Lecture 2: Quantum Circuits

Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

Overview

- Qiskit
- Quantum Circuits
- Deutsch Algorithm
- Quantum Gates
- Universality

Overview

- Qiskit is an open-source software development kit (SDK) for working with quantum computers at the level of circuits and algorithms
  - https://quantum-computing.ibm.com/
  - https://www.ibm.com/quantum/qiskit

- Qiskit provides different backend simulator functions by Aer package
- We will use use two simulators

- Qiskit provides different backend simulator functions by Aer package
- We will use use two simulators
  - The statevector simulator performs an ideal execution of qiskit circuits and returns the final state vector off the simulator after application (all qubits). The state vector of the circuit can represent the probability values (quasi probability in newer version) that correspond to the multiplication of the state vector by the unitary matrix that represents the circuit. The statevector simulator will take longer than other simulation methods and requires more computer mem- ory, since the state vector dimension grows exponentially with the number of qubits with n = 2m (with m number of qubits).

statevector simulator

- The *statevector* simulator performs an ideal execution of qiskit circuits and returns the final state vector off the simulator after application (all qubits)
- The state vector of the circuit can represent the probability values (quasi prob-ability in newer version) that correspond to the multiplication of the state vector by the unitary matrix that represents the circuit
- The statevector simulator will take longer than other simulation methods and requires more computer memory, since the state vector dimension grows exponentially with the number of qubits with $n = 2^m$ (with m number of qubits).

qasm simulator

- The *qasm* simulator promises to behave like an actual device of today, which is prone to noise resulting from decoherence
- It returns count, which is the sampling of the measured qubits that have to be defined in the circuit
- It is much smaller in size and will not increase in size exponentially as the number of qubits increases.

Quantum Coin

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$$

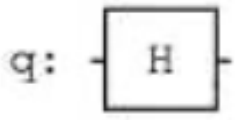
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle.$$

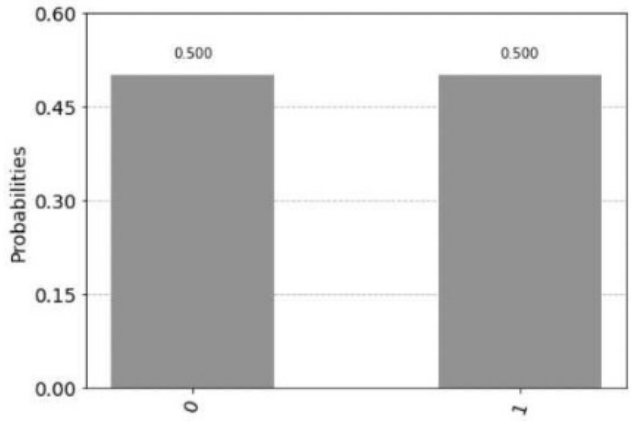
$$Statevector = \left[\frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{2}}\right]$$

qc = QuantumCircuit(1)

qc.h(0)

qc.draw()





Quantum Coin

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$$

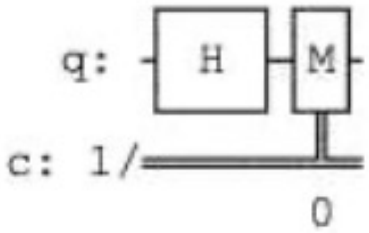
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle.$$

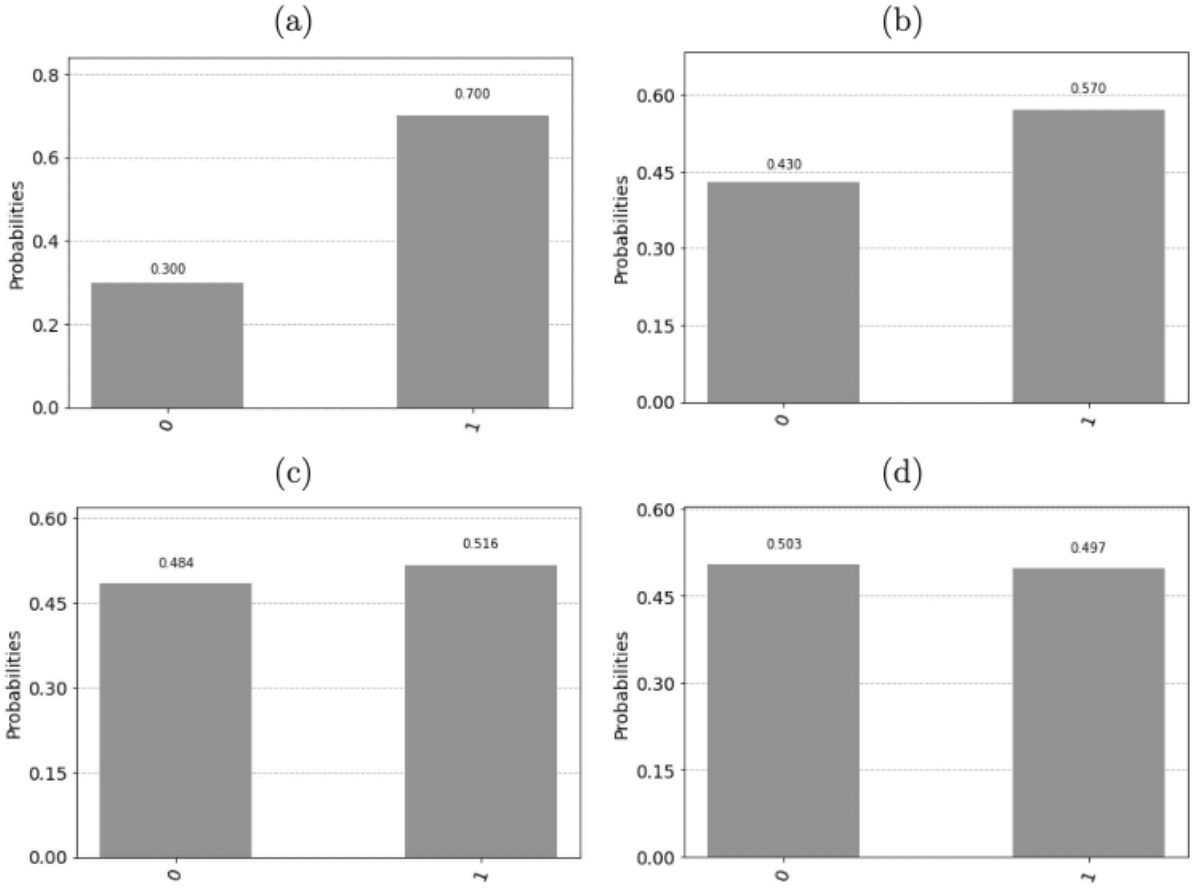
qc = QuantumCircuit(1,1)

qc.h(0)

qc.measure(0,0)

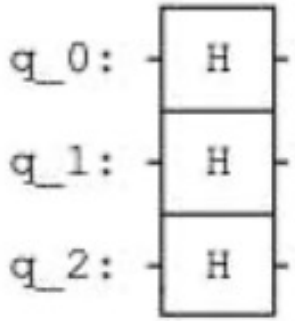
qc.draw()

