle$ to $|\bar{0}\rangle$ than arbitrary target state.
  (practically, $|a\rangle \rightarrow |\bar{0}\rangle \rightarrow |b\rangle$)

Our algorithm for transforming $|a\rangle = (|a_1|e^{i\omega_1}, |a_2|e^{i\omega_2}, \ldots, |a_N|e^{i\omega_N})^T$ into $|e_1\rangle$ works as follows:

- First we equalize the phases ω_i using a cascade of uniformly controlled z-rotations Ξ_z, rendering the vector real up to the global phase ϕ: $\Xi_z |a\rangle = e^{i\phi} |\hat{a}\rangle$.
- Then we rotate the real state vector $|\hat{a}\rangle$ into the direction of $|e_1\rangle$ using a similar sequence of uniformly controlled y-rotations Ξ_y, thus achieving our goal.

$$\Xi_z = \prod_{j=1}^{n} F_j^{j-1}\left(z, \alpha_{n-j+1}^z\right) \otimes I_{2^{n-j}} \qquad \Xi_y = \prod_{j=1}^{n} F_j^{j-1}\left(y, \alpha_{n-j+1}^y\right) \otimes I_{2^{n-j}}$$

$$\Xi_y \Xi_z |a\rangle = \left(\prod_{j=1}^{n} F_j^{j-1}(\mathbf{y}, \boldsymbol{\alpha}_{n-j+1}^y) \otimes I_{2^{n-j}} \right) \left(\prod_{j=1}^{n} F_j^{j-1}(\mathbf{z}, \boldsymbol{\alpha}_{n-j+1}^z) \otimes I_{2^{n-j}} \right) |a\rangle = e^{i \sum_{j=1}^{N} \omega_j / N} |e_1\rangle . \qquad (7)$$

#1 Uniformly Controlled Rotation

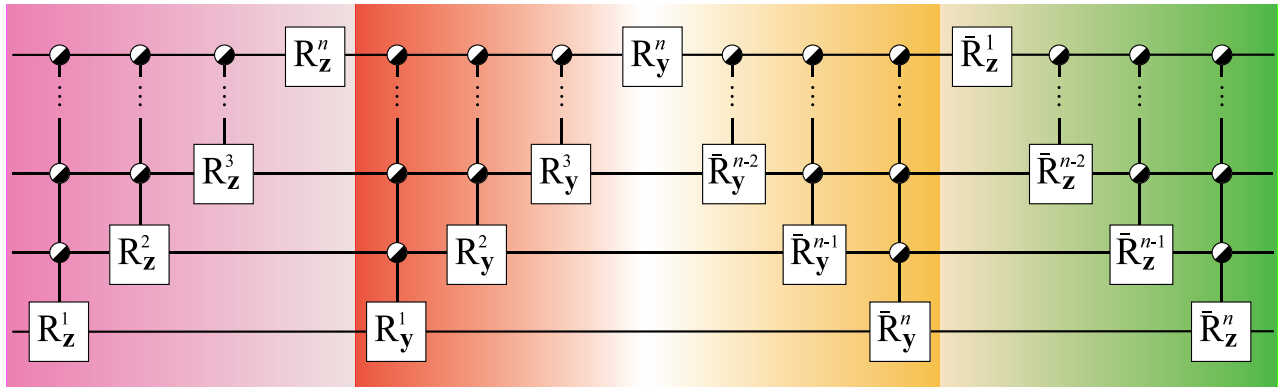


FIG. 3: Gate sequence for state preparation using uniformly controlled rotations. The rotation angles $\{\alpha_{j,k}^q\}$ for the uniformly controlled rotations are given in Eqs. (8) and (5).

- On considering $|a\rangle \rightarrow |\bar{0}\rangle \rightarrow |b\rangle$, the entire circuit requires $2^{n+2} - 4n - 4$ CNOTs and $2^{n+2} - 5$ one-qubit rotations

#1 Unif

Example

`MottonenStatePreparation` creates any arbitrary state on the given wires depending on the input state vector.

```
dev = qml.device('default.qubit', wires=3)

@qml.qnode(dev)
def circuit(state):
    qml.MottonenStatePreparation(state_vector=state, wires=range(3))
    return qml.state()

state = np.array([1, 2j, 3, 4j, 5, 6j, 7, 8j])
state = state / np.linalg.norm(state)

print(qml.draw(circuit, expansion_strategy="device", max_length=80)(state))
```

```
0: ──RY(2.35)─╭●──────────╭●──────────╭●──────────────────────╭●
1: ──RY(2.09)─╰X──RY(0.21)─╰X─╭●──────┤──────────╭●──────────┤─
2: ──RY(1.88)─────────────────╰X──RY(0.10)─╰X──RY(0.08)─╰X──RY(0.15)─╰X

──╭●────────╭●────╭●────╭●─┤  State
──╰X────────╰X─╭●─┤──╭●─┤──┤  State
──RZ(1.57)─────╰X─╰X─╰X─╰X─┤  State
```

The state preparation can be checked by running:

```
>>> print(np.allclose(state * np.exp(1j * -0.785396), circuit(state)))
    True
```

The state is equal to the input state upto a global phase. This phase is given by `np.exp(1j * -0.785396)` in this example.

