# Lecture 5: Binary Patterns and Associative Memory

Andreas Wichert

Department of Computer Science and Engineering

Técnico Lisboa

# Overview

- Associative Memory
- Binary Vectors
- Ventura and Martinez Method
  - QuAM
  - Ventura Martinez Trick
- Entanglement of Binary Patterns
- Lernmatrix (Willshaw's associative memory)
  - Monte Carlo Lernmatrix
  - Quantum Counting of Ones
  - Quantum Lernmatrix
- Open Problems

# Associative Memory

- The ability to correct faults if false information is given
- To complete information if some parts are missing
- To interpolate information, that means if a pattern is not stored the most similar stored pattern is determined

- How about quantum associative memory?

The cerebral cortex is a huge associative memory

# Binary Vectors

- A set of binary vectors is represented by the basis encoding
- A uniform superposition of binary vectors can be easily generated by Hadamard gates

$$H_2|00\rangle = H_1|0\rangle \otimes H_1|0\rangle = \frac{1}{\sqrt{2^2}} \sum_{x \in B^2} |x\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

with the amplitude representation

$$\alpha = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}.$$

But how to generate the superposition

$$\frac{1}{\sqrt{3}}\left(|10\rangle + |01\rangle + |11\rangle\right)$$

with the amplitude representation

$$\alpha = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix}$$

# Ventura and Martinez Method

Dan Ventura and Tony Martinez are professors of Computer Science at Brigham Young University

- To generate a superposition of $m$ binary linear independent vectors with dimension $n$ with $m < 2^n$ a method was proposed by Ventura and Martinez
- The procedure is based on successively dividing present superposition into processing and memory branches
- Into each new generated memory branch an input pattern is loaded step by step
- The method is linear in the number of stored patterns and their dimension with $O(n \cdot log_2\, m)$

- At the initial step the system is in the basis state with load qubits, memory qubits and the control qubits $c_1$, $c_2$

$$|memory; c_2, c_1; load\rangle$$

- The idea is to **split** this basis state into a **new superposition** step by step using the control register until the required superposition is present

$$\frac{1}{\sqrt{m}} \sum_{j=1}^{m} |memory; c_2, c_1; load\rangle_j$$

with the memory register in the required superposition

$$\frac{1}{\sqrt{m}} \sum_{j=1}^{m} = |memory; 0, 0; 0 \cdots 0\rangle_j = \left( \frac{1}{\sqrt{m}} \sum_{j=1}^{m} |memory\rangle_j \right) \otimes |0, 0; 0 \cdots 0\rangle,$$

- We make a distinction between a processing branch indicated by the control qubit $c_2$ with the value one ($c_2 = 1$) and the memory branches in superposition with the control qubit $c_2$ with the value zero ($c_2 = 0$)

$$|memory; c_2, c_1; load\rangle$$

$$\frac{1}{\sqrt{3}}|memory; 0\,1; load\rangle + \sqrt{\frac{2}{3}}|memory; 1\,1; load\rangle$$

memory register — processing branch

- The control qubit $c_1 = 1$ indicates the split of the qubit $c_2$ by the operator $CS_p$ represented by the parametrized $U$ gate

- The control qubit $c_1 = 1$ indicates the split of the qubit $c_2$ by the operator $CS_p$ represented by the parametrized $U$ gate

$$CS_p = CU(\arcsin\left(\frac{1}{\sqrt{p}}\right) \cdot 2, \pi, \pi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{p-1}{p}} & \frac{1}{\sqrt{p}} \\ 0 & 0 & \frac{-1}{\sqrt{p}} & \sqrt{\frac{p-1}{p}} \end{pmatrix}$$

with $CS_p|c_2, c_1\rangle$

$$CS_p|01\rangle = |01\rangle, \quad CS_p|11\rangle = \frac{1}{\sqrt{p}} \cdot |10\rangle + \sqrt{\frac{p-1}{p}} \cdot |11\rangle.$$

split operation

# Example

In this example we store three binary patterns,

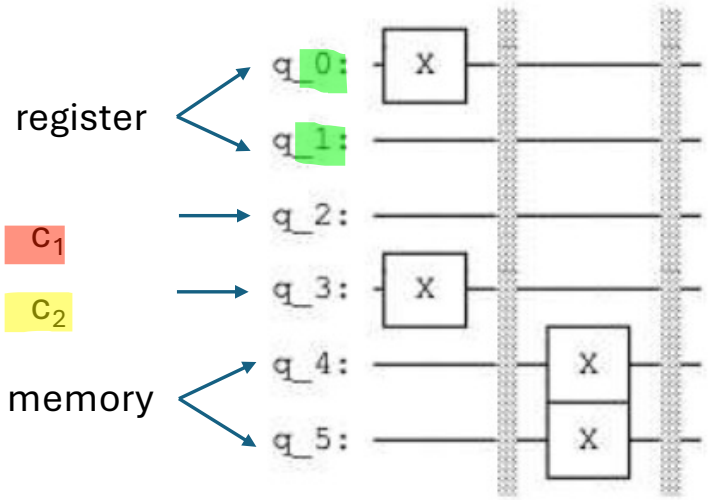
$$|01\rangle_3, \quad |10\rangle_2, \quad |11\rangle_1$$

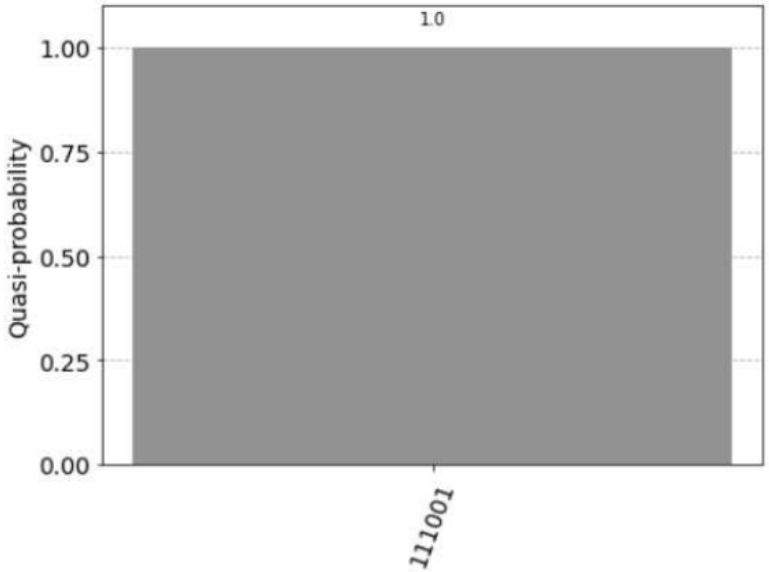
*Qiskit* uses little endian notation

$$|memory; c_2, c_1; load\rangle = |memory; c_2, c_1; register\rangle = |q_5, q_4, q_3, q_2; q_1, q_0\rangle,$$

Qubts 0 and 1 represent the load register, qubits 2 and 3 are the control qubits and the qubits 4 and 5 represent the memory register. We use the *statevector simulator*

$$|memory; c_2, c_1; register\rangle$$

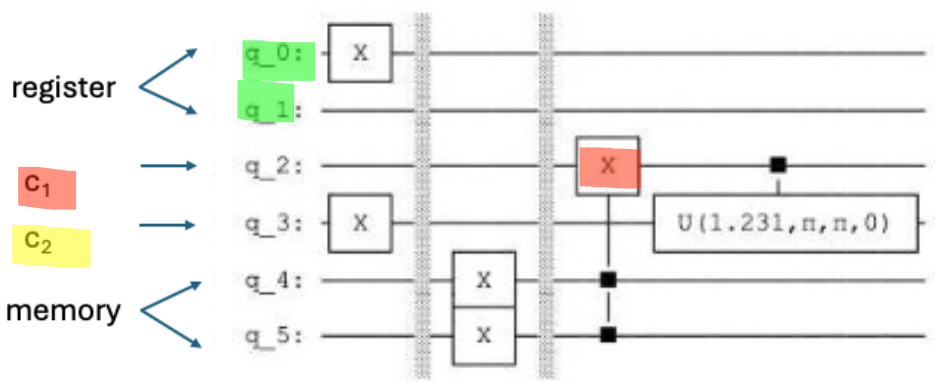


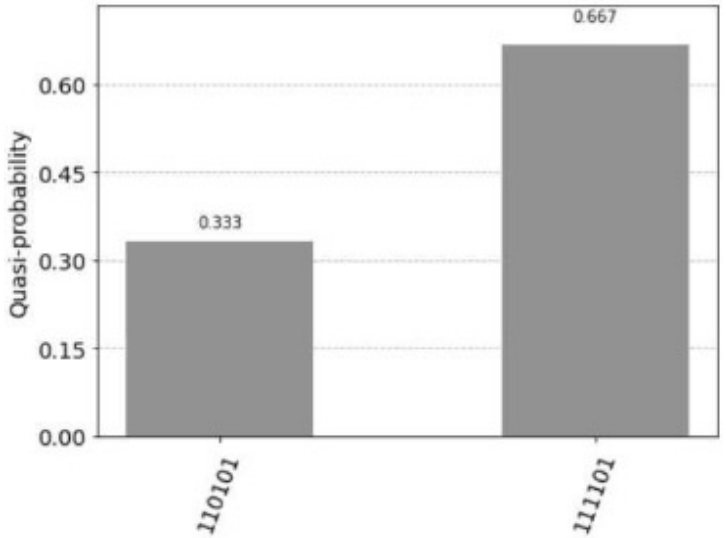




- The qubit 3 is set to one indicating by $c_2 = 1$, that the basis state represents a processing branch.
- In the load register pattern $|01\rangle$ is generated and the memory register is set to $|11\rangle$.
- The resulting state is

$$|\psi\rangle_1 = |1,1;1,0;0,1\rangle$$

$$|memory; c_2, c_1; register\rangle$$

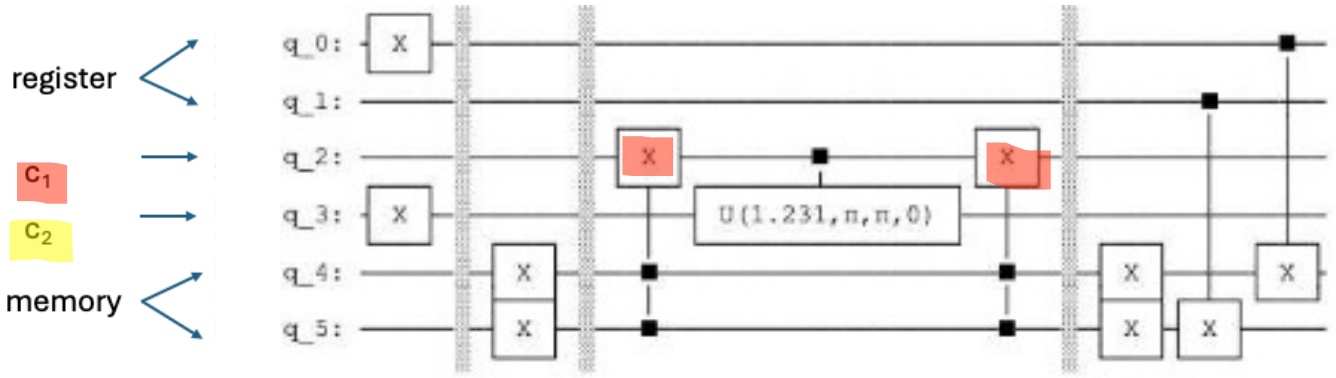



- The control qubit *2 ($c_1 = 1$)* is **entangled** with the memory register $|11\rangle$ by the multi-controlled X (Toffoli) gate
- The processing branch is split by the operator $CS_3$ *(p=3)*,creating a new memory and processing branch

$$|\psi\rangle_2 = \frac{1}{\sqrt{3}}|1,1;0,1;0,1\rangle + \sqrt{\frac{2}{3}}|1,1;1,1;0,1\rangle.$$

memory register  processing branch

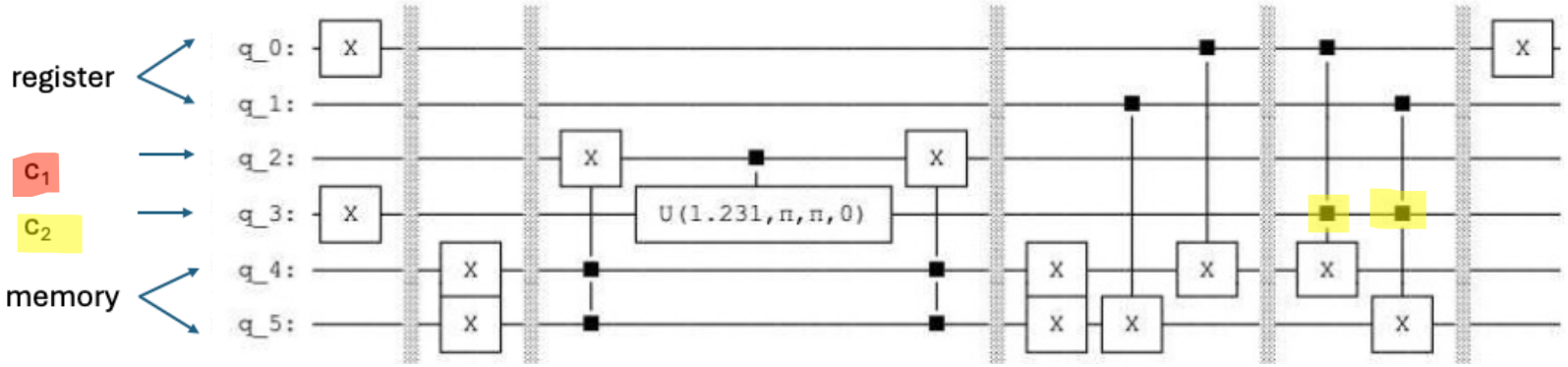



- We un-compute the entanglement of control qubit 2 ($c_1 = 1$) with the memory register $|11\rangle$
- We apply the NOT gates operation and the controlled NOT operation (CNOT gate) to the memory register of both branches.
- As result we write *01* into the memory registers.