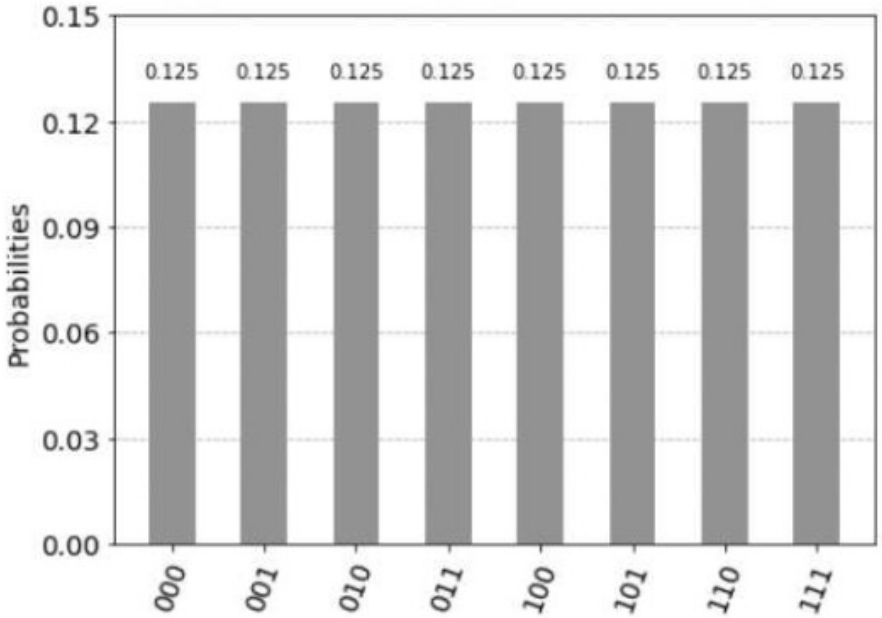




10 shots

$$\frac{1}{\sqrt{8}} \cdot (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$$





$$Circuit = \begin{pmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \end{pmatrix}$$

$$Statevector = \left[\frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}} \; \frac{1}{\sqrt{8}}\right]$$

Quantum Circuits

- The four bits conjunction $x \wedge y \wedge z \wedge v$ requires three quantum Toffoli gates *ccX* and three additional qubits that are zero

  - Qiskit uses the **little endian** notation, it stores the least-significant byte at the smallest address.
  - Qubits are represented from the most significant bit (MSB) on the left to the least significant bit (LSB) on the right (little-endian).
  - This is similar as binary numbers or to bitstring representation on classical computers
  - In our example we represent we represent *x* by the first qubit, *y* by the second qubit and so on

$$|x\rangle|y\rangle|0\rangle|z\rangle|0\rangle|v\rangle|0\rangle.$$

First quantum Toffoli gate

$$(CCX \cdot |x\rangle|y\rangle|0\rangle) \otimes (I_4| \cdot |z\rangle|0\rangle|v\rangle|0\rangle) = |x\rangle|y\rangle|x \wedge y\rangle|z\rangle|0\rangle|v\rangle|0\rangle$$

Second quantum Toffoli gate

$$(I_2 \cdot |x\rangle|y\rangle) \otimes (CCX \cdot |x \wedge y\rangle|z\rangle|0\rangle \otimes (I_2 \cdot |v\rangle|0\rangle) =$$

$$= |x\rangle|y\rangle|x \wedge y\rangle|z\rangle|x \wedge y \wedge z\rangle|v\rangle|0\rangle.$$

Third quantum Toffoli gate

$$(I_4 \cdot |x\rangle|y\rangle|x \wedge y\rangle|z\rangle) \otimes (CCX \cdot |x \wedge y \wedge z\rangle|v\rangle|0\rangle) =$$

$$= |x\rangle|y\rangle|x \wedge y\rangle|z\rangle|x \wedge y \wedge z\rangle|v\rangle|x \wedge y \wedge z \wedge v\rangle.$$

The circuit corresponds to the following unitary mapping

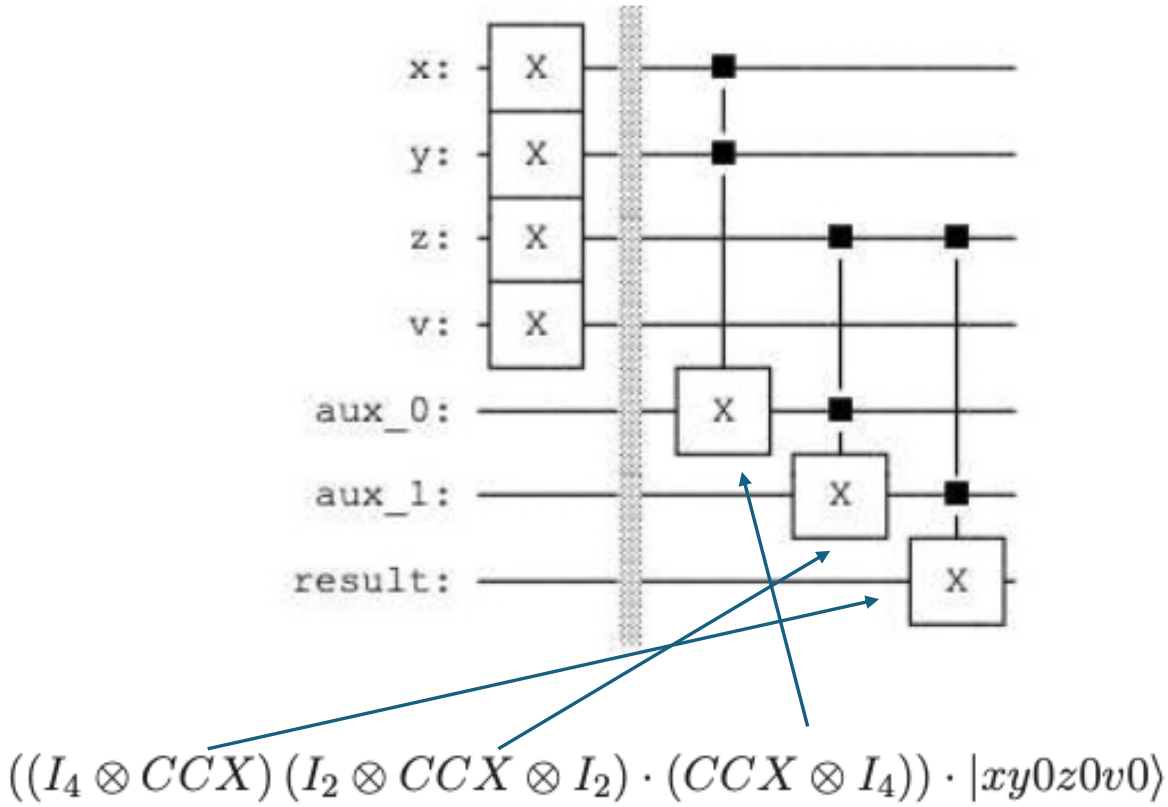
$$((I_4 \otimes CCX)\,(I_2 \otimes CCX \otimes I_2) \cdot (CCX \otimes I_4)) \cdot |xy0z0v0\rangle$$