#2 Schmidt Decomposition Method (PRA 83, 032302)

- **(Schmidt Decomposition)**
  If $|\psi\rangle$ is a vector in a tensor product space $\mathcal{H}_A \otimes \mathcal{H}_B$, then there exists an orthonormal basis $\{(|\varphi_i^A\rangle)\}$ for $\mathcal{H}_A$, and an orthonormal basis $\{(|\varphi_i^B\rangle)\}$ for $\mathcal{H}_B$, and non-negative real numbers $\{p_i\}$ so that

$$|\psi\rangle = \sum_i \alpha_i |\varphi_i^A\rangle |\varphi_i^B\rangle .$$

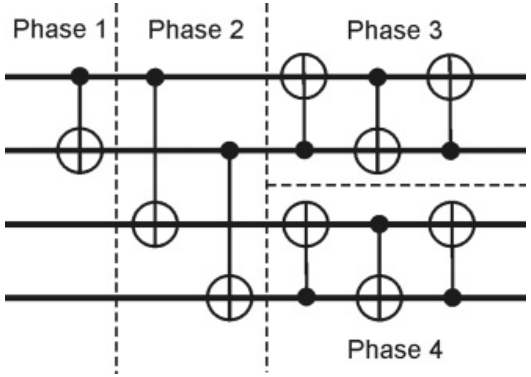
- The coefficients α_i are called ***Schmidt coefficients***.

#2 Schmidt Decomposition Method

- **(4-Qubit Case Study)**
  An arbitrary 4-qubit state can be expressed using the Schmidt decomposition as $|\psi\rangle = \sum_{i=1}^{4} \alpha_i |\psi_i\rangle|\phi_i\rangle$.
  Note that $|\psi_i\rangle$ ($|\phi_i\rangle$) are the orthonormal states of the first (second) two qubits.

- (Procedures: a total of 4 steps)

1) Given the initial state $|0000\rangle$, generate the state with the Schmidt coefficients on the first two qubits:
   $(\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle)|00\rangle$
2) Perform qubit-wise CNOT gates
   $(\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle)|00\rangle$
   $\rightarrow (\alpha_1|00\rangle|00\rangle + \alpha_2|01\rangle|01\rangle + \alpha_3|10\rangle|10\rangle + \alpha_4|11\rangle|11\rangle)$



<source: PRA 83, 032302>

#2 Schmidt Decomposition Method

- (Procedures: cont'd)

3) Apply the unitary operation to transform the basis states of the first two qubits into the four orthonormal states $|\psi_i\rangle$:

$$|00\rangle \rightarrow |\psi_1\rangle, |01\rangle \rightarrow |\psi_2\rangle, |10\rangle \rightarrow |\psi_3\rangle, |11\rangle \rightarrow |\psi_4\rangle$$

4) Apply the unitary operation to transform the basis states of the second two qubits into the four orthonormal states $|\phi_i\rangle$:

$$|00\rangle \rightarrow |\phi_1\rangle, |01\rangle \rightarrow |\phi_2\rangle, |10\rangle \rightarrow |\phi_3\rangle, |11\rangle \rightarrow |\phi_4\rangle$$

#2 Schmidt Decomposition Method

- **(5-Qubit Case Study)**
  Factorize the Hilbert space into two parts, with one part based on two qubits and the other part is associated with three qubits.
- The Schmidt decomposition of an arbitrary 5-qubit state has the almost the same structure with that the 4-qubit case, $|\psi\rangle = \sum_{i=1}^{4} \alpha_i |\psi_i\rangle|\phi_i\rangle$, except that $|\phi_i\rangle$ are 3-qubit states.
- Therefore, the steps 1~3 are the same with those in the 4-qubit procedure.
- Step 4: Apply the unitary operation to transform the basis states of the second three qubits into the four orthonormal states $|\phi_i\rangle$:

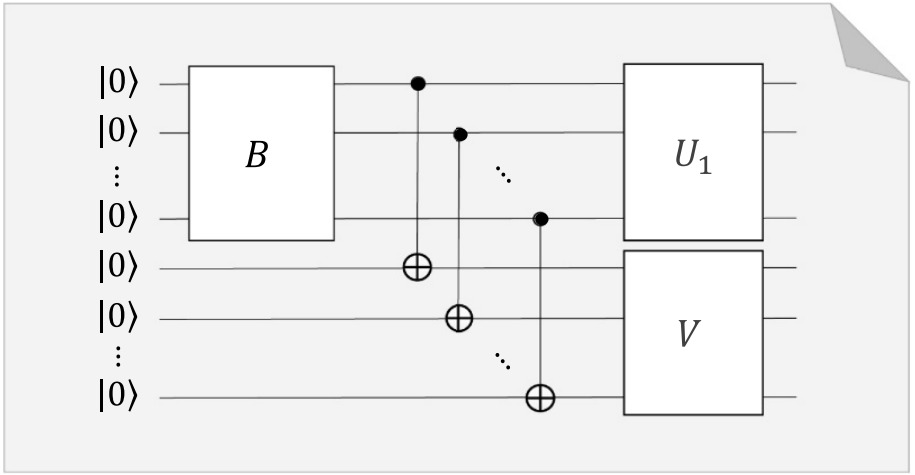
$$|00\rangle|0\rangle \rightarrow |\phi_1\rangle, |01\rangle|0\rangle \rightarrow |\phi_2\rangle, |10\rangle|0\rangle \rightarrow |\phi_3\rangle, |11\rangle|0\rangle \rightarrow |\phi_4\rangle$$