$$|\psi\rangle_3 = \frac{1}{\sqrt{3}}|0,1;0,0;0,1\rangle + \sqrt{\frac{2}{3}}|0,1;1,0;0,1\rangle.$$

memory register  processing branch

load is reseated to the ground state

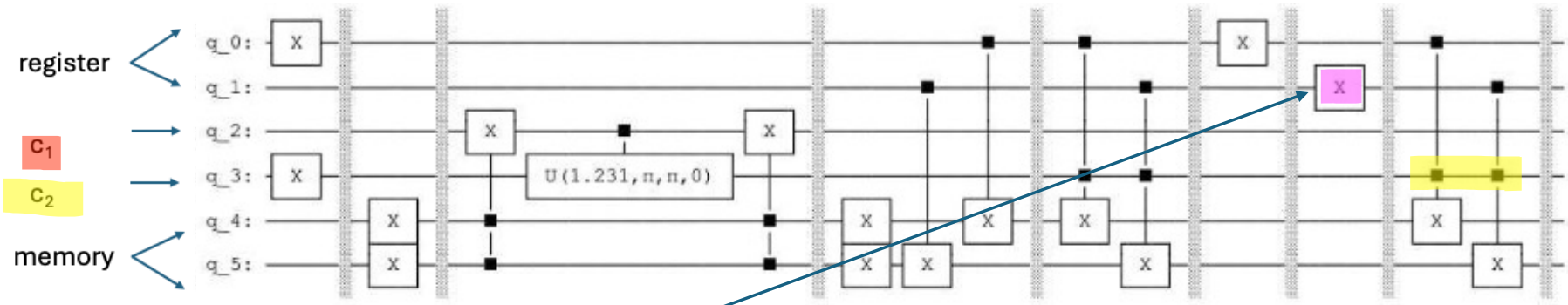



- We un-compute the memory register of the processing branch, setting it to the ground state $|00\rangle$
- We use the ccX gate controlled by the load register and the control qubit 3 ($c_2 = 1$ indicates the processing branch)
  - We reset the load register to the ground state $|00\rangle$

*First pattern is stored!*

$$|\psi\rangle_4 = \frac{1}{\sqrt{3}}|0,1;0,0;0,0\rangle + \sqrt{\frac{2}{3}}|0,0;1,0;0,0\rangle.$$

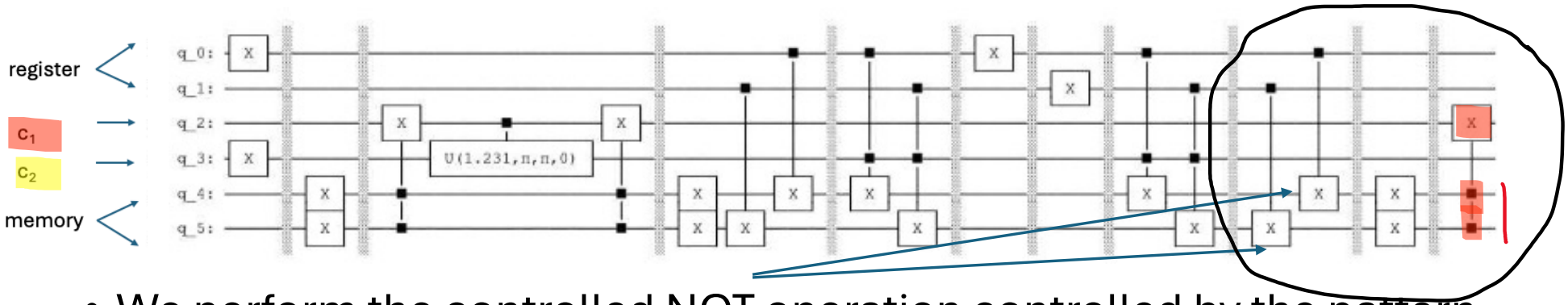
memory register  processing branch




- The pattern $|10\rangle$ is generated in the load register
- We copy $|10\rangle$ into the memory register of the processing branch (the memory register before the operation is in the ground state)
  - We use the ccX gate controlled by the load register and the control qubit 3 ($c_2 = 1$ indicates the processing branch)

$$|\psi\rangle_5 = \frac{1}{\sqrt{3}}|0,1;0,0;1,0\rangle + \sqrt{\frac{2}{3}}|1,0;1,0;1,0\rangle.$$

memory register  processing branch




- We perform the controlled NOT operation controlled by the pattern |10⟩ with the memory register of both branches using cX gate
  - As a result, the memory register of the processing branch is in the ground state |00⟩
  - This is not the case for the memory register of memory branch, where the bits are flipped leading to the state |11⟩.
- We apply the NOT operation to the memory register of both branches.
  - As a result, the memory register of the processing branch is in the state |11⟩
  - The control qubit 2 ($c_1 = 1$) is entangled with the memory register |11⟩

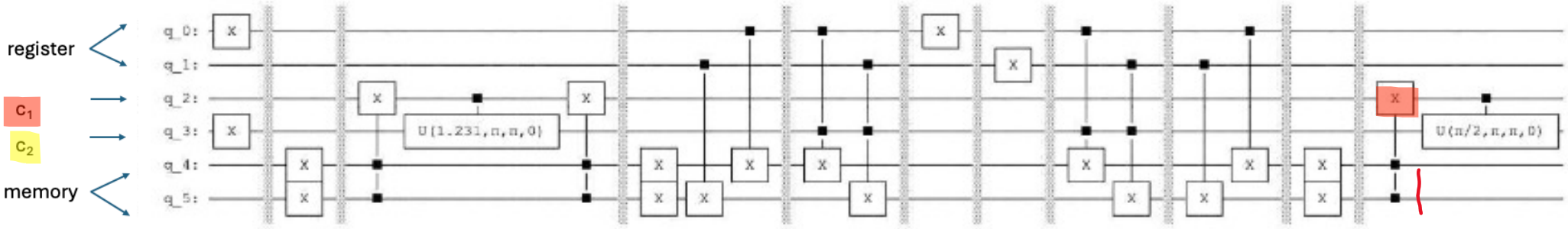
with first pattern → |11⟩ with **itself** → |00⟩ NOT

$$|\psi\rangle_5 = \frac{1}{\sqrt{3}}|0,1;0,0;1,0\rangle + \sqrt{\frac{2}{3}}|1,0;1,0;1,0\rangle.$$

memory register processing branch

$$|\psi\rangle_6 = \frac{1}{\sqrt{3}}|0,0;0,0;1,0\rangle + \sqrt{\frac{2}{3}}|1,1;1,1;1,0\rangle.$$

memory register processing branch

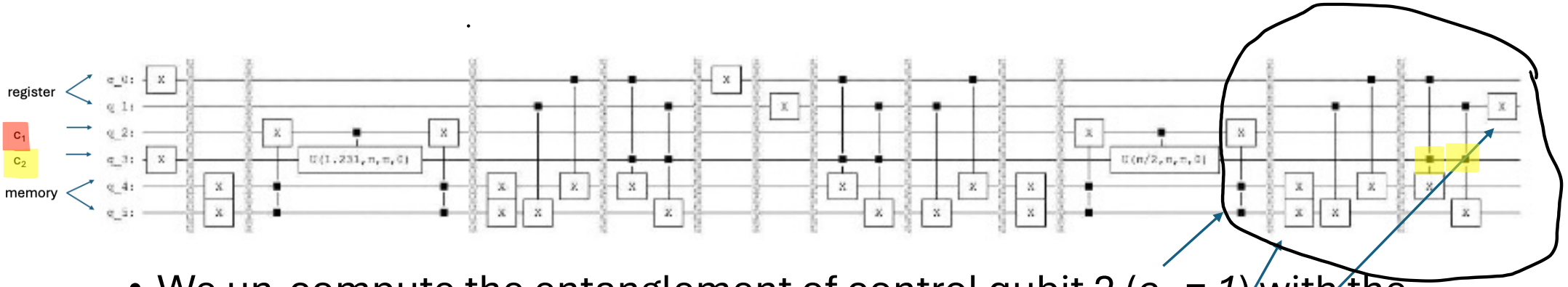



- The control qubit 2 ( $c_1 = 1$ ) is entangled with the memory register $|11\rangle$ of the processing ranch by the multi-controlled X gate
- The processing branch is split by the operator $CS_2$ ,creating a new memory branch

$$|\psi\rangle_6 = \frac{1}{\sqrt{3}}|0,0;0,0;1,0\rangle + \sqrt{\frac{2}{3}}|1,1;1,1;1,0\rangle.$$

$$|\psi\rangle_7 = \frac{1}{\sqrt{3}}|0,0;0,0;1,0\rangle + \frac{1}{\sqrt{3}}|1,1;0,1;1,0\rangle + \frac{1}{\sqrt{3}}|1,1;1,1;1,0\rangle.$$

memory register　　processing branch



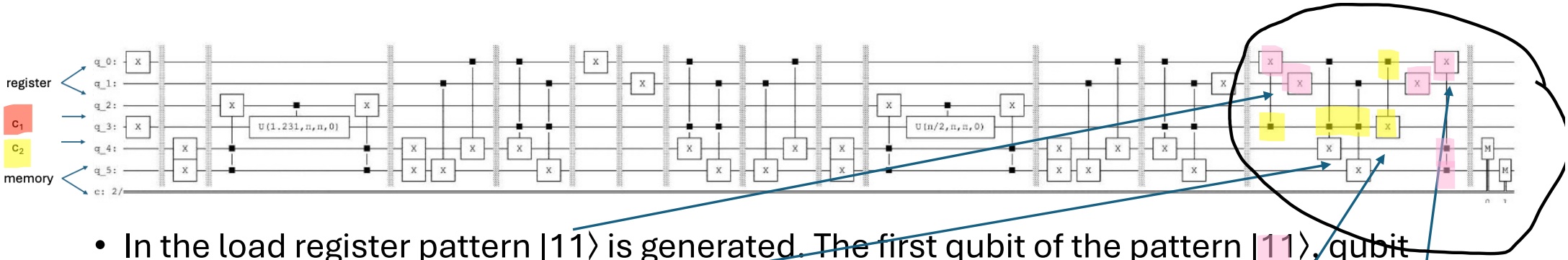

- We un-compute the entanglement of control qubit 2 ($c_1 = 1$) with the memory register $|11\rangle$ (using the multi-controlled X (Toffoli) gate).
  - We apply the NOT gates operation and the controlled NOT operation to the memory register of the three branches.
  - As result we un-flip (recover) the memory register of the first memory branch to $|01\rangle$ and copy $|10\rangle$ into the memory register of the other two branch
  - We un-compute the register of the processing branch, setting it to the the ground state $|00\rangle$.
    - We use the ccX gate controlled by the load register and the control qubit 3 ($c_2 = 1$ indicates the processing branch)
  - We reset the load register to the ground state $|00\rangle$.

$$|\psi\rangle_7 = \frac{1}{\sqrt{3}}|0,0;0,0;1,0\rangle + \frac{1}{\sqrt{3}}|1,1;0,1;1,0\rangle + \frac{1}{\sqrt{3}}|1,1;1,1;1,0\rangle.$$

$$|\psi\rangle_8 = \frac{1}{\sqrt{3}}|0,0;1,0;0,0\rangle + \frac{1}{\sqrt{3}}|0,1;0,0;0,0\rangle + \frac{1}{\sqrt{3}}|1,0;0,0;0,0\rangle.$$

processing branch

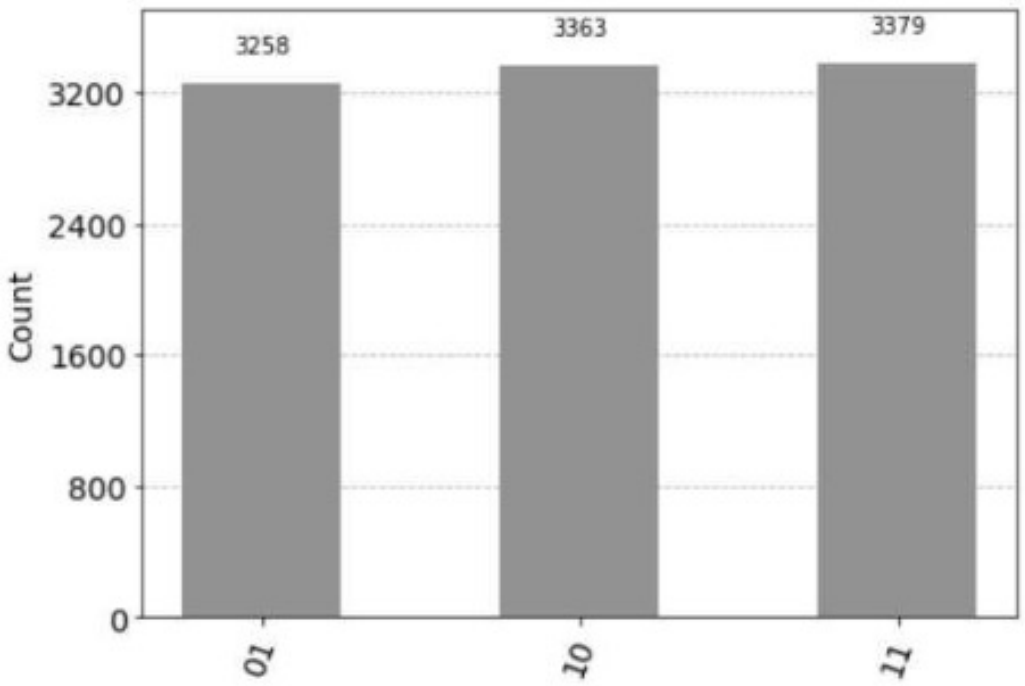
memory register




- In the load register pattern $|11\rangle$ is generated. The first qubit of the pattern $|11\rangle$, qubit 0, is entangled with the qubit 3 of control register $c_2 = 1$ using the controlled NOT gate cX.
- We copy $|11\rangle$ into the memory register of the processing branch by the ccX gate (CCNOT gate) controlled by the load register and the control qubit 3 ($c_2 = 1$ indicates the processing branch).
- We convert the processing branch into a memory branch by setting the qubit 3 of the control register $c_2$ to zero by the entangled qubit 0 of the pattern $|11\rangle$ (using controlled NOT gate cX).
- We reset the load register to the ground state $|00\rangle$
- The entangled qubit in the load register is set to zero by the ccX gate (CCNOT gate) with the control qubits represented by the memory register of the stored pattern $|11\rangle$
- We measure the memory register: qubit 4 and 5

$$|\psi\rangle_9 = \frac{1}{\sqrt{3}} \cdot (|0,1\rangle + |1,0\rangle + |1,1\rangle) \otimes |0,0;0,0\rangle.$$

memory register




- We measure the memory register: qubit 4 and 5 using the qasm simulator with shots = 10000