with the result

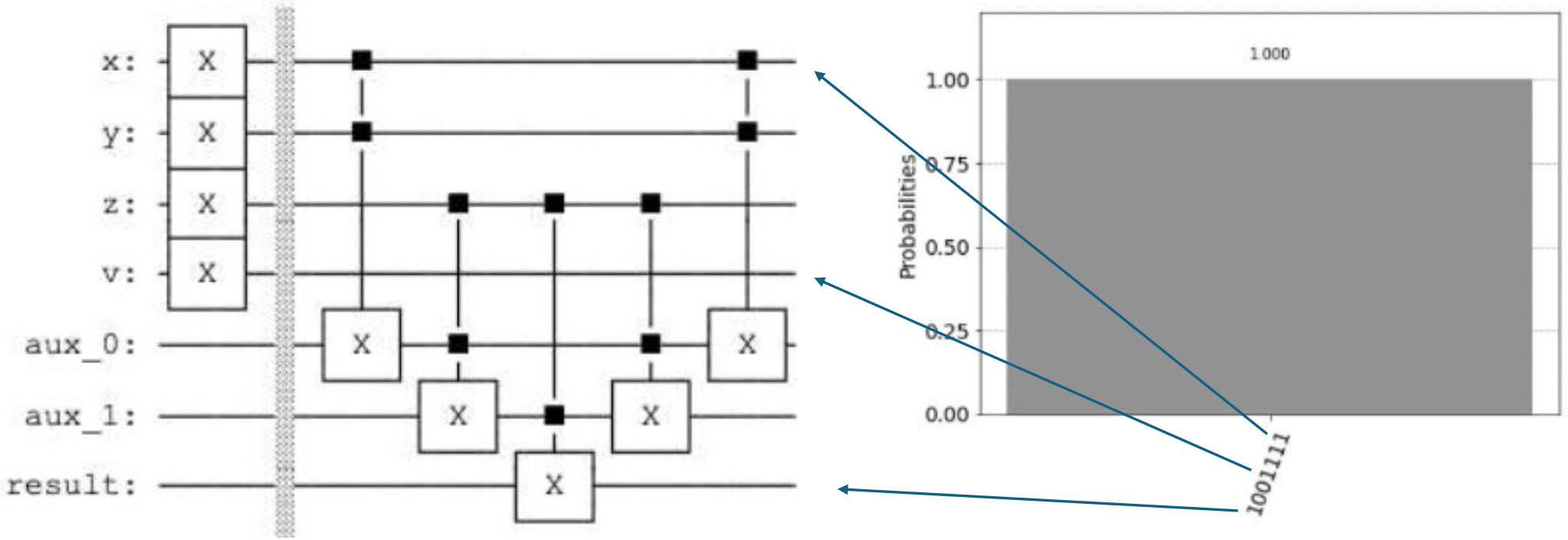
$$|x\rangle|y\rangle|x \wedge y\rangle|z\rangle|x \wedge y \wedge z\rangle|v\rangle|x \wedge y \wedge z \wedge v\rangle.$$

We initialize the qubits *x, y, z v* to one

```
x = QuantumRegister(1, 'x')
y = QuantumRegister(1, 'y')
z = QuantumRegister(1, 'z')
v = QuantumRegister(1, 'v')
aux =  QuantumRegister(2,'aux')
r = QuantumRegister(1, 'result')
qc = QuantumCircuit(x,y,z,v,aux,r)
qc.x(x)
qc.x(y)
qc.x(z)
qc.x(v)
qc.barrier()
qc.ccx(x,y,aux[0])
qc.ccx(aux[0],z,aux[1])
qc.ccx(aux[1],z,r)
```



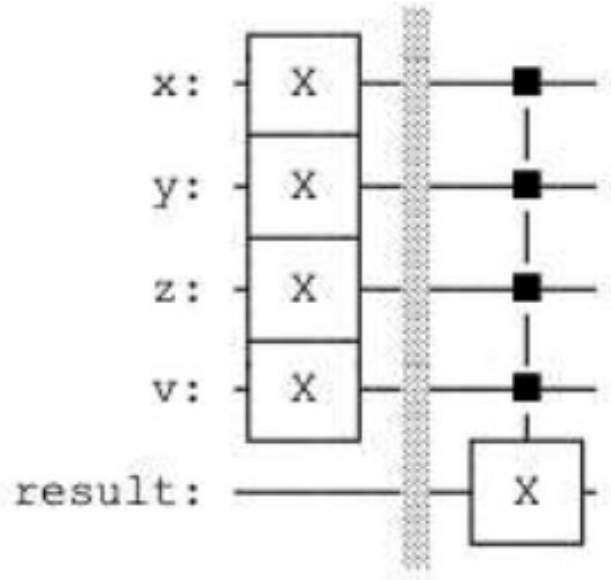
- The qubits aux0=1, aux1 = 1 are usually not required for further computation be- cause the result is represented in the output qubit result.
- However, they are entangled with the output qubit.
- **It is not possible to reset them to zero.**
- Instead, they are un- computed.
- Because for the matrix $CCX^{-1} = CCX$, we recompute the first and the second quantum Toffoli gate after determining the result.


x:
y:
z:
v:
aux_0:
aux_1:
result:
X
1.000
1.00
0.75
0.50
0.25
0.00
Probabilities
1001111

General multi-controlled X gate

- Instead one can used as well the command *MCXGate(4)* without the need of defining auxiliary registers
- The *MCXGate(n)* gate is a general, multi-controlled X gate

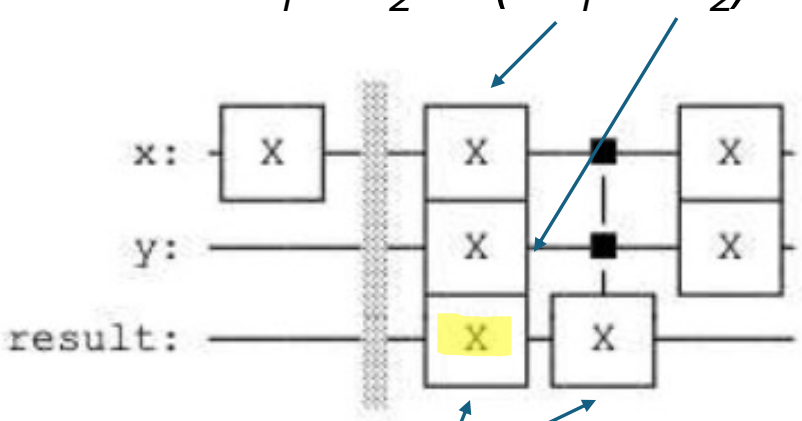
```
gate = MCXGate(4)
qc.append(gate,[x,y,z,v,r])
qc.draw()
```



OR operation

- The OR operation is represented by the unitary mapping according to the De Morgan's laws $x_1 \vee x_2 = \neg(\neg x_1 \wedge \neg x_2)$

Toffoli Gate:



It computes the $NAND$ operation, the ancilla (fixed) bit x_3 is set to 1.

$$ccX(x_1, x_2, 1) = (x_1, x_2, \neg(x_1 \wedge x_2))$$

- **David Elieser Deutsch** was born into a Jewish family in Haifa, Israel on 18 May 1953
  - He is a British physicist at the University of Oxford.