#2 Schmidt Decomposition Method

Step	Even Size (n=2k)	Odd Size (n=2k+1)
1	$\|0\rangle^{\otimes 2k} \longrightarrow \left(\sum_{i=1}^{2^k} \alpha_i \|i\rangle\right) \|0\rangle^{\otimes k}$	$\|0\rangle^{\otimes 2k+1} \longrightarrow \left(\sum_{i=1}^{2^k} \alpha_i \|i\rangle\right) \|0\rangle^{\otimes k+1}$
2	$\left(\sum_{i=1}^{2^k} \alpha_i \|i\rangle\right) \|0\rangle^{\otimes k} \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|i\rangle \|i\rangle$	$\left(\sum_{i=1}^{2^k} \alpha_i \|i\rangle\right) \|0\rangle^{\otimes k+1} \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|i\rangle \|i\rangle\|0\rangle$
3	$\sum_{i=1}^{2^k} \alpha_i \|i\rangle \|i\rangle \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|i\rangle$	$\sum_{i=1}^{2^k} \alpha_i \|i\rangle \|i\rangle \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|i\rangle\|0\rangle$
4	$\sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|i\rangle \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|\phi_i\rangle$	$\sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|i\rangle \longrightarrow \sum_{i=1}^{2^k} \alpha_i \|\psi_i\rangle \|\phi_i\rangle$

#2 Schmidt Decomposition Method

- **Schmidt Decomposition and Singular Value Decomposition**:
  A quantum state $|s\rangle$ can be expressed in terms of two subspace V and W. The orthogonal basis $\{f_1, f_2, \cdots, f_k\} \in V$ and $\{g_1, g_2, \cdots, g_k\} \in W$ are chosen, and then $|s\rangle$ is represented as $|s\rangle = \sum_{i,j} b_{i,j} \cdot f_i \otimes g_j$.
- Then, SVD of the matrix $M = \{b_{i,j}\}$ is computed as $M = (U_1 U_2) \begin{pmatrix} A \\ 0 \end{pmatrix} V^*$.
- The entries of the diagonal matrix A build the set $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, which defined the Schmidt decomposition of $|s\rangle$: $|s\rangle = \sum_i \alpha_i |\varphi_i^A\rangle |\varphi_i^B\rangle$