# QuAM

- In the quantum associative memory (QuAM) as proposed by Venture and Martinez, a modified version of Grover's search algorithm is applied to determine the answer vector to a query vector
- To generate a superposition of $m$ binary linear independent vectors with dimension $n$ with $m < 2^n$
  - *Cost* $O(n \cdot m)$ requiring $log_2(m)$ "*units*" *(qubits)*
  - The query costs are $O(n \cdot \sqrt{2^n})$ compared to $O(n \cdot m)$ on a conventional computer.

# Not Uniform Distribution

$$|\psi\rangle = \frac{1}{2} \cdot (|1100\rangle + |1001\rangle + |1111\rangle + |0110\rangle)\,.$$

so that we can track the amplitude distribution

$$\frac{1}{2}(0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,1)^T.$$

We notice that the amplitude distribution is not uniform, a uniform distribution of amplitudes of four qubits would correspond to

$$\frac{1}{4}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)^T.$$

Could this be the cause of the problem?

From the initial distribution we mark the target state $|0110\rangle$ by a negative phase

$$\frac{1}{2}(0,0,0,1,0,0,-1,0,0,1,0,0,0,0,0,1)^T.$$

and perform a Grover's rotation with the result

$$\frac{1}{8}(-1,-1,-1,3,-1,-1,-5,-1,-1,3,-1,-1,-1,-1,-1,3)^T$$

We mark again and the target state $|0110\rangle$ by a negative phase and perform a Grover's rotation with the result

$$(-0.2,-0.2,-0.2,0.3,-0.2,-0.2,0.6,-0.2,-0.2,0.3,-0.2,-0.2,-0.2,-0.2,-0.2,0.3)^T.$$

The four states that represent our distribution have high non negative values of amplitude (the results are rounded for representation clarity). After marking the target state $|0110\rangle$ by a negative phase and a Grover's rotation

$$(0.02,0.02,0.02,0.5,0.02,0.02,-0.4,0.02,0.02,0.5,0.02,0.02,0.02,0.02,0.02,0.5)^T$$

all four basis states have nearly equal distribution, the information of the target state $|0110\rangle$ is negative. If we mark the target by a negative phase the information about it will be lost. How can we deal with non uniform distributions?

# Ventura Martinez Trick

- Venture and Martinez, proposed a modified version of Grover's search algorithm
  - As before From the initial distribution we mark the target state $|0110\rangle$ by a negative phase

$$\frac{1}{2}(0,0,0,1,0,0,-1,0,0,1,0,0,0,0,0,1)^T.$$

and perform a Grover's rotation with the result

$$\frac{1}{8}(-1,-1,-1,3,-1,-1,-5,-1,-1,3,-1,-1,-1,-1,-1,3)^T$$

However, now me mark all four states that represent our distribution by a negative phase and perform a Grover's rotation with the result

$$\frac{1}{8}(1,1,1,-1,1,1,7,1,1,-1,1,1,1,1,1,-1)^T$$

We mark again all four states that represent our distribution by a negative phase and perform a Grover's rotation

$$(0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0)^T$$

with the amplitude one indicating the target state.

- The cost depend on the distribution and the relation between *m* (the number of patterns) and *n* (the dimension of patterns)
- We correct by marking all present state till all states without the target state reach a **uniform distribution**
- The correction cost influence the cost of Grover's amplification algorithm
  - In our case we need three Grover's rotation with two times the cost of marking all present states
  - For four states in superposition and one target value we would only need one rotation
- The actual costs are $\sqrt{2^n} = 2^{n/2}$
  - which gives an advantage for $2^{n/2} < m$ ☹
  - *However, we need only $\log_2 m$ units!* ☺

# Entanglement of Binary Patterns

- Is there another morse simpler method?
  - Entanglement of binary patterns

$$|\psi\rangle = \left( \frac{1}{\sqrt{m}} \sum_{j=1}^{m} |pattern_j\rangle \otimes |0 \cdots 0; 0, 0\rangle \right),$$

while the entanglement of binary pattern vectors leads to the superposition

$$|\psi\rangle = \frac{1}{\sqrt{m}} \left( \sum_{j=1}^{m} \quad |index_j\rangle |pattern_j\rangle \right).$$

  - The method of entanglement of binary pattern is easier to understand and implement an allows us to address the patterns **through the index**
  - The same pattern can be represented several times with a different index

# Entanglement of Binary Patterns

The method is based the on the entanglement of the index qubits that are in the superposition with the patterns. To stores $m$ binary paterns

$$|pattern_m\rangle, |pattern_{m-1}\rangle, \cdots |pattern_1\rangle$$

we entangle the index qubits using multi-controlled NOT gates (ccX gate or MCX-Gate). First, we generate $v = |log_2(m)$ index qubits using $v$ Hadamard gates

$$H^{\otimes^v}|0\rangle^{\otimes^v} = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} |index_j\rangle.$$

and entangle $m$ binary pattern with the index qubits with a resulting superposition

$$|\psi\rangle = \frac{1}{\sqrt{m}} \left( \sum_{j=1}^{m} \quad |index_j\rangle|pattern_j\rangle \right)$$

# Example

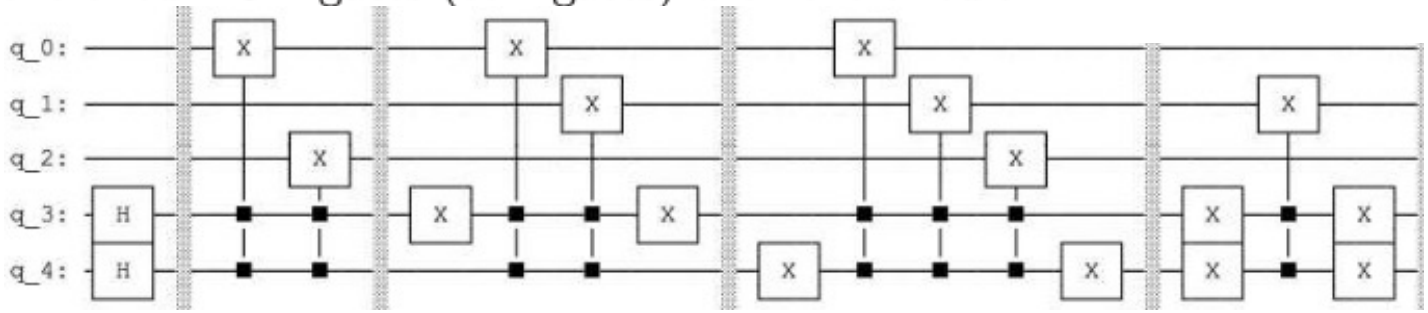
In this example we store four binary patterns,

$$|101\rangle_4, \quad |011\rangle_3, \quad |111\rangle_2, \quad |010\rangle_1$$

by entanglement with the four index qubits $|index_j\rangle$ in superposition

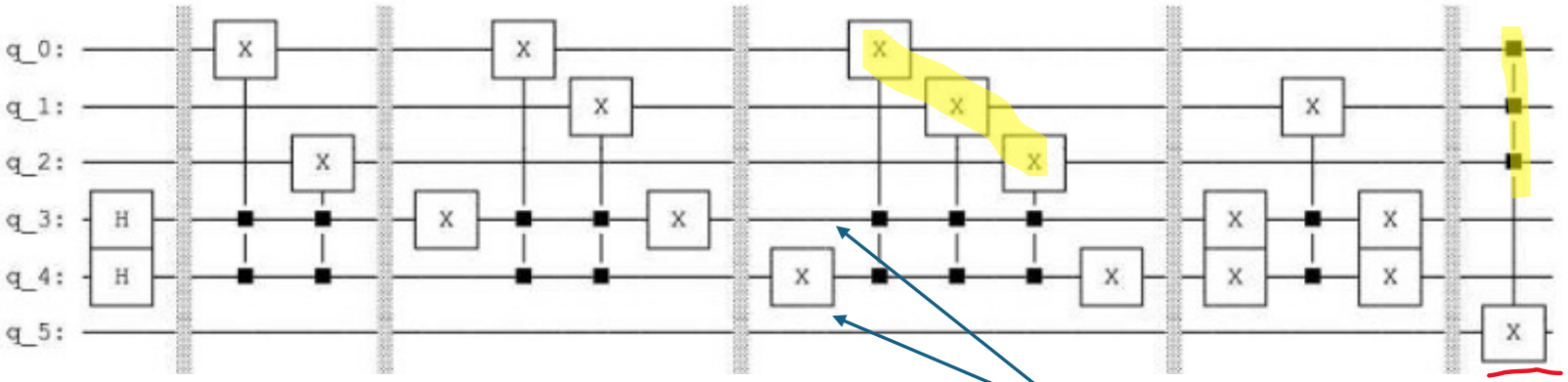
$$|index_4\rangle = |11\rangle \quad |index_3\rangle = |10\rangle \quad |index_2\rangle = |01\rangle \quad |index_1\rangle = |00\rangle$$

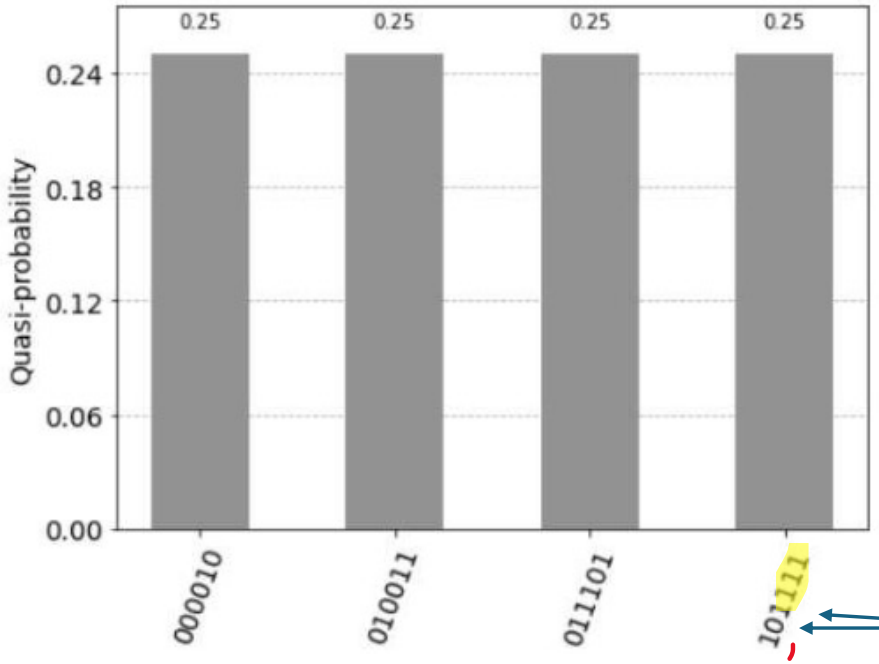
using controlled NOT gates (ccX gates) with the circuit




$$|\psi\rangle = \frac{1}{2} \cdot (|00010\rangle_1 + |01111\rangle_2 + |10011\rangle_3 + |11101\rangle_4) \,.$$