- "David Elieser Deutsch laid the foundations of the quantum theory of computation, and has subsequently made or participated in many of the most important advances in the field, **including the discovery of the first quantum algorithm"**

- A number of physics journals rejected some of Deutsch's early quantum-computing work, saying it was "**too philosophical**."

https://www.daviddeutsch.org.uk

Deutsch Algorithm

- The Deutsch algorithm exploits the superposition of qubits generated by Hadamard gates and is more powerful than **any** classical algorithm
- It determines if an unknown function $f: \boldsymbol{B}^1 \rightarrow \boldsymbol{B}^1 : f(x)=y$ of one bit is constant or not by calling the function $f(x)$ **one time**.

- A constant function on one bit is either $f(x) = 1$ or $f(x) = 0$.
- A non constant function is either the identity function
  - $f(0) = 0$ and $f(1) = 1$
- or the flip function
  - $f(0) = 1$ and $f(1) = 0$.

- The condition of the function being constant $f(0) = f(1)$ implies that the XOR operation $\oplus$ is $f(0) \oplus f(1) = 0$ is zero
- The condition of the function is not constant $f(0) \neq f(1)$ implies that the XOR operation $\oplus$ is $f(0) \oplus f(1) = 1$ is one

- We can define a unitary operator U_f that acts on the two qubits

$$U_f \cdot |xy\rangle = |x\rangle|f(x) \oplus y\rangle.$$

- U_f can be implemented by a quantum Boolean circuit including CNOT gate

f(x) constant

- There are four different cases, for f*(x)* = *0* with the identity mapping

$$i)\quad U_f|00\rangle = |0\rangle|0 \oplus 0\rangle = |00\rangle,\quad U_f|01\rangle = |01\rangle,$$

$$U_f|10\rangle = |10\rangle,\quad U_f|11\rangle = |11\rangle$$

- for f(x) = 1 with the permutation of all elements

$$ii)\quad U_f|00\rangle = |0\rangle|1 \oplus 0\rangle = |01\rangle,\quad U_f|01\rangle = |00\rangle,$$

$$U_f|10\rangle = |11\rangle,\quad U_f|11\rangle = |10\rangle$$

f(x) not constant

- A non constant function, *f(x)* = *x* corresponds to a permutation of two elements

$$iii)\quad U_f|00\rangle = |0\rangle|0 \oplus 0\rangle = |00\rangle,\quad U_f|01\rangle = |01\rangle,$$
$$U_f|10\rangle = |11\rangle,\quad U_f|11\rangle = |10\rangle$$

- *f(x)* = ¬*x* with a permutation of two elements as well

$$vi)\quad U_f|00\rangle = |0\rangle|1 \oplus 0\rangle = |01\rangle,\quad U_f|01\rangle = |00\rangle,$$
$$U_f|10\rangle = |10\rangle,\quad U_f|11\rangle = |11\rangle.$$

- *f(x)* constant:
  - No permutation i) or permutation of all elements *ii)* indicates that the function is constant.

- *f(x)* non constant:
  - Permutation of two elements *iii), iv)* indicates that the function is non constant.

- The algorithm to determine if f(x) is constant or not is composed of four steps
  - In the first step of the algorithm, we build a superposition of two qubits

$$H_2 \cdot |01\rangle = H_1 \cdot |0\rangle \otimes H_1 \cdot |1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) =$$

$$H2 \cdot |01\rangle = \frac{1}{2} \cdot (|00\rangle - |01\rangle + |10\rangle - |11\rangle)\,.$$

$$Statevector = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

- In the second step we apply the U_f , gate.

$$U_f \cdot H_2 \cdot |01\rangle = U_f \left(\frac{1}{2} \cdot (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \right) =$$

$$= \frac{1}{2} \cdot (U_f \cdot |00\rangle - U_f \cdot |01\rangle + U_f \cdot |10\rangle - U_f \cdot |11\rangle) .$$

- In the second step we apply the U_f, gate.

$$U_f \cdot H_2 \cdot |01\rangle = U_f \left(\frac{1}{2} \cdot (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \right) =$$

$$= \frac{1}{2} \cdot (U_f \cdot |00\rangle - U_f \cdot |01\rangle + U_f \cdot |10\rangle - U_f \cdot |11\rangle) .$$

- There are four possible outcomes

- For constant function

$$i) \quad = \frac{1}{2} \cdot (|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

$$ii) \quad = \frac{1}{2} \cdot (|01\rangle - |00\rangle + |11\rangle - |10\rangle) = \frac{1}{2} \cdot (-|00\rangle + |01\rangle - |10\rangle + |11\rangle)$$

$$= \left(\frac{-|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = -\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

- For non constant function

$$iii) \quad = \frac{1}{2} \cdot (|00\rangle - |01\rangle + |11\rangle - |10\rangle) = \frac{1}{2} \cdot (|00\rangle - |01\rangle|10\rangle + |11\rangle)$$

$$= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

$$iv) \quad = \frac{1}{2} \cdot (|01\rangle - |00\rangle + |10\rangle - |11\rangle) = \frac{1}{2} \cdot (-|00\rangle + |01\rangle + |10\rangle - |11\rangle)$$

$$\left(\frac{-|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = -\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

- In the third step a Hadamard gate is applied to the first qubit

$$(H_1 \otimes I_1) \cdot U_f \cdot H_2 \cdot |01\rangle.$$

- There are four possible outcomes:
  - The fourth step the first qubit (that is in the basis state) is measured.
  - It is $|0\rangle$ if the function is constant,
  - otherwise $|1\rangle$

$$i) \quad |0\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

$$ii) \quad -|0\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

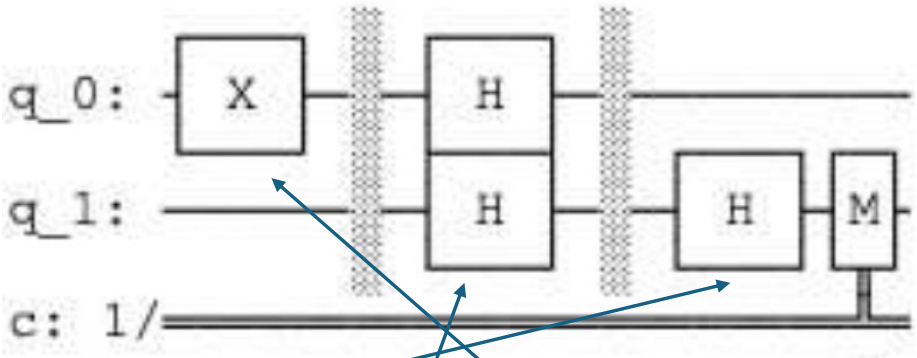
$$iii) \quad |1\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