<state preparation circuit>

#2 Schmidt Decomposition Method

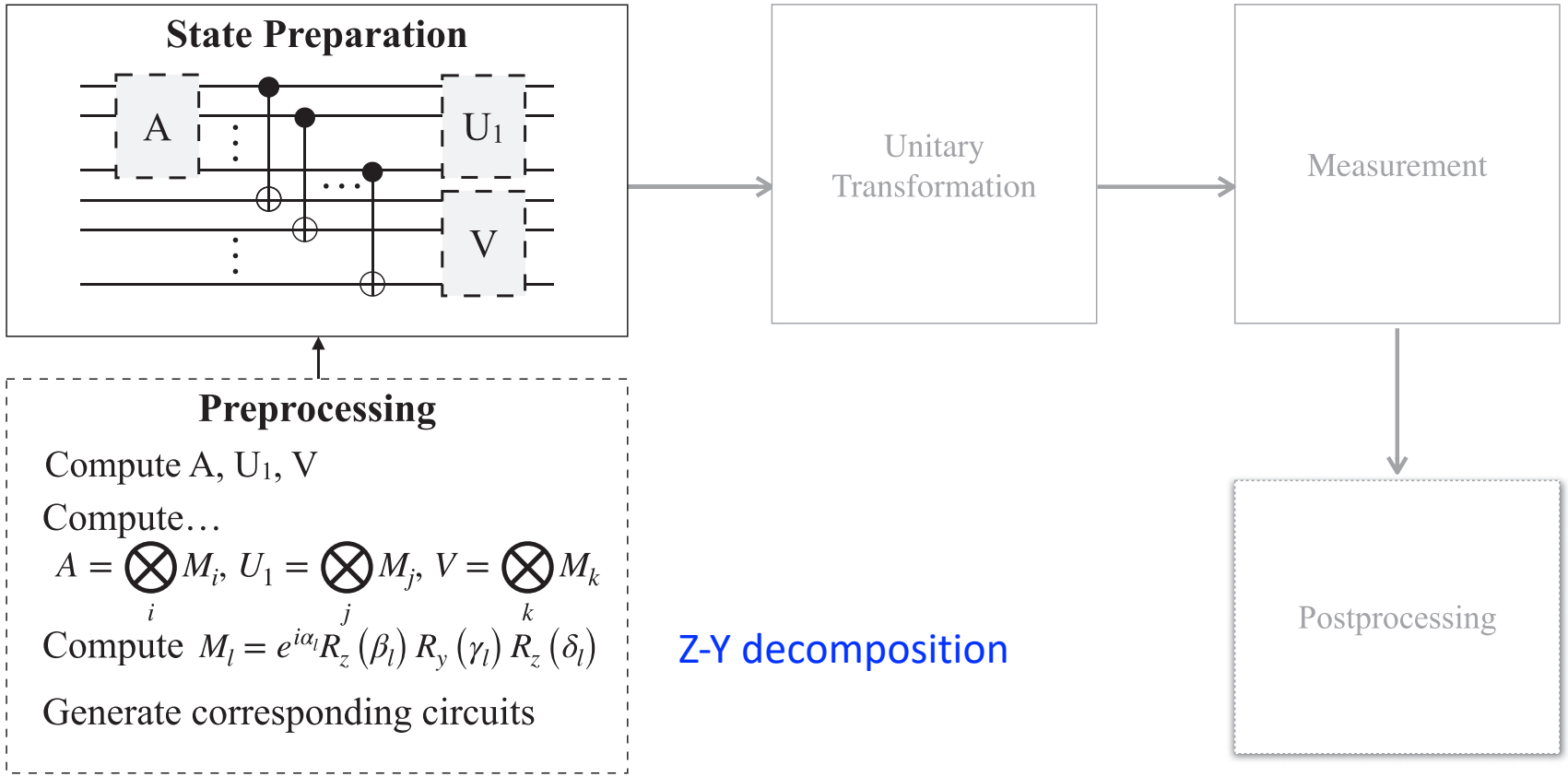


Figure 4. Preparing input based on Schmidt decomposition.

Robust Data Embedding (PRA 102, 032420)

Quantum Classifier

- (**Quantum Classifier**) A (binary) quantum classifier is defined as a procedure for encoding data into quantum circuit, processing it through trainable circuit (QNN), and outputting a (binary) predictable label

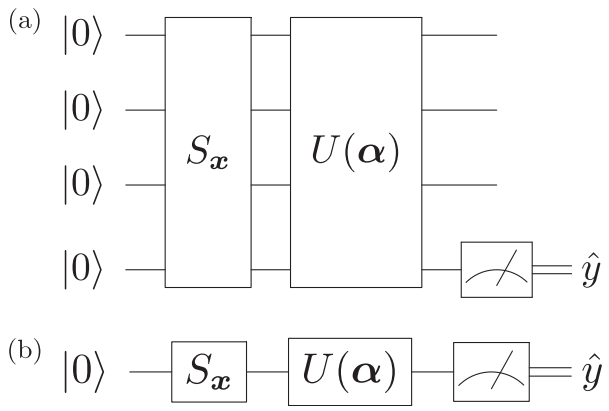


FIG. 1. A common architecture for a binary quantum classifier that we study in this paper. The general circuit structure is shown in (a) and the structure for a single qubit is highlighted in (b). In both, a feature vector $\boldsymbol{x}$ is encoded into a quantum state ρ_x via a "state preparation" unitary S_x. The encoded state ρ_x then evolves to $U\rho_x U^\dagger =: \tilde{\rho}_x$ where $U(\boldsymbol{\alpha})$ is a unitary Ansatz with trainable parameters $\boldsymbol{\alpha}$. A single qubit of the evolved state $\tilde{\rho}_x$ is measured to yield a predicted label $\hat{y}$ for the vector $\boldsymbol{x}$.

A (binary) Quantum classifier consists of three well-defined functions:

1) Encoding function:

$$E: \mathcal{X} \mapsto \mathcal{D}_n, E(x) = \rho_x$$

2) Evolution Function:

$$\mathcal{U}: \mathbb{C}^{2^n \times 2^n} \to \mathbb{C}^{2^m \times 2^m}, \mathcal{U}(\rho_x) = \widetilde{\rho_x}$$

3) Decision Rule:

$$\hat{y} : \mathbb{C}^{2^m \times 2^m} \to \{0, 1\}$$

$$\hat{y}[\widetilde{\rho_x}] = \begin{cases} 0 & \text{if } \mathrm{Tr}[\prod_0^c \widetilde{\rho_x}] \geq 1/2 \\ 1 & \text{otherwise} \end{cases}$$

decision boundary

Noise in Quantum System

- (**Error**) Errors occur on a qubit when its evolution differs from the desired one. This difference can occur due to **imprecise control** or by **interaction of the qubit with an environment**.

  Everything external to the qubit of interest

- (**Quantum Channel**) Formal description of *how qubits are affected by their environment*.
- (**Pauli Channel**) The Pauli channel maps a single qubit state ρ to $\mathcal{E}_{\boldsymbol{p}}^{p}(\rho)$ defined by $\mathcal{E}_{\boldsymbol{p}}^{p}(\rho) = p_I\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z$, where $p_I + p_X + p_Y + p_Z = 1$.
- (**Depolarizing Channel**) Pauli Channel with $p_X = p_Y = p_Z = p$ and $p_I = 1 - 3p$.