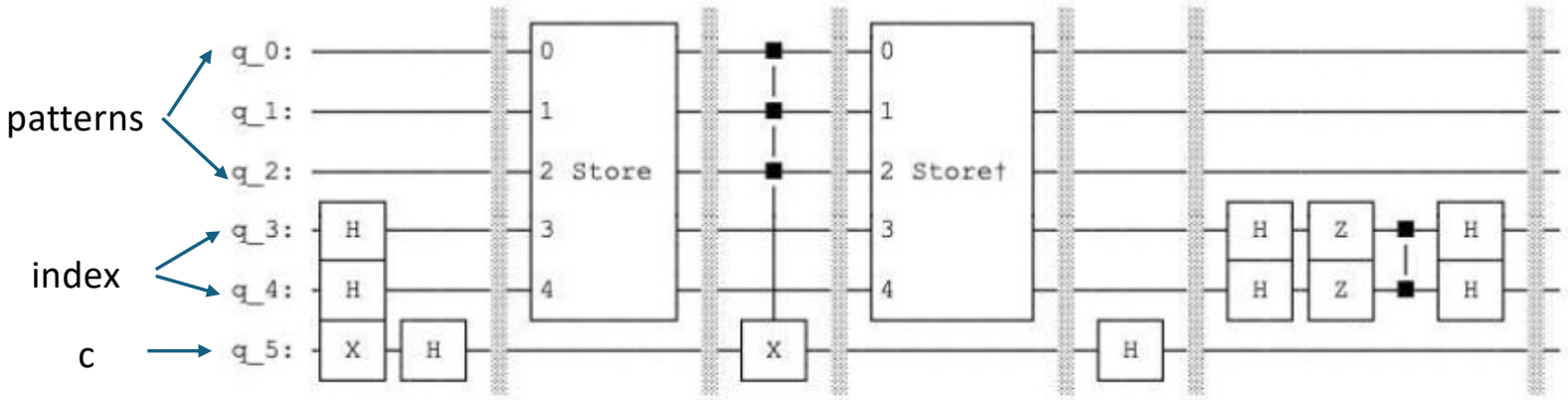
# Recall



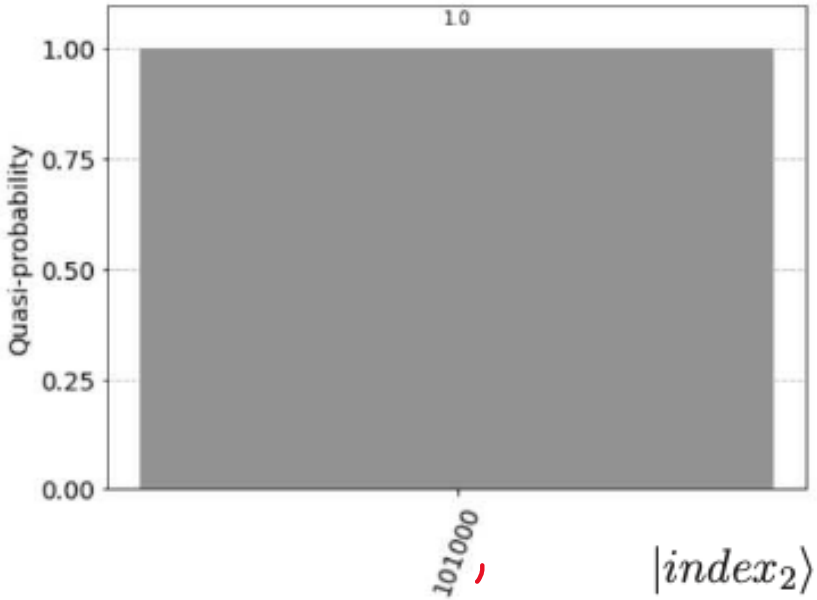

- Our query vector is represented by the pattern $|111\rangle$

$|index_2\rangle = |01\rangle$




$|index_2\rangle = |01\rangle$ :

oracle un-compute Grover's amplification to index qubits




$|index_2\rangle = |01\rangle$ :

Our query vector is represented by the pattern |111⟩

The result is the **index** of pattern |111⟩, **not the pattern itself**

- We could use instead of Grover's algorithm  Quantum Counting of Ones
- The idea of Quantum Counting and application to quantum associative memories (1 nearest neighbor, 1 NN) was first proposed by

VOLUME 87, NUMBER 6 PHYSICAL REVIEW LETTERS 6 AUGUST 2001

**Probabilistic Quantum Memories**

C. A. Trugenberger*
*InfoCodex, chemin du Petit-Saconnex 28, CH-1209 Genève, Switzerland*
(Received 26 December 2000; published 18 July 2001)

*Quantum Information Processing, Vol. 1, No. 6, December 2002 (© 2003)*

# Quantum Pattern Recognition

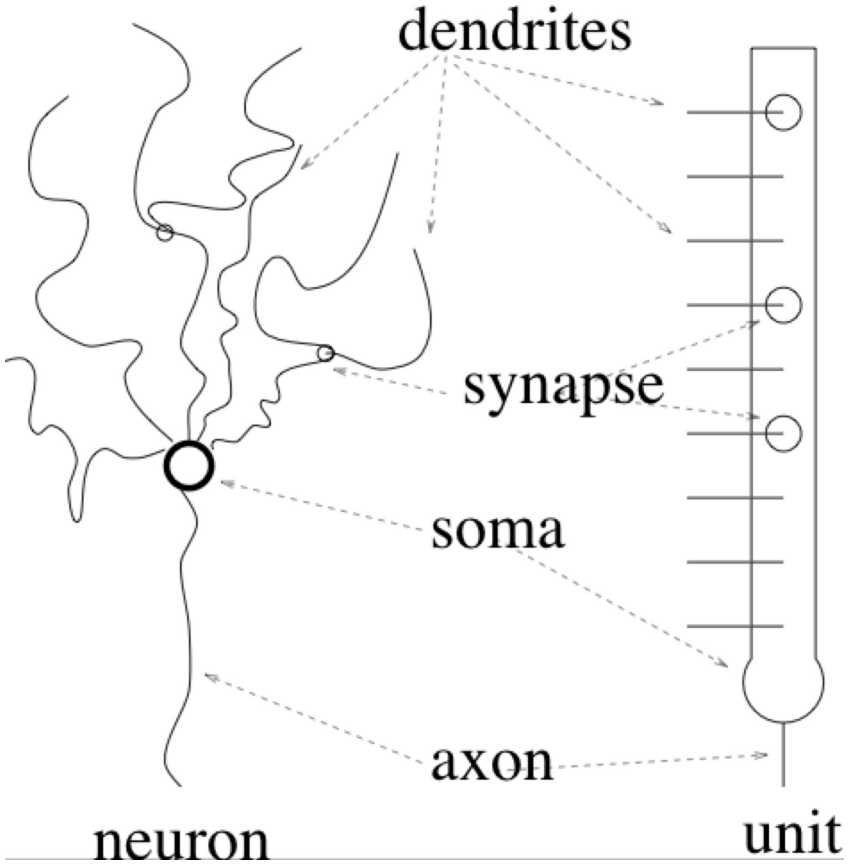
**Carlo A. Trugenberger**[1]

*Received November 4, 2002; accepted January 25, 2003*

*I review and expand the model of* quantum associative memory *that I have recently proposed. In this model binary patterns of* n *bits are stored in the quantum superposition of the appropriate subset of the computational basis of* n *qbits. Information can be retrieved by performing an input-dependent rotation of the memory quantum state within this subset and measuring the resulting state. The amplitudes of this rotated memory state are peaked on those stored patterns which are closest in Hamming distance to the input, resulting in a high probability of measuring a memory pattern very similar to it. The accuracy of pattern recall can be tuned by adjusting a parameter playing the role of an effective temperature. This model solves the well-known capacity shortage problem of classical associative memories, providing a* large improvement *in capacity.*

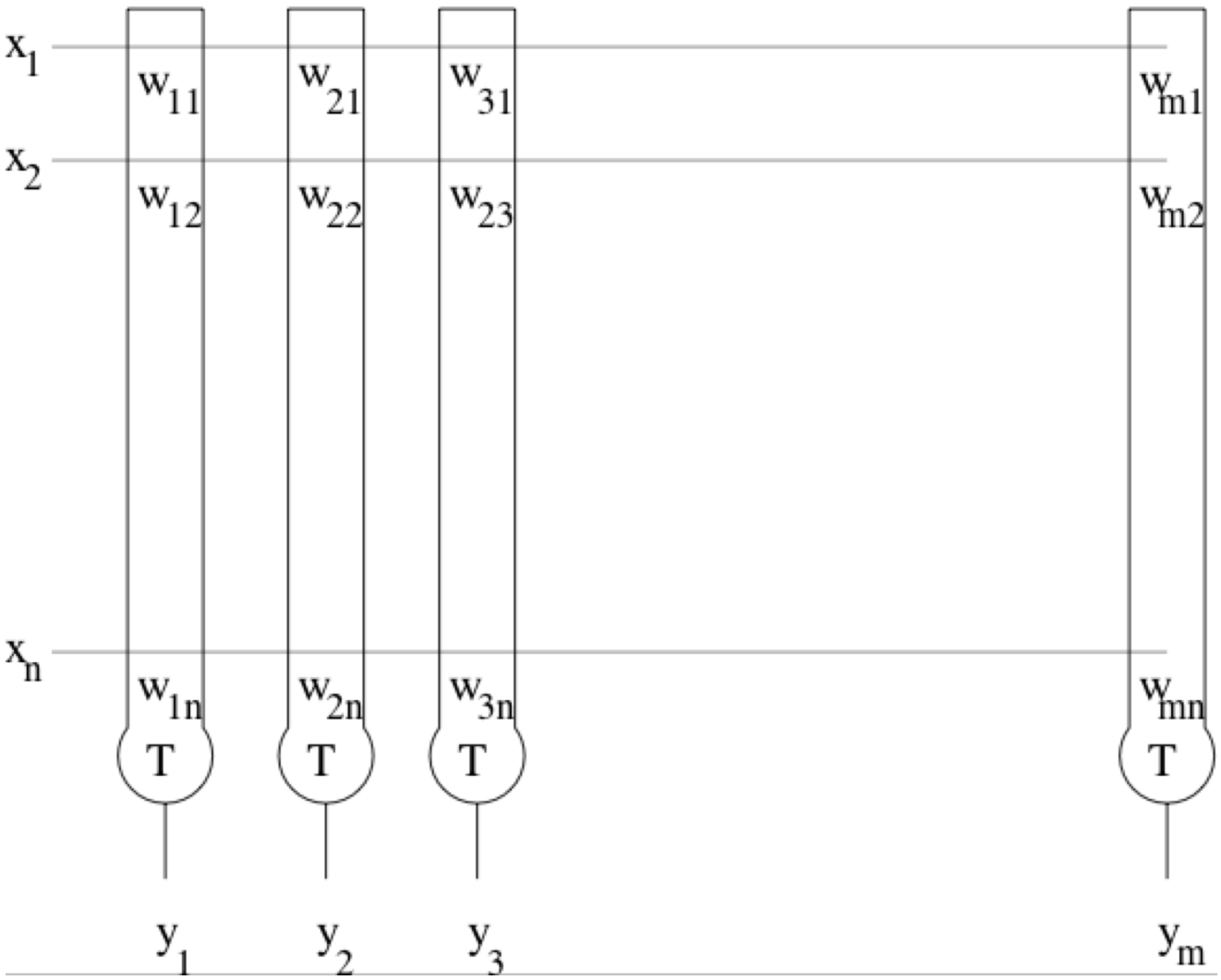
# Lernmatrix

- We will use instead of *1**NN*** the *Lernmatrix* (Willshaw's associative memory)
- The *Lernmatrix*, also simply called associative memory was developed by Steinbuch in 1958 as a biologically inspired model from the effort to explain psychological phenomenon of conditioning
- *Lernmatrix i*s composed of a cluster of units
- Each unit represents a simple model of a real biological neuron.
- Each unit is composed of binary weights, which correspond to the synapses and dendrites in a real neuron




- The *Lernmatrix* is composed of a cluster of units which represent a simple model of a real biological neuron

- Weights are described by $w_{ij} \in \{0,1\}$ and $T$ is the threshold of the unit
- The presence of a feature is indicated by a “one” component of the vector, its absence through a “zero” component of the vector.
- A pair of these vectors is associated and this process of association is called learning
- The first of the two vectors is called the query vector and the second, the answer vector
- After learning, the query vector is presented to the associative memory and the answer vector is determined by the retrieval rule in “one” step


x1
x2
xn
w11
w21
w31
wm1
w12
w22
w23
wm2
w1n
w2n
w3n
wmn
T
T
T
T
y1
y2
y3
ym

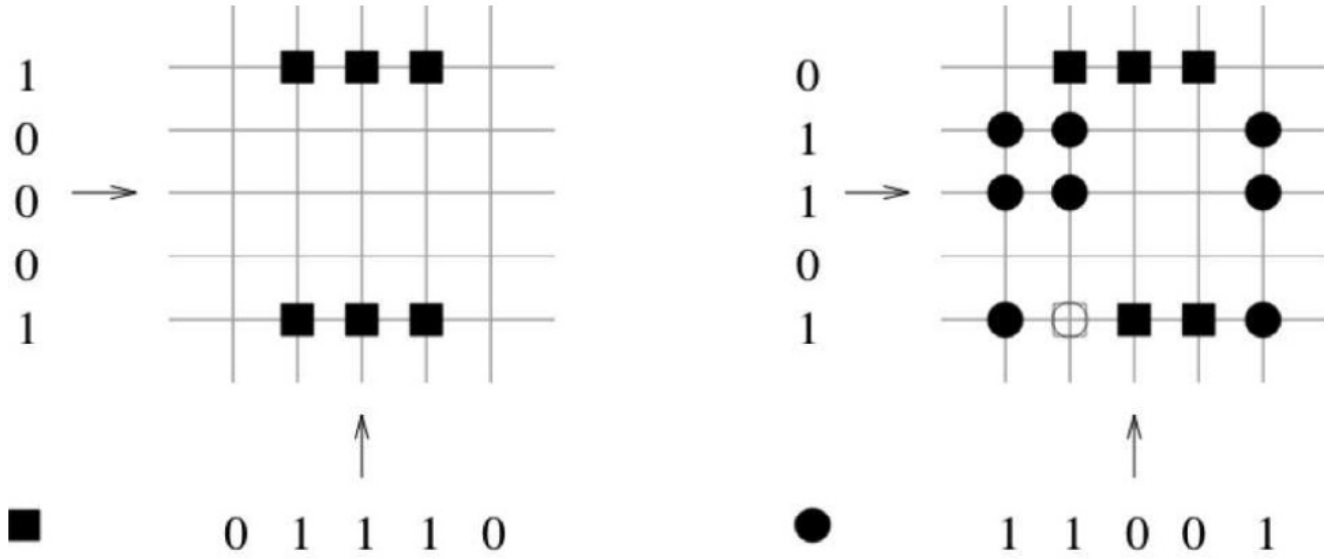
# Learning

- Initially, no information is stored in the associative memory. Because the information is represented in weights, all unit weights are initially set to zero
- In the learning phase, pairs of binary vector are associated
- Let ***x*** be the query vector and ***y*** the answer vector, the learning rule is

$$w_{ij}^{new} = \begin{cases} 1 & if \quad y_i \cdot x_j = 1 \\ w_{ij}^{old} & \text{otherwise.} \end{cases}$$

  - This rule is called the binary Hebbian rule
  - Every time a pair of binary vectors is stored, this rule is used.