$$vi) \quad -|1\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

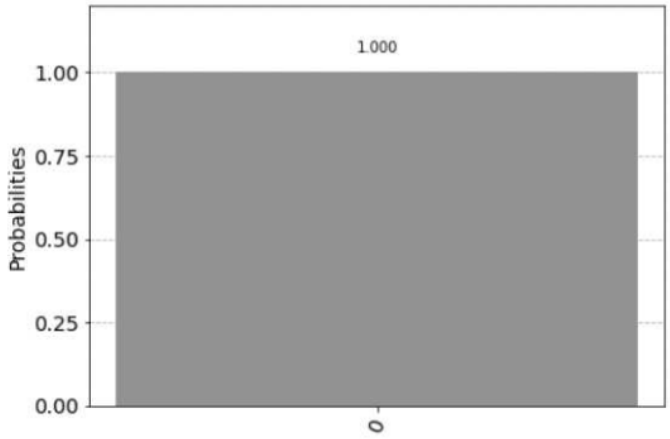
- In our simulation for simplicity, we assume the unknown function is the the identity mapping *f(x) = 0*

$$i)\ \ U_f|00\rangle = |0\rangle|0 \oplus 0\rangle = |00\rangle, \quad U_f|01\rangle = |01\rangle,$$

$$U_f|10\rangle = |10\rangle, \quad U_f|11\rangle = |11\rangle$$

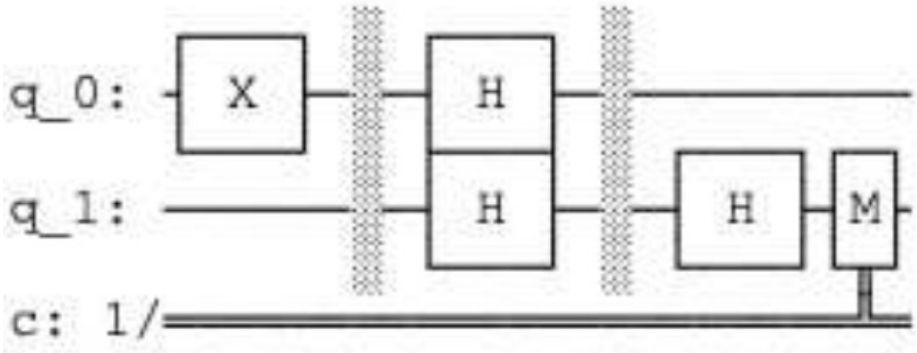


$$(H_1 \otimes I_1) \cdot U_f \cdot H_2 \cdot |01\rangle.$$



|0⟩, the function is constant

```
qc = QuantumCircuit(2,1)
qc.x(0)
qc.barrier()
qc.h(0)
qc.h(1)
#constant function f(x)=0, do nothing
qc.barrier()
qc.h(1)
#Measure the qubit 1
qc.measure(1,0)
qc.draw()
```

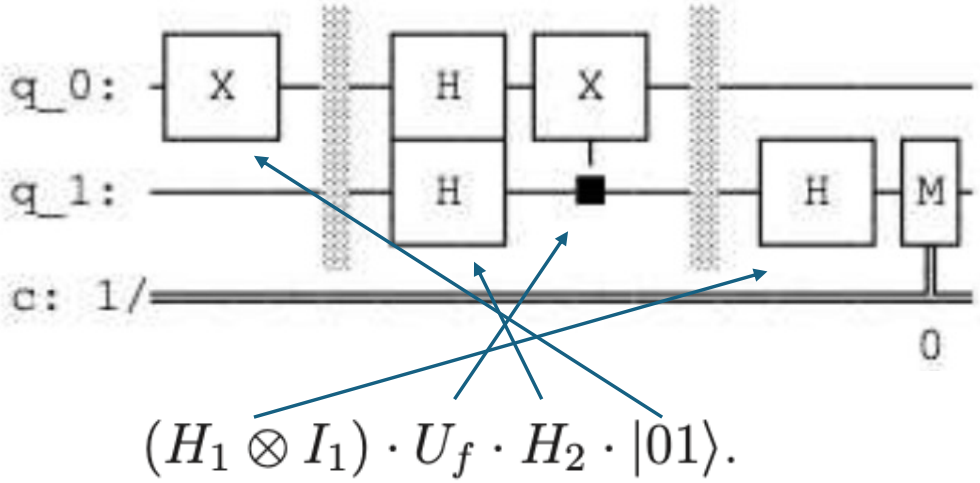


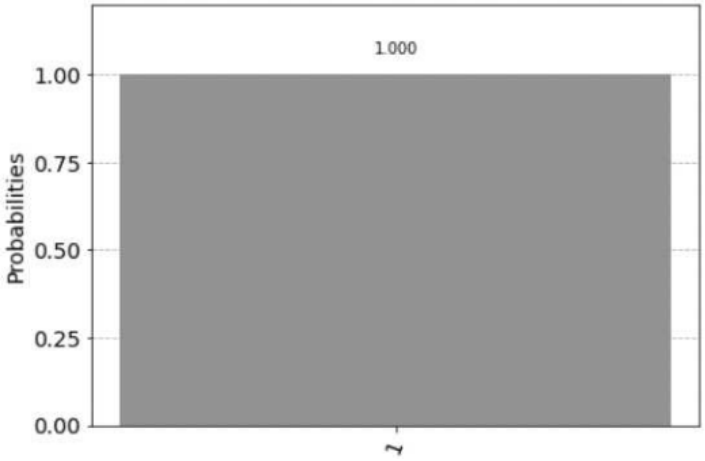
- For the non constant function, *f(x)* = *x* corresponds to a permutation of two elements

$$iii)\ \ U_f|00\rangle = |0\rangle|0 \oplus 0\rangle = |00\rangle, \quad U_f|01\rangle = |01\rangle,$$

$$U_f|10\rangle = |11\rangle, \quad U_f|11\rangle = |10\rangle$$

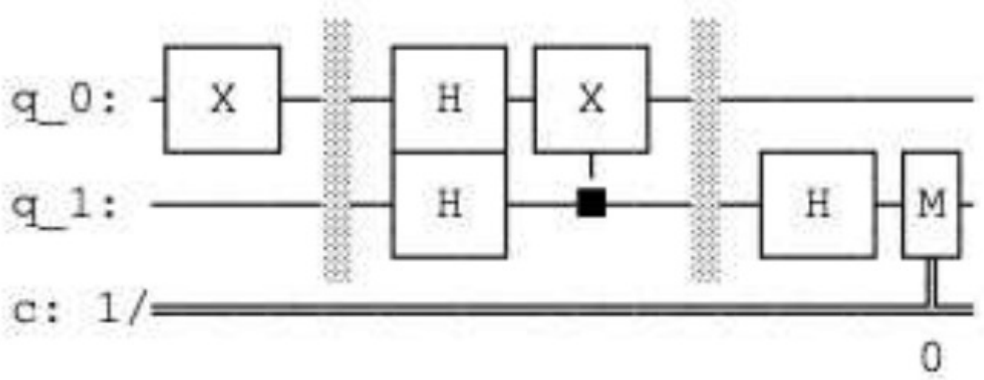
  - This operation can be achieved by the Controlled NOT gate. The CNOT gate flips the second qubit (the target qubit) if and only if the first qubit (the control qubit) is one. In our case the control qubit is *q1*