Robust Data Encoding

- (**The effect of noise**) Noise shifts data points on the Bloch sphere

- (**Robust Data Point**) Let $\mathcal{E}$ be a quantum channel, and consider a (binary) quantum classifier with the decision rule $\hat{y}$. We say that the state $\rho_x \in \mathcal{D}_n$ encoding a data point $x \in \mathcal{X}$ is a robust point of the quantum classifier iff $\hat{y}[\mathcal{E}(\widetilde{\rho_x})] = \hat{y}[\widetilde{\rho_x}]$ where $\widetilde{\rho_x}$ is evolved state by $\mathcal{U}$.

$$\hat{y}[\mathcal{E}_2(\mathcal{U}(\mathcal{E}_1(\rho_x)))] = \hat{y}[\widetilde{\rho_x}]$$

- (**Robust Data Set**) The set of robust data points, or simply robust data set is $\mathcal{R}(\mathcal{E}, E, \hat{y}) \coloneqq \{x \in \mathcal{X} : \hat{y}[\mathcal{E}(\widetilde{\rho_x})] = \hat{y}[\widetilde{\rho_x}]\}$

Robust Data Encoding

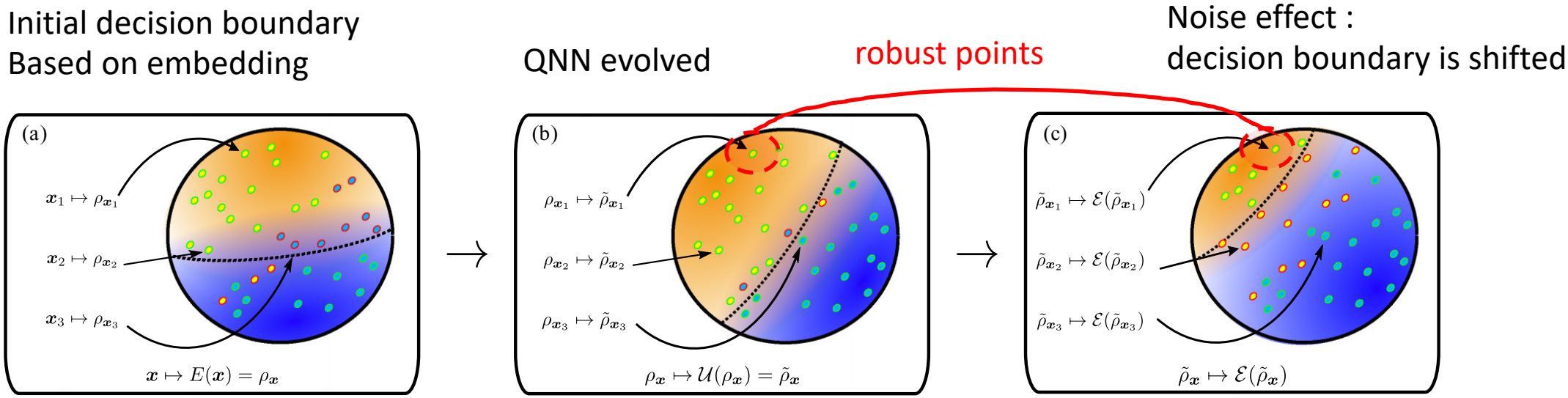


FIG. 3. Cartoon illustration of robust points for a single qubit classifier. In panel (a), input training data points $\boldsymbol{x}_i$ with classes $y_i \in$ {yellow, blue} are mapped into quantum states $\rho_{\boldsymbol{x}_i}$ according to some encoding function E. The dashed line through the Bloch sphere indicates the initial decision boundary. Points with a green outline are classified correctly, while points with a red outline are misclassified. In (b), data points are processed by the QNN with optimal unitary parameters (after minimizing a cost function to find such parameters). For clarity, we keep data points fixed and adjust the location of the decision boundary, which is now rotated to correctly classify more points (fewer points with red outlines). In (c), a noise process $\mathcal{E}$ occurs which shifts the location of the final processed points (or location of decision boundary), causing some points to be misclassified. The set of points which maintain the same classification in (b) and (c) are the robust points. Example points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are correctly classified in (a) and (b) then misclassified in (c) due to the noise. Example point $\boldsymbol{x}_3$ is incorrectly classified in (a), correctly classified in (b) after propagating through the QNN, and remains correctly classified in (c).

Robust Data Encoding

- (**Completely Robust Data Encoding**)
  We say that E is a ***completely robust data encoding*** for the quantum classifier iff $\mathcal{R}(\mathcal{E}, E, \hat{y}) = \mathcal{X}$.
  But, in practice, the robustness is determined relative to the training set.
  Therefore, we empirically observe that E is a completely robust data encoding for the quantum classifier iff $\mathcal{R}(\mathcal{E}, E, \hat{y}) = \{x_i\}_{i=1}^{M}$.