# Learning



# Retrieval

- In the “one-step” retrieval phase of the associative memory, a fault tolerant answering mechanism recalls the appropriate answer vector for a query vector ***x***
- The retrieval rule for the determination of the answer vector **y** is:

$$net_i = \sum_{j=1}^{n} w_{ij} x_j,$$

$$y_i = \begin{cases} 1 & \text{if} \quad net \geq T \\ 0 & \text{otherwise.} \end{cases}$$

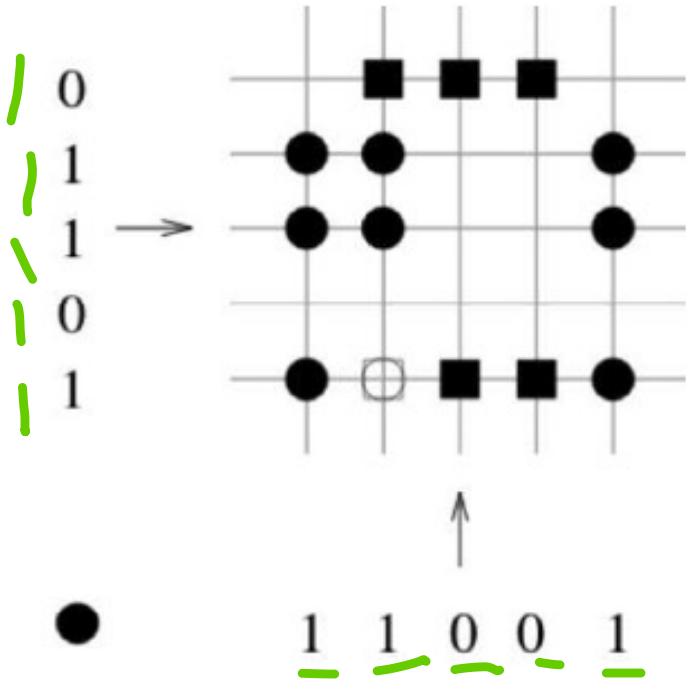
# Retrieval

$$net_i = \sum_{j=1}^{n} w_{ij} x_j,$$

$$y_i = \begin{cases} 1 & \text{if} \quad net \geq T \\ 0 & \text{otherwise.} \end{cases}$$

- *T* is the threshold of the unit.
- The threshold *T* is set to the number of “one” components in the query vector ***x***, *T* := |***x***|
  - If the output of the unit is *1* we say that the unit fires
  - for the output *0* the unit does not fire
- The cost of the one-step retrieval is *O(n · m)*.

## Learning

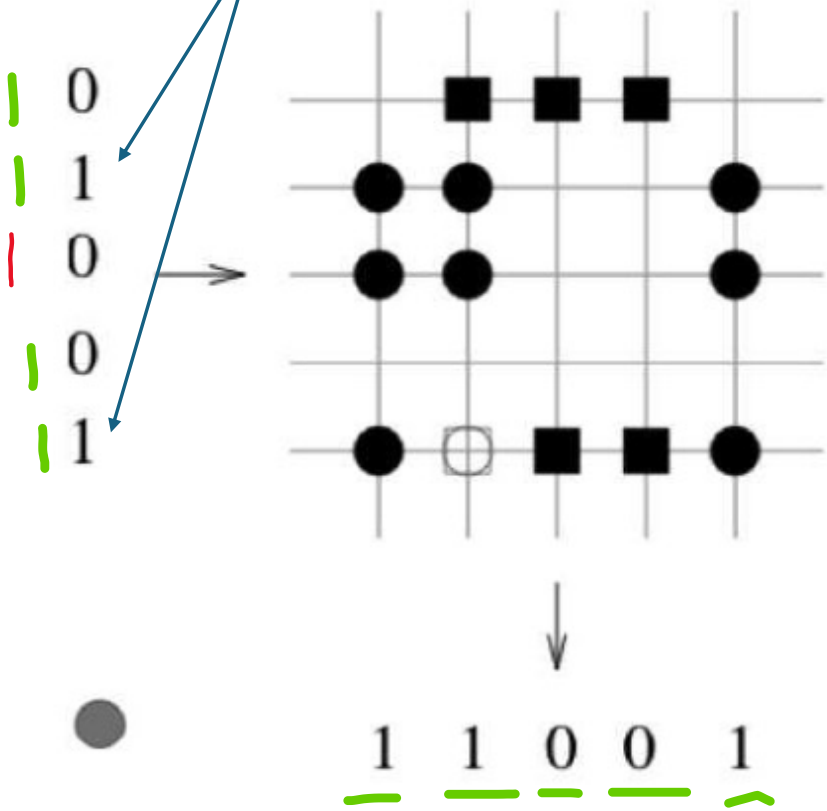



## Retrieval

$$net_i = \sum_{j=1}^{n} w_{ij} x_j,$$

$$y_i = \begin{cases} 1 & \text{if } net \geq T \\ 0 & \text{otherwise.} \end{cases}$$

*T = 2*



# Retrieval

The retrieval is called:

- **hetero-association,** if both vectors are different x≠ y,
- **association**, if x = y, the answer vector represents the reconstruction of the disturbed query vector.

For simplicity we assume that the dimension of the query vector and the answer vector are the same, $n = m$.

# Monte Carlo Lernmatrix

$$net_i = \sum_{j=1}^{n} w_{ij} x_j,$$

The suggested probabilistic retrieval rule for the determination of the answer vector $\mathbf{y}$ for the query vector $\mathbf{x}$ is

$$p(y_i = 1|\mathbf{x}) = \frac{1}{n} \cdot \left( \frac{net_i}{\sum_{v=1}^{n} net_v} \right)$$

and

$$p(y_i = 0|\mathbf{x}) = \frac{1}{n} \cdot \left( 1 - \frac{net_i}{\sum_{v=1}^{n} net_v} \right)$$

describing the probability of firing or not firing of one unit with

$$1 = \sum_{i=1}^{n} \left( p(y_i = 1|\mathbf{x}) + p(y_i = 0|\mathbf{x}) \right).$$

During the query operation one unit is randomly sampled and either it fires or not according to the probability distribution. To determine the answer vector we have to sample the Monte Carlo Lernmatrix several times. For the reconstructed vector three states will be present: 1 for fired units, 0 for not fired units and *unknown* for silent units.

# Quantum Counting of Ones

- In a binary string of the length *N* we can represent the fraction of *k* ones by the simple formula *k/N* and of the zeros as *(N − k)/N* resulting in a linear relation.
  - We can interpret these numbers as probability values
  - We can map these linear relations into the sigmoid-like probability functions for the presence of ones using Euler's formula in relation to trigonometry,

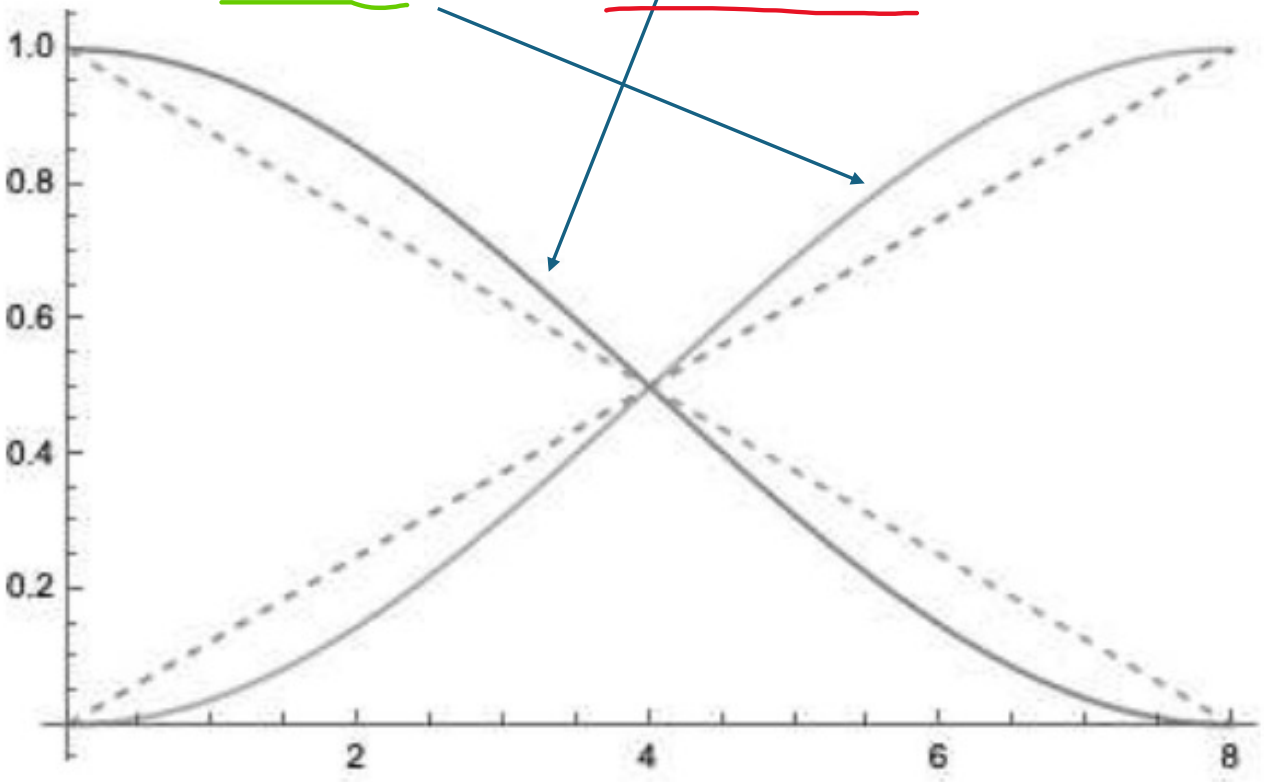
for ones

$$\left(\sin\left(\frac{\pi \cdot k}{2 \cdot N}\right)\right)^2 = \left|\frac{e^{i \cdot \frac{\pi \cdot k}{2 \cdot N}} - e^{-i \cdot \frac{\pi \cdot k}{2 \cdot N}}}{2}\right|^2 \in [0, 1]$$

and of zeros with

$$\left(\cos\left(\frac{\pi \cdot k}{2 \cdot N}\right)\right)^2 = \left|\frac{e^{i \cdot \frac{\pi \cdot k}{2 \cdot N}} + e^{-i \cdot \frac{\pi \cdot k}{2 \cdot N}}}{2}\right|^2 \in [0, 1]$$

$$\left(\sin\left(\frac{\pi \cdot k}{2 \cdot N}\right)\right)^2 + \left(\cos\left(\frac{\pi \cdot k}{2 \cdot N}\right)\right)^2 = 1$$




- Sigmoid-like probability functions for N = 8 is indicated by continuous line, the linear relation by the dashed lines.
- The x-axis indicates the k values, the y-axis the probabilities

- For count the number of ones we introduced the control qubit in superposition $1/\sqrt{2} \cdot (|0\rangle + |1\rangle)$
  - For the superposition part represented by the control qubit 0 the phase $e^{i \cdot \frac{\pi \cdot k}{2 \cdot N}}$ is applied for each one
  - For the superposition part represented by the control qubit 1 the phase $e^{-i \cdot \frac{\pi \cdot k}{2 \cdot N}}$ is applied for each one

  The string *101* has two ones, *k=2, N=3*

$$\frac{1}{\sqrt{2}} \cdot |0\rangle \otimes \left(e^{i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \otimes |0\rangle \otimes e^{i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle\right) + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes \left(e^{-i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \otimes |0\rangle \otimes e^{-i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle\right) =$$

$$\frac{e^{i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |0101\rangle + \frac{e^{-i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |1101\rangle$$

- If we apply a Hadamard gate to the control qubit

$$(H \otimes I \otimes I \otimes I) \cdot \left( \frac{e^{i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |0101\rangle + \frac{e^{-i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |1101\rangle \right) =$$

$$\frac{e^{i \cdot \frac{\pi \cdot 2}{2 \cdot 3}} + e^{-i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{2} |0101\rangle + \frac{e^{i \cdot \frac{\pi \cdot 2}{2 \cdot 3}} - e^{-i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{2} |1101\rangle =$$

$$\cos\left(\frac{\pi \cdot 2}{2 \cdot 3}\right) \cdot |0101\rangle + i \cdot \sin\left(\frac{\pi \cdot 2}{2 \cdot 3}\right) \cdot |1101\rangle =$$

$$\left(\cos\left(\frac{\pi \cdot 2}{2 \cdot 3}\right) \cdot |0\rangle + i \cdot \sin\left(\frac{\pi \cdot 2}{2 \cdot 3}\right) \cdot |1\rangle\right) \otimes |101\rangle$$