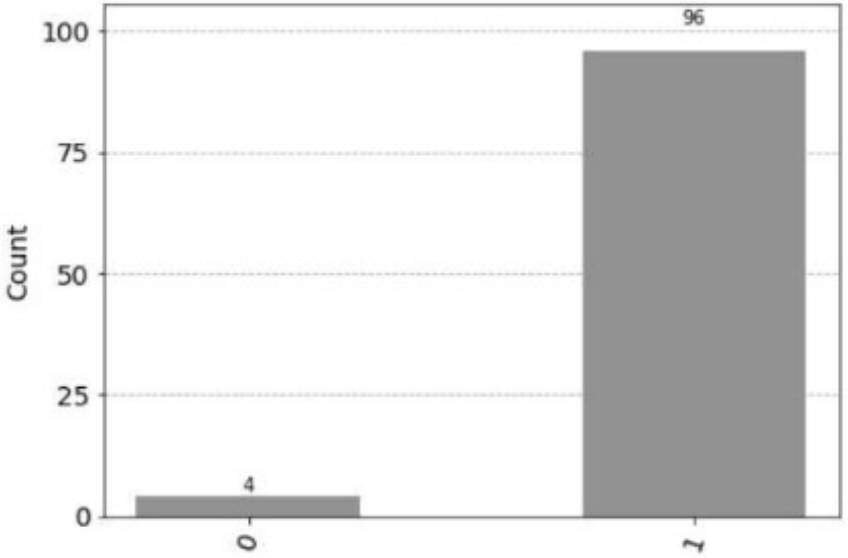


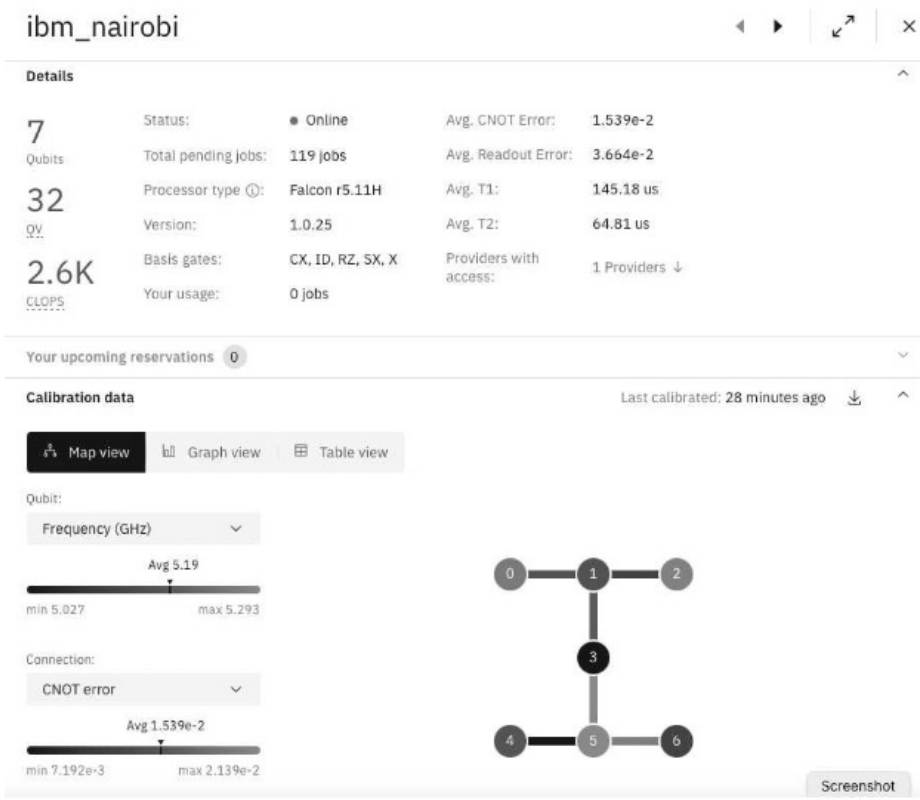
|1⟩, the function is non constant

```
qc = QuantumCircuit(2,1)
qc.x(0)
qc.barrier()
qc.h(0)
qc.h(1)
#identity function f(x)=x
qc.cx(1,0)
qc.barrier()
qc.h(1)
qc.measure(1,0)
qc.draw()
```



Deutsch Algorithm on a real Quantum Computer

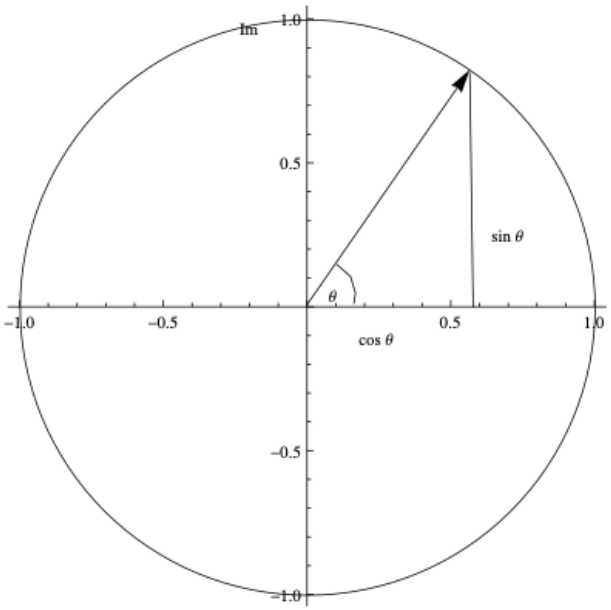




In the fourth step the first qubit (that is in the basis state) is measured. It is $|1\rangle$ with "probability" value 0.96 after 100 shots, the function is non constant with high probability. Total count are: $\{'0' : 4, '1' : 96\}$. The measurements of $|0\rangle$ result from noise through decoherence.