- (**Partially Robust Data Encoding**)
  We say that E is a ***partially robust data encoding*** for the quantum classifier iff $\mathcal{R}(\mathcal{E}, E, \hat{y}) \subset \mathcal{X}$.
  For $0 \leq \delta \leq 1$, we say that E is a ***δ-robust data encoding*** iff $|\mathcal{R}(\mathcal{E}, E, \hat{y})| = \delta M$

Robust Data Encoding

Theorem 1. Let $\mathcal{E}_{\mathbf{p}}^{\mathrm{P}}$ be a Pauli channel (19) and consider a quantum classifier on data from the set $\mathcal{X}$. Then, for any encoding $E : \mathcal{X} \to \mathcal{D}_2$, we have complete robustness

$$\mathcal{R}\big(\mathcal{E}_{\mathbf{p}}^{\mathrm{P}}, E, \hat{y}\big) = \mathcal{X}$$

if $p_X + p_Y \leqslant 1/2$. (Recall that $\boldsymbol{p} = [p_I, p_X, p_Y, p_Z]$.)

Theorem 6. Given a data point $\boldsymbol{x} \in \mathcal{X}$, a trace-preserving quantum channel $\mathcal{E}$, and decision rule $\hat{y}$ defined in (5), there exists an encoding E such that

$$\hat{y}[\mathcal{E}(E(\boldsymbol{x}))] = \hat{y}[E(\boldsymbol{x})]. \tag{63}$$

Ansatz

Ansatz (가설 풀이)

- 물리학과 수학에서 **가설풀이** (독일어: Ansatz) 는 어떤 주어진 문제를 풀기 위하여 그 해에 대하여 세우는 가설이다. (위키피디아)

- "**Ansatz**" : A **trial** or **approximation** of the form of **the wavefunction** that describes a quantum system. The goal of employing an ansatz is to simplify the mathematical description of a quantum state or a quantum algorithm by making a reasonable guess about the structure of the wavefunction. (ChatGPT)

- **"Ansatz Circuit"** : The core component of circuit-based quantum machine learning models is a *parameterized quantum circuit* called ansatz.

Parameterized Quantum Circuit

- Parameters are encoded into a unitary $U(\theta)$ that is applied to the input state,

$$U(\theta) = U_L(\theta_L) \cdots U_1(\theta_1) \text{ with}$$

$$U_l(\theta_l) = \prod_m e^{-i\theta_m H_m} W_m.$$

Here, W_m is an unparameterized unitary and H_m is a Hermitian operator

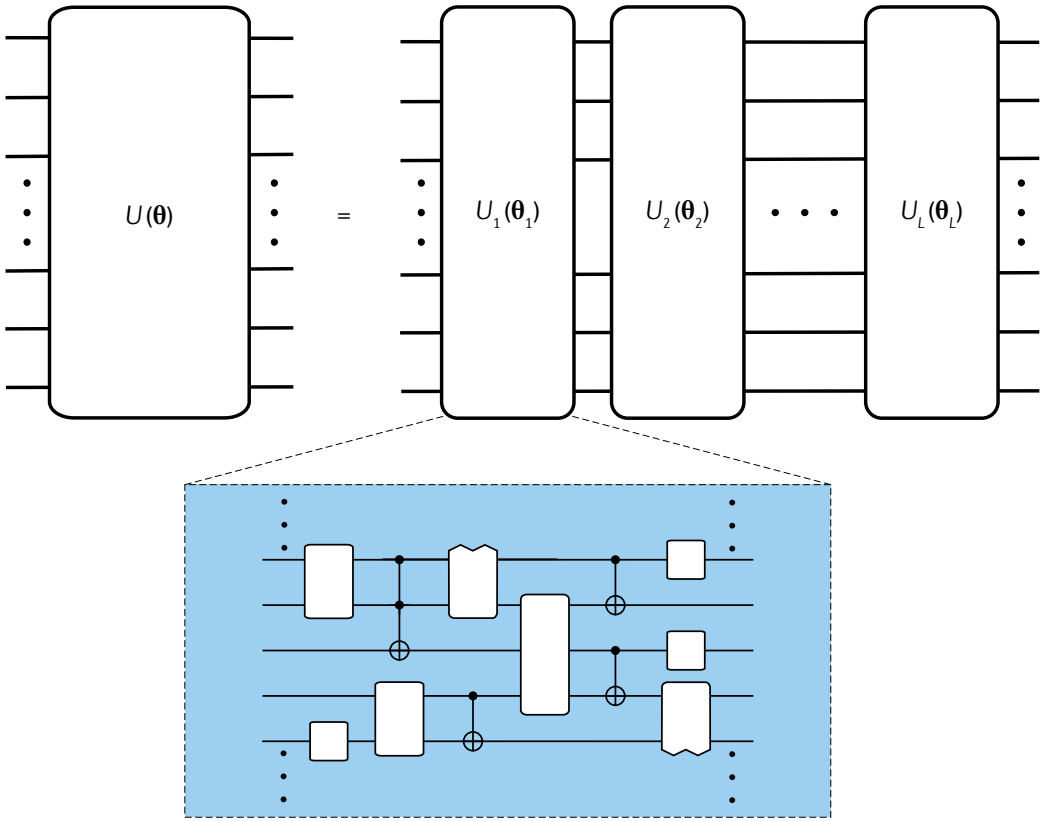
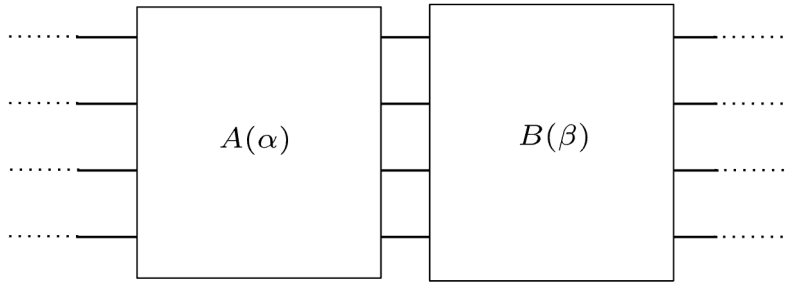