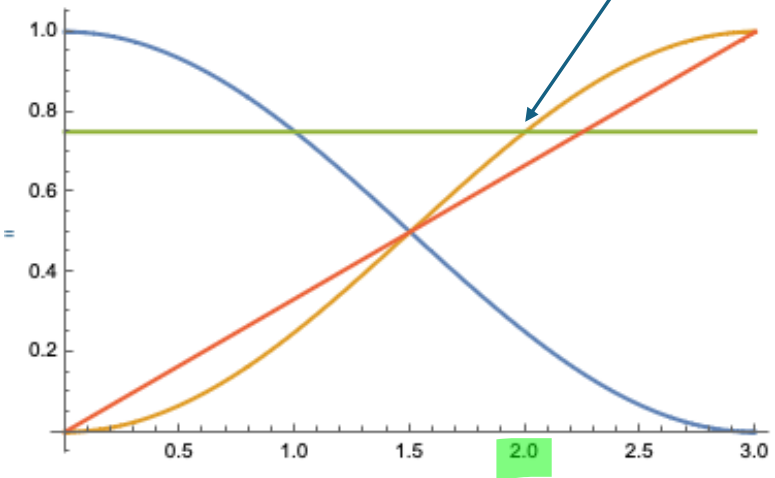
The probability of measuring the control qubit $|0\rangle$ is

$$p(|0\rangle) = p(|0101\rangle) = \left(\cos\left(\frac{\pi \cdot 2}{2 \cdot 3}\right)\right)^2 = 0.25$$

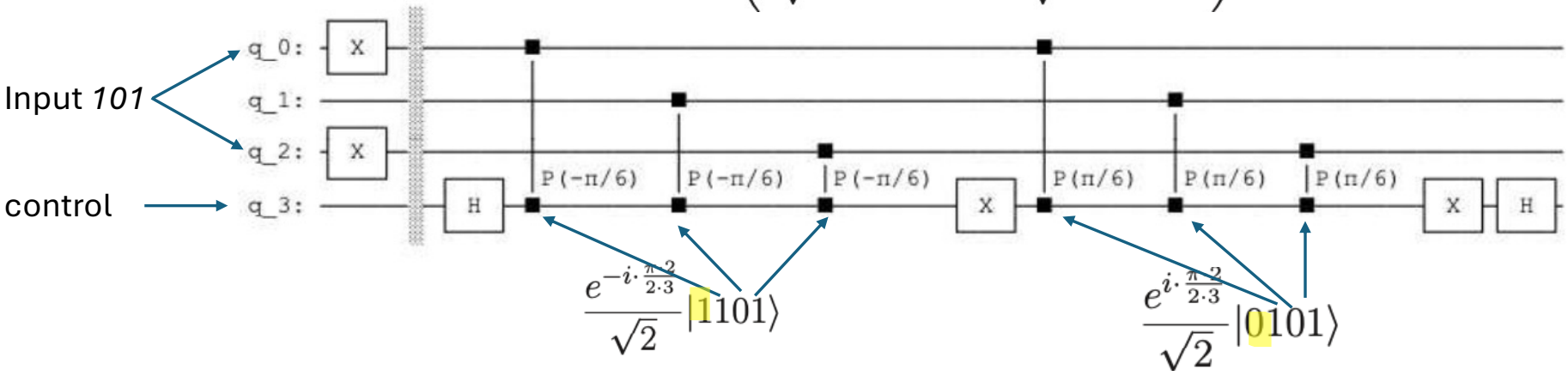
and the probability of measuring the control qubit $|1\rangle$ is

$$p(|1\rangle) = p(|1101\rangle) = \left(\sin\left(\frac{\pi \cdot 2}{2 \cdot 3}\right)\right)^2 = 0.75$$

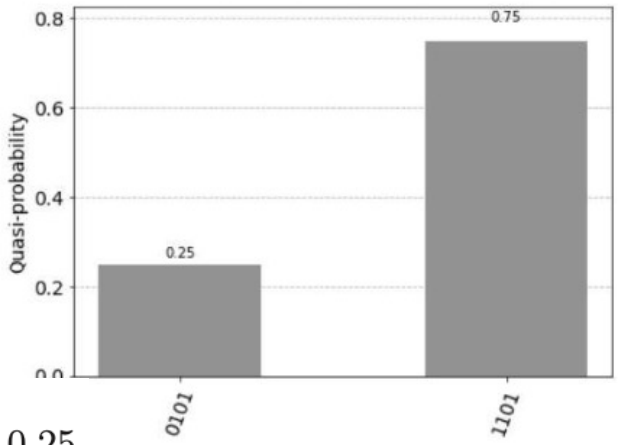
indicating the presence of two ones.



$$(H \otimes I \otimes I \otimes I) \cdot \left( \frac{e^{i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |0101\rangle + \frac{e^{-i \cdot \frac{\pi \cdot 2}{2 \cdot 3}}}{\sqrt{2}} |1101\rangle \right)$$




$$\frac{1}{\sqrt{2}} \cdot |0\rangle \otimes \left( e^{i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \otimes |0\rangle \otimes e^{i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \right) + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes \left( e^{-i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \otimes |0\rangle \otimes e^{-i \cdot \frac{\pi}{2 \cdot 3}} \cdot |1\rangle \right)$$

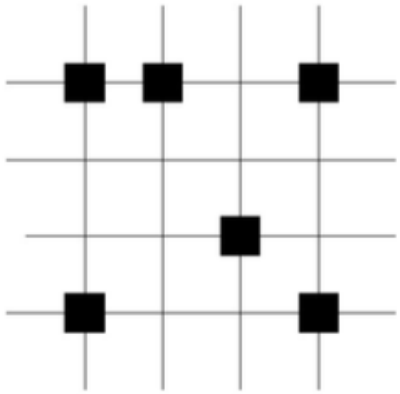



$$p(|0\rangle) = p(|0101\rangle) = \left( \cos\left( \frac{\pi \cdot 2}{2 \cdot 3} \right) \right)^2 = 0.25$$

$$p(|1\rangle) = p(|1101\rangle) = \left( \sin\left( \frac{\pi \cdot 2}{2 \cdot 3} \right) \right)^2 = 0.75$$

Probability of measuring

# Quantum Lernmatrix



- Weight matrix represented by four units after learning the correlation of the three patterns
  - $x_1 = (1,0,0,1)$; $y_1 = (1,0,0,1)$,
  - $x_2 = (1,0,0,0)$; $y_2 = (0,1,0,0)$
  - $x_3 = (0, 0, 1, 0)$; $y_3 = (0, 0, 1, 0)$
- The learning is identical with the learning phase of the Lernmatrix.

# Quantum Lernmatrix

$$|index_1\rangle = |11\rangle \quad |index_2\rangle = |10\rangle$$

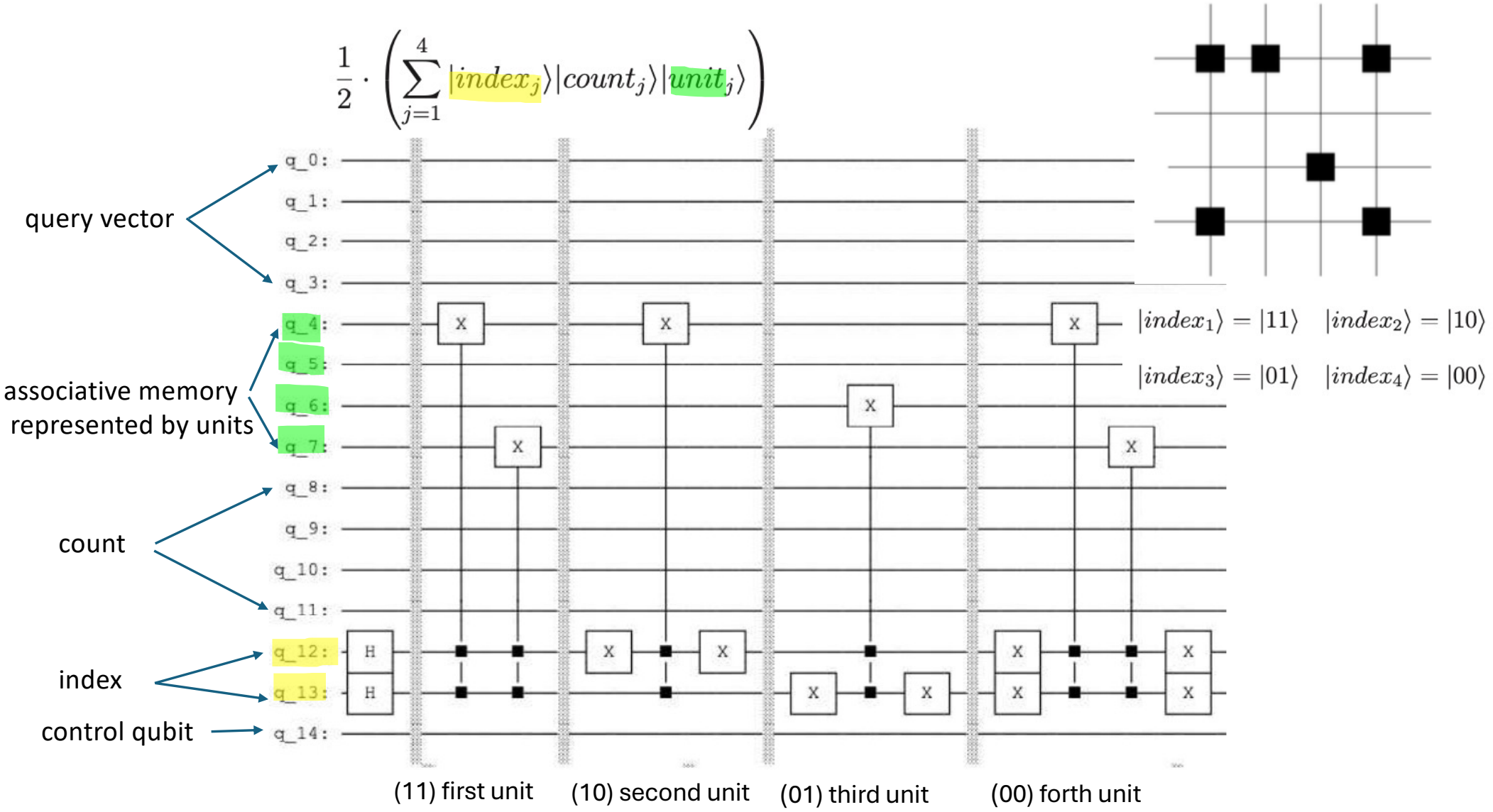
$$|index_3\rangle = |01\rangle \quad |index_4\rangle = |00\rangle$$

with the weight vectors the following state is present, the state $count_j$ and $unit$ represented by four qubits each for the four binary weights, with

$$|unit_j\rangle = |(w_4 w_3 w_1 w_1)_j\rangle$$

$$\frac{1}{2} \cdot \left( \sum_{j=1}^{4} |index_j\rangle |count_j\rangle |unit_j\rangle \right)$$

The value $|count_j\rangle$ is the unary representation of the Lernmatrix value $net_j$


$\frac{1}{2} \cdot \left( \sum_{j=1}^{4} |index_j\rangle |count_j\rangle |unit_j\rangle \right)$
query vector
associative memory represented by units
count
index
control qubit
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
q_13:
q_14:
H
X
$|index_1\rangle = |11\rangle$
$|index_2\rangle = |10\rangle$
$|index_3\rangle = |01\rangle$
$|index_4\rangle = |00\rangle$
(11) first unit
(10) second unit
(01) third unit
(00) forth unit

query vector is $\mathbf{x}_q = (1, 0, 0, 1)$,

$$\frac{1}{2} \cdot \left( \sum_{j=1}^{4} |index_j\rangle |count_j\rangle |unit_j\rangle \right) \otimes |query\rangle =$$

$$\frac{1}{2} \cdot \left( \sum_{j=1}^{4} |(i_2 i_1)_j\rangle |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle$$

*For simplicity, we will ignore the index qubits*

We perform quantum counting using the control bit that is set in superposition

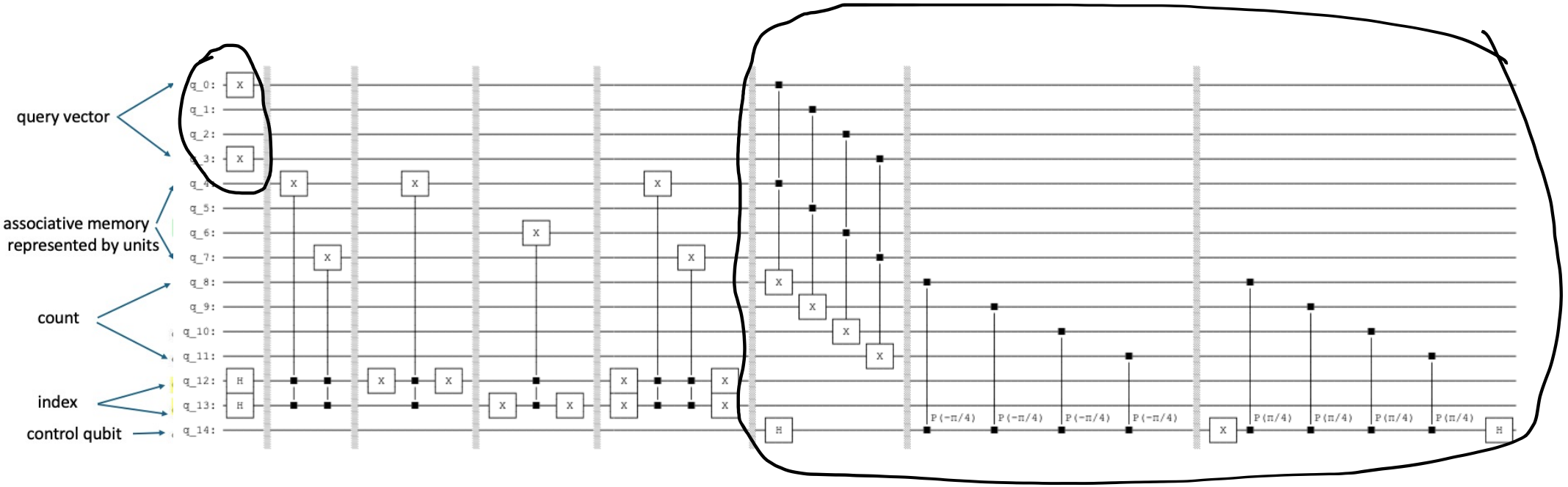
$$\frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) \otimes \frac{1}{2} \cdot \left( \sum_{j=1}^{4} |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle =$$