Quantum Gates

- Boolean quantum Gates
- Gates for one Qubit
  - Can change the phase of a qubit, $e^{i\theta}$ x
- Parameterized Rotation Gates
  - Parameterized gates play an important role in quantum machine learning
- Controlled-U Gates



$e^{i \cdot \theta} = \cos \theta + i \cdot \sin \theta$

Boolean quantum gates

- Toffoli Gate - ccX

$$
\begin{aligned}
CCX|000\rangle &\longrightarrow |000\rangle & CCX|100\rangle &\longrightarrow |100\rangle \\
CCX|001\rangle &\longrightarrow |001\rangle & CCX|101\rangle &\longrightarrow |101\rangle \\
CCX|010\rangle &\longrightarrow |010\rangle & CCX|110\rangle &\longrightarrow |111\rangle \\
CCX|011\rangle &\longrightarrow |011\rangle & CCX|111\rangle &\longrightarrow |110\rangle
\end{aligned}
$$

$$
CCX = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
$$

$$
ccX(x_1, x_2, x_3) = (x_1, x_2, (x_1 \wedge x_2) \oplus x_3)
$$

It computes the AND operation, the ancilla (fixed) bit x_3 is set to 0.

$$ccX(x_1, x_2, 0) = (x_1, x_2, x_1 \wedge x_2)$$

It computes the XOR operation, the bit x_1 is set to 1.

$$ccX(1, x_2, x_3) = (1, x_2, x_2 \oplus x_3)$$

It computes the NOT operation on x_3.

$$ccX(1, 1, x_3) = (1, 1, \neg x_3)$$

It computes the $NAND$ operation, the ancilla (fixed) bit x_3 is set to 1.

$$ccX(x_1, x_2, 1) = (x_1, x_2, \neg(x_1 \wedge x_2))$$

It computes the $FANOUT$ operation (the value of bit x_2 is copied into x_3).

$$ccX(1, x_2, 0) = (1, x_2, x_2)$$

Controlled NOT Gate - cX

$$CX|00\rangle = |00\rangle \qquad CX|01\rangle = |01\rangle,$$
$$CX|10\rangle = |11\rangle \qquad CX|11\rangle = |10\rangle.$$

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Controlled not operation. The second qubit is only flipped in the case that the first qubit is 1. In this case a not operation X on the second qubit is executed.

$$cX(1, x) = (1, \neg x_1)$$

$$cX(0, x) = (0, x_1)$$

Fan-out operation. For this operation the second qubit has to be zero. In this case the value of the first qubit is copied into the second one.

$$cX(x_1, 0) = (x_1, x_1)$$

SWAP Gate - SWAP

$$SWAP|00\rangle = |00\rangle \qquad SWAP|01\rangle = |10\rangle,$$
$$SWAP|10\rangle = |01\rangle \qquad SWAP|11\rangle = |11\rangle.$$

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Controlled SWAP Gate - cS

$$CSWAP|000\rangle \longrightarrow |000\rangle \qquad CSWAP|100\rangle \longrightarrow |100\rangle$$
$$CSWAP|001\rangle \longrightarrow |001\rangle \qquad CSWAP|101\rangle \longrightarrow |110\rangle$$
$$CSWAP|010\rangle \longrightarrow |010\rangle \qquad CSWAP|110\rangle \longrightarrow |101\rangle$$
$$CSWAP|011\rangle \longrightarrow |011\rangle \qquad CSWAP|111\rangle \longrightarrow |110\rangle$$

$$CSWAP = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If control qubit is zero no swap operation is performed

$$CSWAP(0, x_1, x_2) = (0, x_1, x_2)$$

If control qubit is one swap operation on is performed

$$CSWAP(1, x_1, x_2) = (1, x_2, x_1).$$

Gates for one Qubit

- Pauli gates: *X* gate (NOT gate), *Y* gate and the *Z* gate
- Pauli Y Gate: $Y|0\rangle \dashrightarrow i|1\rangle$ $Y|1\rangle \dashrightarrow i|0\rangle$ $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
- Pauli Z Gate: $Z|0\rangle \dashrightarrow |0\rangle$ $Z|1\rangle \dashrightarrow -|1\rangle$ $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
- S Gate : $S|0\rangle \dashrightarrow |0\rangle$ $S|1\rangle \dashrightarrow i|1\rangle$ $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

Rotation Gates

- T Gate

$$T|0\rangle \longrightarrow |0\rangle \qquad T|1\rangle \longrightarrow e^{i\frac{\pi}{4}} \cdot |1\rangle = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right) \cdot |1\rangle$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

with T gate being the square root of the S gate

$$S = T^2$$

- T† Gate: qc.tdg(qubit)

$$T^\dagger|0\rangle \longrightarrow |0\rangle \qquad T^\dagger|1\rangle \longrightarrow e^{-i\frac{\pi}{4}} \cdot |1\rangle = \left(\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}\right) \cdot |1\rangle$$

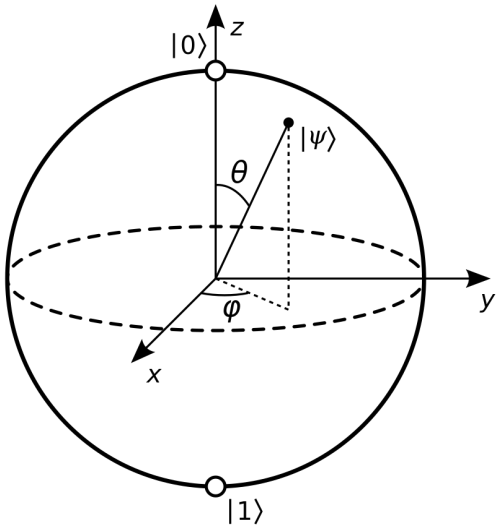
$$T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix}$$

with $T^\dagger$ gate being the square root of the S gate

$$S = \left(T^\dagger\right)^2$$

Parameterized Rotation Gates

- A parameterized rotation gate is a parameterized gate with the parameter being the amount of rotation to be performed around the three axes



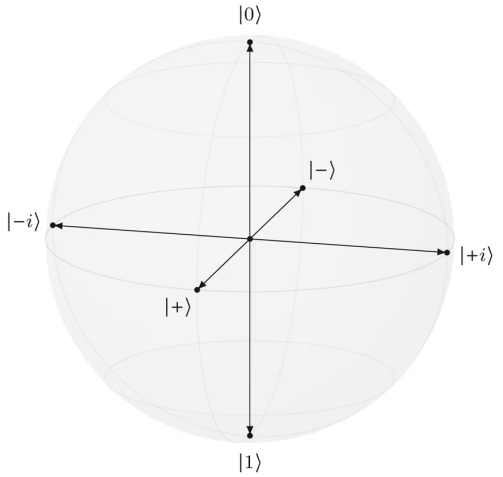
Bloch sphere representation of a qubit. The probability amplitudes for the superposition state,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ are given by}$$

$$\alpha = \cos\left(\frac{\theta}{2}\right) \text{ and}$$

$$\beta = e^{i\varphi}\sin\left(\frac{\theta}{2}\right)$$

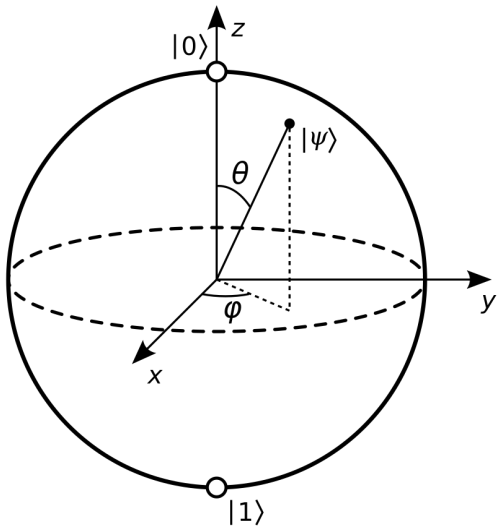
https://bloch.kherb.io



RX Gate

- The RX gate performs a rotation of one qubit along the x-axis by the rotation angle θ. The rotation angle is in radians.