Fig. 3 | **Schematic diagram of an ansatz.** The unitary $U(\boldsymbol{\theta})$, with $\boldsymbol{\theta}$ a set of parameters, can be expressed as a product of L unitaries $U_l(\theta_l)$ sequentially acting on an input state. As indicated, each unitary $U_l(\theta_l)$ can in turn be decomposed into a sequence of parameterized and unparameterized gates.

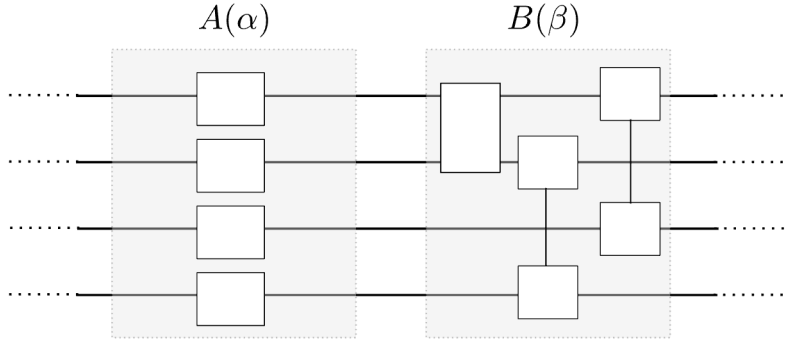
<source: nature reviews 3, 625>

Layered Gate Ansatz

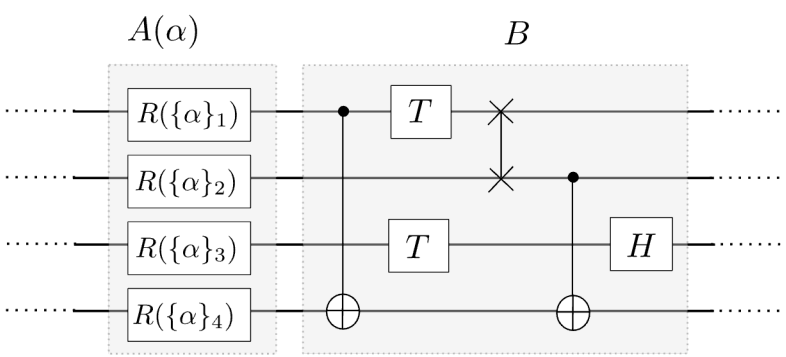
- A layer is a sequence of gates that is repeated. The number of repetitions of a layer forms a hyperparameter of the variational circuit. The layer can be decomposed into two overall unitaries A and B.



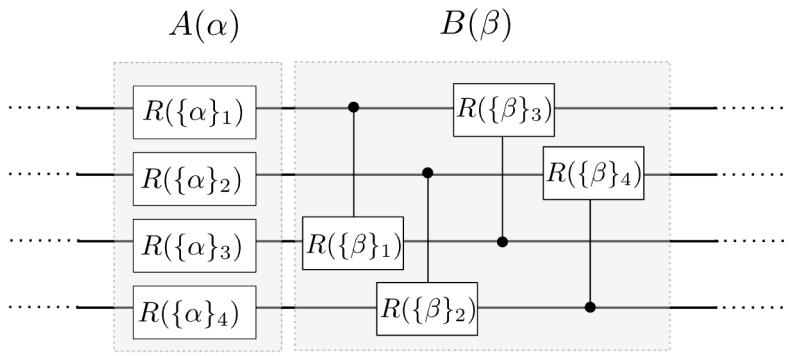
- Block A contains single-qubit gates applied to every subsystem or wire (qubits). Block B consists of both single-wire gates as well as entangling gates