Applying controlled phase operation with $N = 2$ since two ones are present in the query vector and $count_j \leq 2$

$$\frac{1}{2 \cdot \sqrt{2}} \cdot |0\rangle \left( \sum_{j=1}^{4} e^{i \cdot \frac{\pi \cdot count_j}{2 \cdot 2}} \cdot |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle +$$

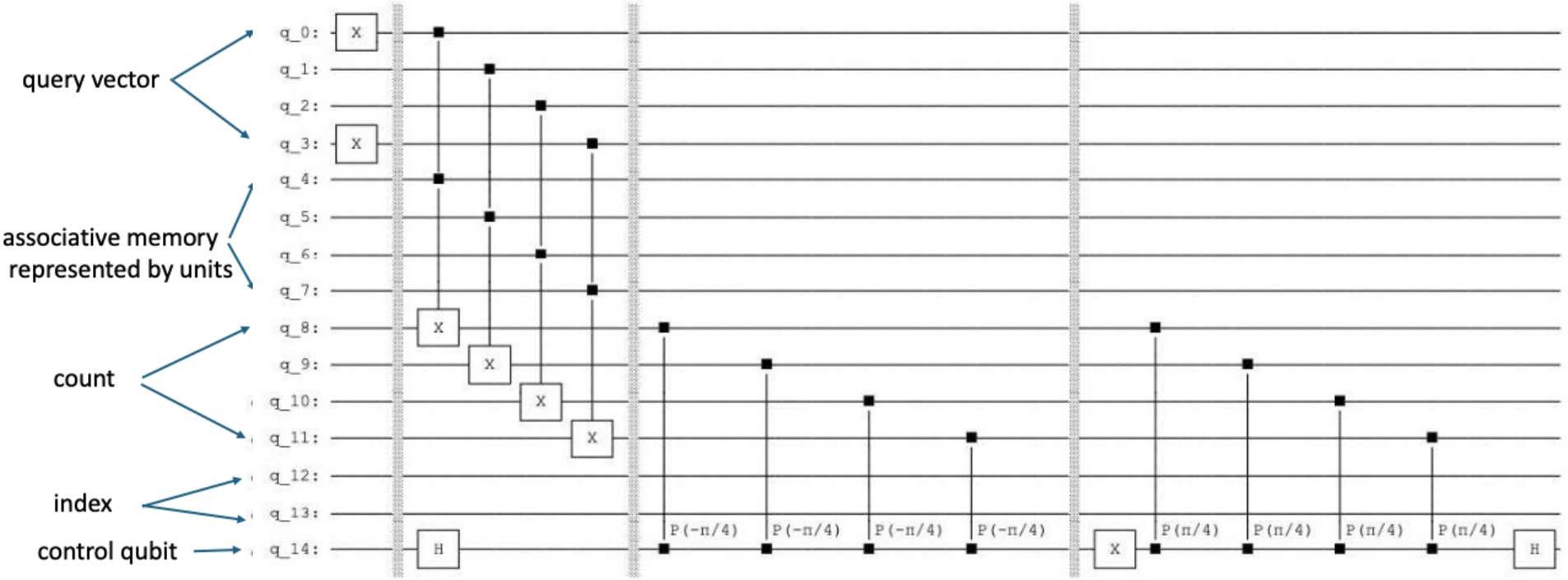
$$\frac{1}{2 \cdot \sqrt{2}} \cdot |1\rangle \left( \sum_{j=1}^{4} e^{-i \cdot \frac{\pi \cdot count_j}{2 \cdot 2}} \cdot |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle$$

# Quantum Lernmatrix




$$\frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) \otimes \frac{1}{2} \cdot \left( \sum_{j=1}^{4} |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle$$

$$\frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) \otimes \frac{1}{2} \cdot \left( \sum_{j=1}^{4} |(c_4 c_3 c_2 c_1)_j\rangle |(w_4 w_3 w_2 w_1)_j\rangle \right) \otimes |1001\rangle$$



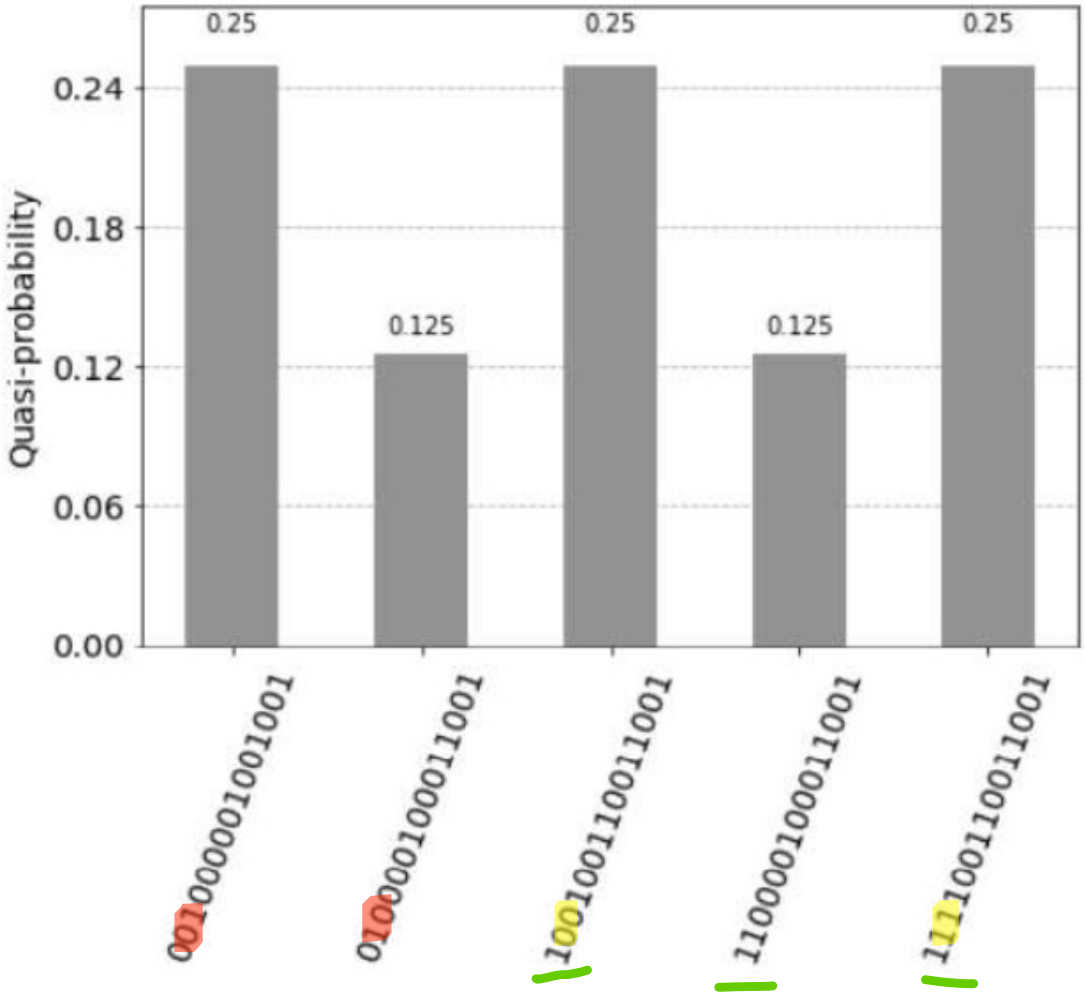
$$\frac{1}{2\cdot\sqrt{2}}\cdot|0\rangle\left(\sum_{j=1}^{4}e^{i\cdot\frac{\pi\cdot count_j}{2\cdot 2}}\cdot|(c_4c_3c_2c_1)_j\rangle|(w_4w_3w_2w_1)_j\rangle\right)\otimes|1001\rangle+$$

$$\frac{1}{2\cdot\sqrt{2}}\cdot|1\rangle\left(\sum_{j=1}^{4}e^{-i\cdot\frac{\pi\cdot count_j}{2\cdot 2}}\cdot|(c_4c_3c_2c_1)_j\rangle|(w_4w_3w_2w_1)_j\rangle\right)\otimes|1001\rangle$$

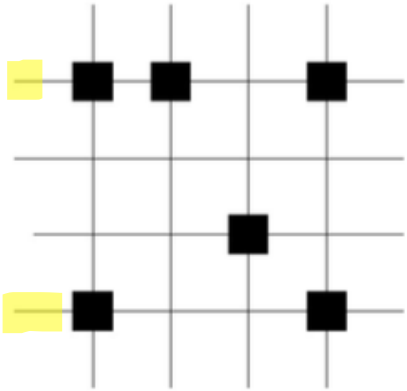
and applying the Hadamard gate to the control qubit we get

$$\left(\sum_{j=1}^{4}\frac{1}{2}\cdot\left(\cos\left(\frac{\pi\cdot count_j}{2\cdot 2}\right)\right)\cdot|0\rangle|(c_4c_3c_2c_1)_j\rangle|(w_4w_3w_2w_1)_j\rangle\right)\otimes|1001\rangle+$$

$$\left(\sum_{j=1}^{4}\frac{1}{2}\cdot\left(i\cdot\sin\left(\frac{\pi\cdot count_j}{2\cdot 2}\right)\right)\cdot|1\rangle|(c_4c_3c_2c_1)_j\rangle|(w_4w_3w_2w_1)_j\rangle\right)\otimes|1001\rangle$$




query vector $\mathbf{x}_q = (1, 0, 0, 1)$

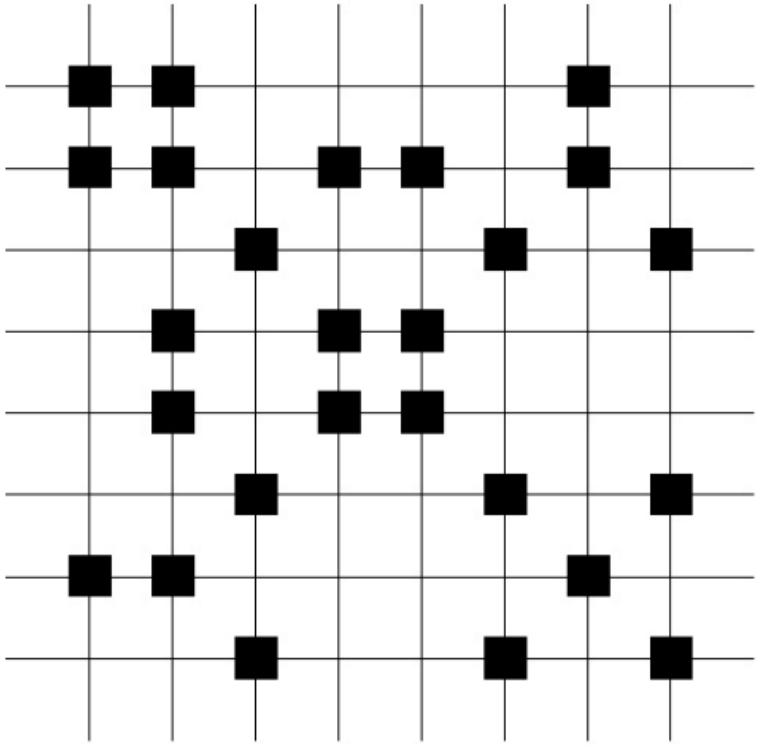



solution (1,0,0,1)

# Lernmatrix

In this example we store three patterns representing three associations:

- $\mathbf{x}_1 = (1,1,0,0,0,0,1,0)$; $\mathbf{y}_1 = (1,1,0,0,0,0,1,0)$,
- $\mathbf{x}_2 = (0,1,0,1,1,0,0,0)$; $\mathbf{y}_2 = (0,1,0,1,1,0,0,0)$,
- $\mathbf{x}_3 = (0,0,1,0,0,1,0,1)$; $\mathbf{y}_3 = (0,0,1,0,0,1,0,1)$