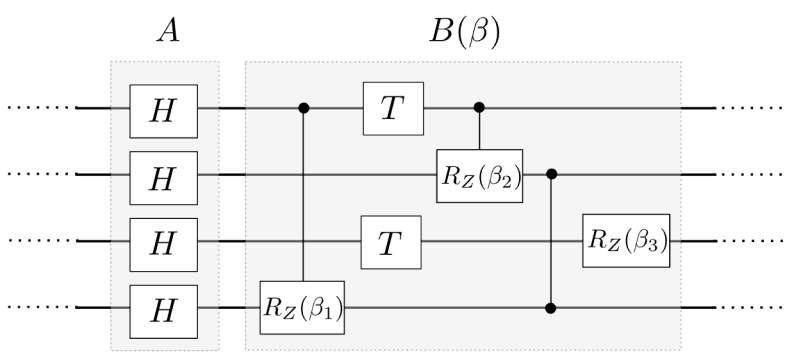
Layered Gate Ansatz



<A parametrized, B fixed>



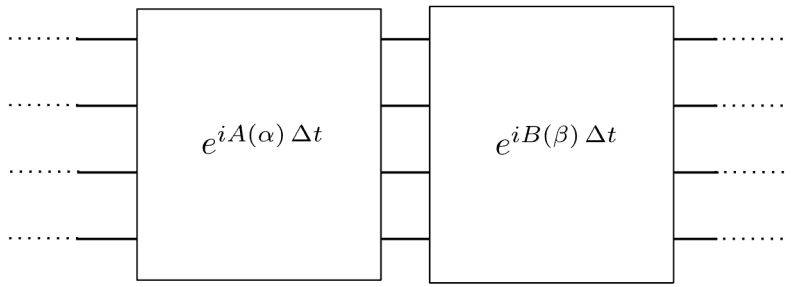
<A parameterized, B parameterized>



<A fixed, B parametrized>

Alternating Operator Ansatz

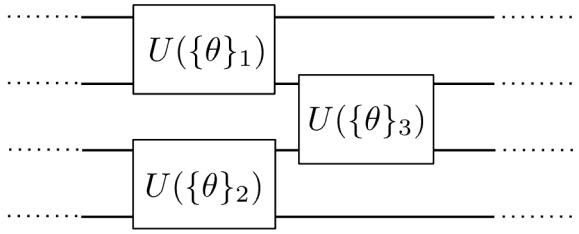
- We use layers of two blocks, but the difference is that here we apply the unitaries representing the Hamiltonians A and B which are evolved for a short time Δt.



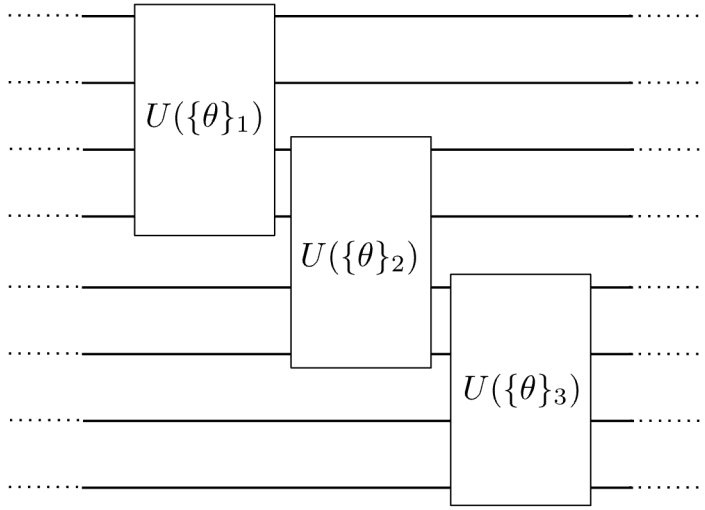
- The idea of this ansatz is based on analogies to adiabatic quantum computing, in which the system starts in the ground state of A and adiabatically evolves to the ground state of B. Quickly alternating applications of A and B for very short time Δt can be used as a heuristic to approximate this evolution

Tensor Network Ansatz

- Gate sequence inspired by tensor networks. The simplest one is a **tree architecture** that consecutively entangles subsets of qubits.



- Another tensor network is based on **matrix product states**. The circuit unitaries can be decomposed in different ways, and their size corresponds to the “bond dimension” of the matrix product state – the higher the bond dimension, the more complex the circuit ansatz.



Hardware efficient (friendly) Ansatz

- The hardware efficient ansatz is a generic name used for ansatzes that are aimed at reducing the circuit depth needed to implement $U(\theta)$ when using a given quantum hardware.
- One uses unitaries W_m and $e^{-i\theta_m H_m}$ that are taken from **a gate set determined from the connectivity and interactions specific to a quantum hardware** which avoids the circuit depth overhead arising from translating an arbitrary unitary into the sequence of native gates

Ansatz Expressibility

- Given the wide range of ansatzes one can use, a relevant question is whether a given architecture can prepare a target state by optimizing its parameters.
- Two ways to judge the quality of an ansatz: **expressibility** and **entangling capability**
- An ansatz is *expressible* if the circuit can be used to uniformly explore the entire space of a quantum state. One way to quantify the **expressibility** of an ansatz $U(\theta)$ is to compare the distribution of states obtained from $U(\theta)$ to the maximally expressive uniform (Haar) distribution of states U_{Haar},
$A^{(t)}(U) = \int dU_{Haar} U_{Haar}^{\otimes t} |0\rangle\langle 0| \left(U_{Haar}^{\dagger}\right)^{\otimes t} - \int dU U^{\otimes t} |0\rangle\langle 0| \left(U^{\dagger}\right)^{\otimes t}$.
- A measure of **entangling capability** for ansatz quantifies the average entanglement of states produced from randomly sampling the circuit parameter θ.


Thank you