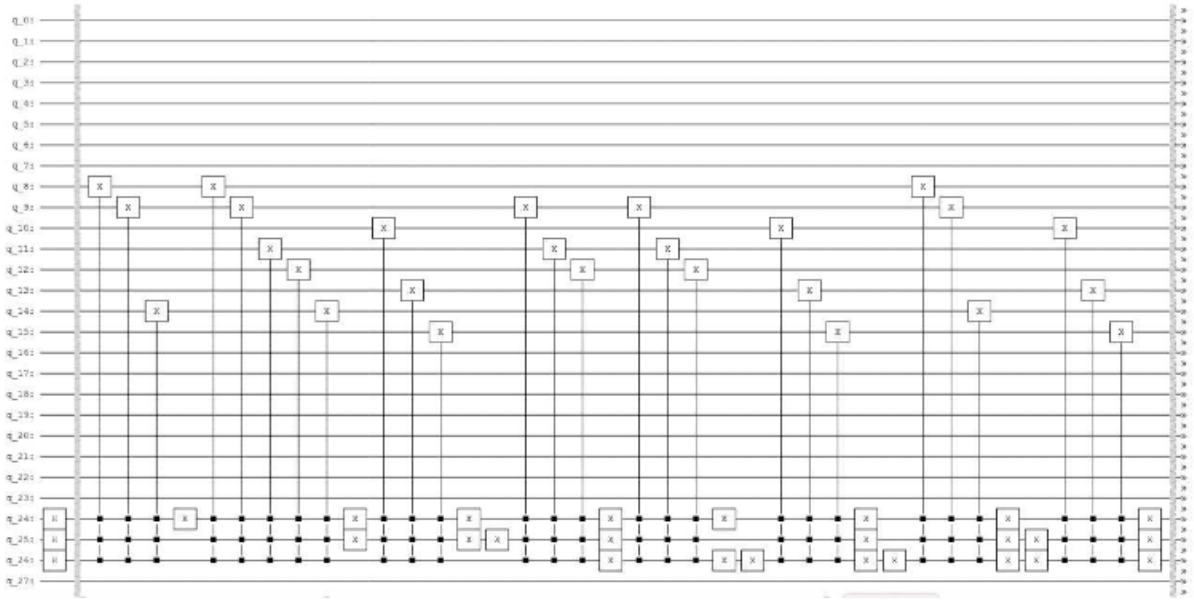
$$|index_1\rangle = |111\rangle \quad |index_2\rangle = |110\rangle$$

$$|index_3\rangle = |101\rangle \quad |index_4\rangle = |100\rangle$$

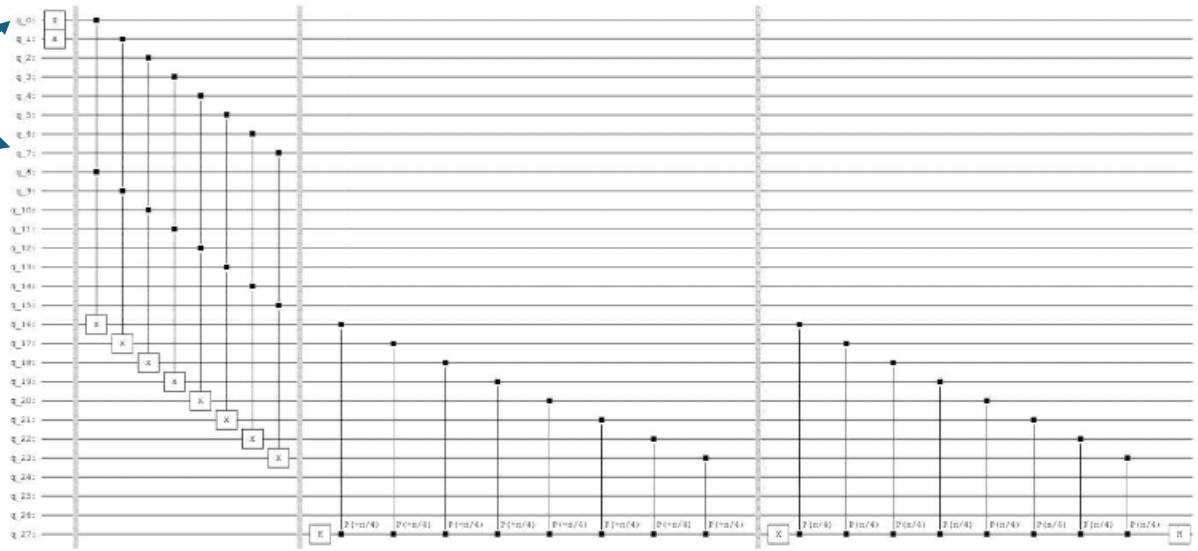
$$|index_5\rangle = |011\rangle \quad |index_6\rangle = |010\rangle$$

$$|index_7\rangle = |001\rangle \quad |index_8\rangle = |000\rangle$$

storing patterns



query

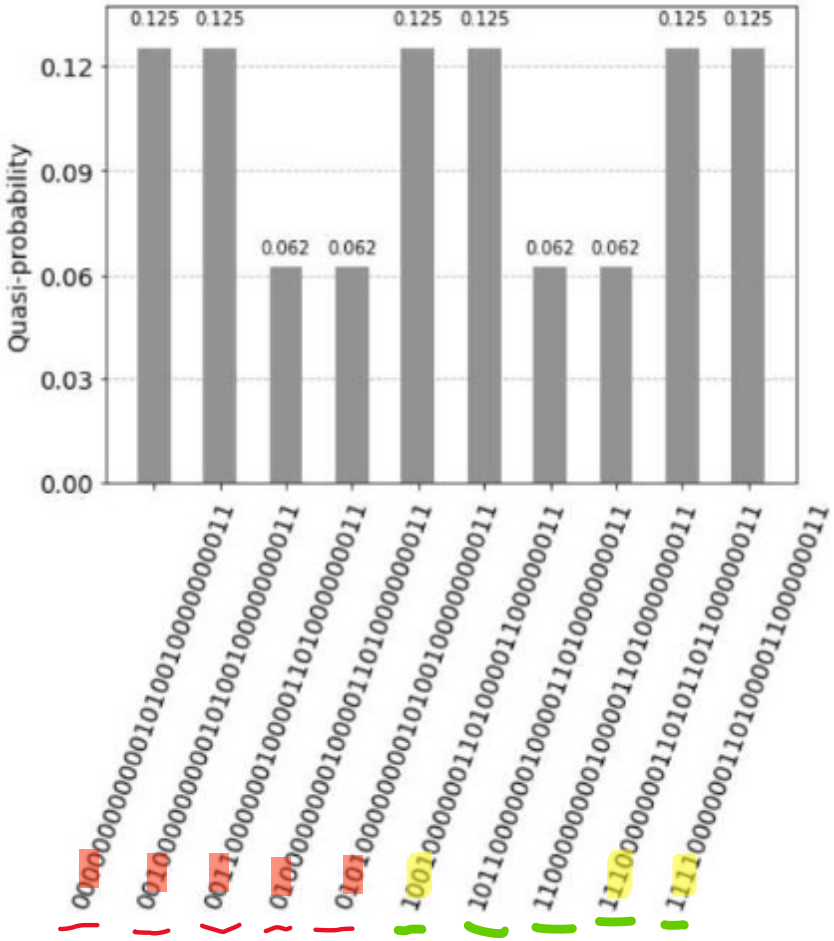


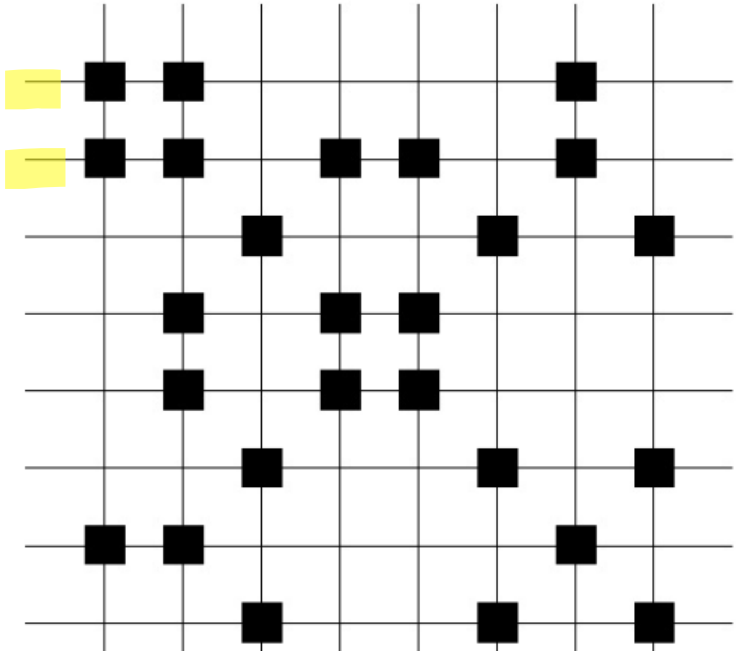
$$\frac{1}{\sqrt{8}} \cdot \left( \sum_{j=1}^{8} |index_j\rangle |count_j\rangle |unit_j\rangle \right) \otimes |00000011\rangle.$$

$$\left( \sum_{j=1}^{8} \frac{1}{\sqrt{8}} \cdot \left( \cos\left( \frac{\pi \cdot count_j}{2 \cdot N} \right) \right) \cdot |0\rangle |count_j\rangle |unit_j\rangle \right) \otimes |00000011\rangle +$$

$$\left( \sum_{j=1}^{8} \frac{1}{\sqrt{8}} \cdot \left( i \cdot \sin\left( \frac{\pi \cdot count_j}{2 \cdot N} \right) \right) \cdot |1\rangle |count_j\rangle |unit_j\rangle \right) \otimes |00000011\rangle$$

query vector $\mathbf{x}_q$ = (1,1,0,0,0,0,0,0)



111 110. 101. 100. 011 010. 001 000

solution(1, 1, 0, 0, 0, 0, 1, 0)

$$|index_1\rangle = |111\rangle \quad |index_2\rangle = |110\rangle$$
$$|index_3\rangle = |101\rangle \quad |index_4\rangle = |100\rangle$$
$$|index_5\rangle = |011\rangle \quad |index_6\rangle = |010\rangle$$
$$|index_7\rangle = |001\rangle \quad |index_8\rangle = |000\rangle$$

- For *n* classical neurons we need $log_2 n$ quantum neurons,
- For one query the cost are $log_2 n$
- The cost of determining the determine the answer vector are related to the number of ones in the query vector
- The number of ones is *k* and *k* <<*n*
- The costs are $\alpha \cdot k \cdot log_2 n$ and we require $\alpha \cdot k$ copies of quantum lernmatrices
- $\alpha$ is a constant that determines the frequency of the sampling process

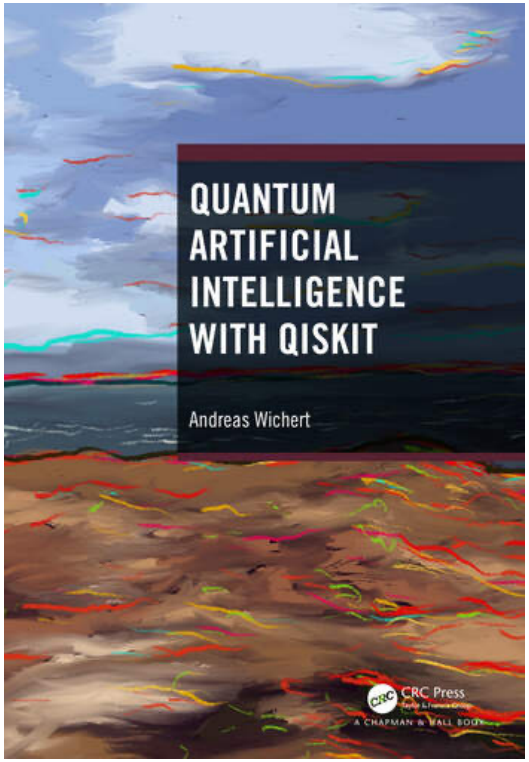
# Binary Patterns

- Applications:
  - Quantum Associative Memory
  - Quantum Clustering
  - Quantum kNN
- Advantages:
  - Capacity (exponential less)
  - Speed up (quadratic faster)

- But: *only if we ignore the preparation*....

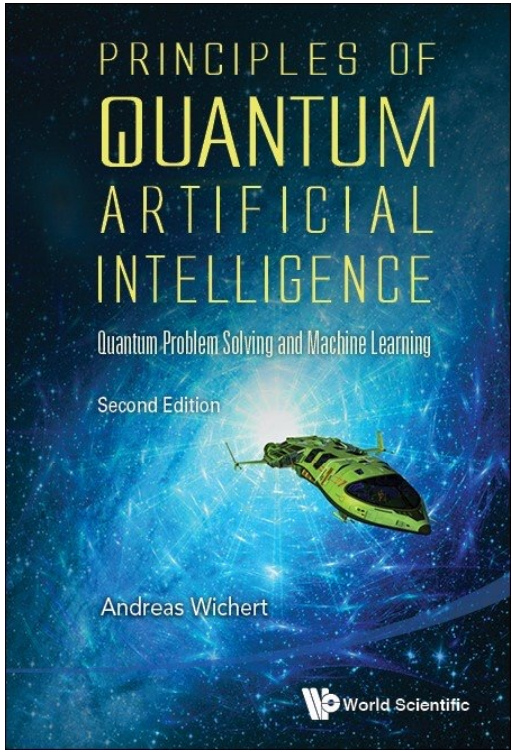
# Open Problems

- The naive assumption that the speed-up is quadratic is not realistic.
- The input (reading) problem:
  - Although the existing quantum algorithm requires only $O(\sqrt{N})$ steps and is faster than the classical algorithms, N data points must be read.
- The destruction problem:
  - A quantum associative memory can be queried only once because of the collapse during measurement (destruction)
- The input destruction problem **has not yet been solved**
- Theoretical speed-ups are usually analyzed by **ignoring** the input problem, which is the main bottleneck for data encoding




- Chapter 13
- Chapter 14
- Chapter 15

Quantum Artificial Intelligence with Qiskit, A. Wichert, Chapman and Hall/CRC, 2024




- Chapter 4
- (Chapter 12)

Principles of Quantum Artificial Intelligence: Quantum Problem Solving and Machine Learning, 2nd Edition, A. Wichert, World Scientific, 2020