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$



With $\theta = \pi$

$$RX|0\rangle \longrightarrow -i|1\rangle \qquad RX|1\rangle \longrightarrow -i|0\rangle$$

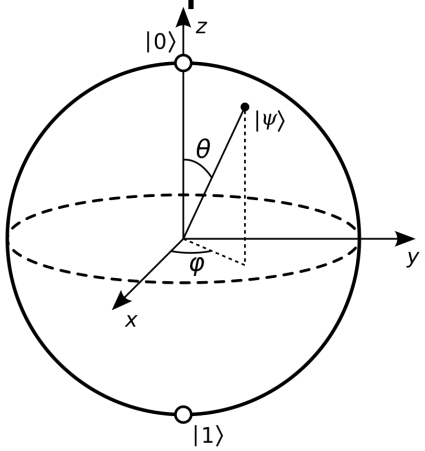
With $\theta = \pi/2$

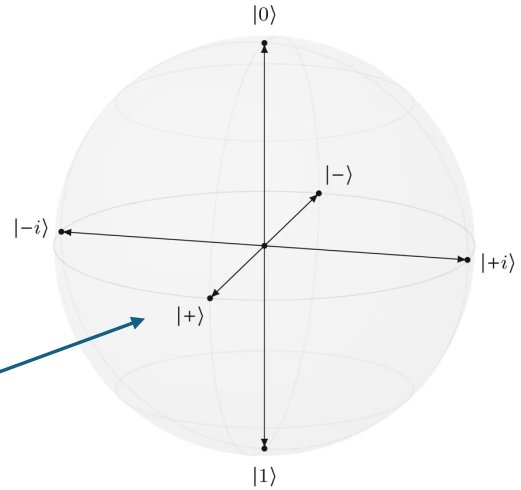
$$RX|0\rangle \longrightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{i}{\sqrt{2}} \cdot |1\rangle \qquad RX|1\rangle \longrightarrow -\frac{i}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$$

RY Gate

- The RY gate performs a rotation of one qubit along the y-axis by the rotation angle θ

$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$





With $\theta = \pi$

$$RY|0\rangle \longrightarrow |1\rangle \qquad RY|1\rangle \longrightarrow -|0\rangle$$

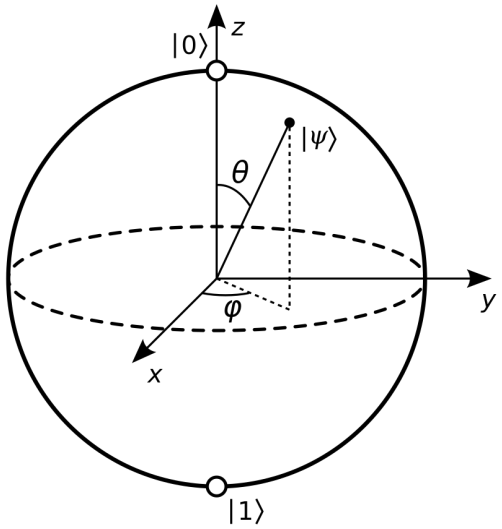
With $\theta = \pi/2$

$$RY|0\rangle \longrightarrow \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \qquad RY|1\rangle \longrightarrow -\frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$$

RZ Gate

- The RZ gate performs a rotation of one qubit along the z-axis by the rotation angle θ

$$R_Z(\phi) = \begin{pmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{pmatrix}$$



With $\phi = \pi$

$$RZ|0\rangle \longrightarrow -i|0\rangle \qquad RZ|1\rangle \longrightarrow i|1\rangle$$

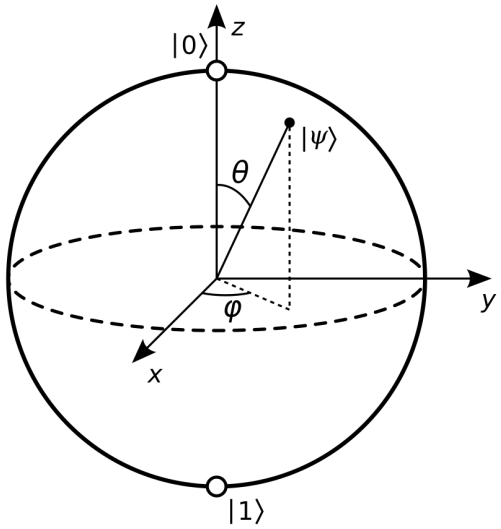
With $\phi = \pi/2$

$$RZ|0\rangle \longrightarrow \left(\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}\right) \cdot |0\rangle \qquad RZ|1\rangle \longrightarrow \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right) \cdot |1\rangle$$

U Gate

- U gate is a single-qubit rotation gate with 3 Euler angles
- The U gate performs a rotation of one qubit along the axes by the rotation angles θ, ϕ and λ

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{-i\cdot\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\cdot\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\cdot(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$



Phase Gate - P

$$P|0\rangle \longrightarrow |0\rangle \qquad P|1\rangle \longrightarrow e^{i\cdot\lambda}\cdot|1\rangle$$

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\cdot\lambda} \end{pmatrix}$$

Controlled-U Gates

- The controlled-U gates performs a controlled operation described by the unitary matrix U

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

- The cU gate defines a quantum gate on two qubits and can be represented by a unitary matrix CU

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix}$$

$$CU|00\rangle \longrightarrow |00\rangle \qquad CU|01\rangle \longrightarrow |01\rangle$$

$$CU|10\rangle \longrightarrow |1\rangle \otimes U|0\rangle \qquad CH|11\rangle \longrightarrow |1\rangle \otimes U|1\rangle$$

Controlled Phase Gate

- The target and the control qubit can be exchanged, since the phase is only applied if both values are one
  - qc.cp(lambda, *control, target*)

$$CP(|00\rangle \longrightarrow |00\rangle \qquad CP|01\rangle \longrightarrow |01\rangle$$
$$CP|10\rangle \longrightarrow |10\rangle \qquad CP|11\rangle \longrightarrow e^{i\cdot\lambda} \cdot |11\rangle$$

$$CP(\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\cdot\lambda} \end{pmatrix}$$

Universality

- Unitary decomposition is the process of translating an arbitrary unitary operator into a universal set of single and two-qubit gates.
- Unitary decomposition is necessary because it is not otherwise possible to execute an arbitrary quantum operator.
- The Z-Y decomposition theorem provides a way of expressing a unitary operation *U* on a single qubits by parametrized rotation gates

Z-Y Decomposition Theorem

- If *U* is a unitary operation on single qubit, then there exist real numbers *α, β, γ and δ*

$$U = e^{i\cdot\alpha} R_Z(\beta) \cdot R_Y(\gamma) \cdot R_Z(\delta)$$

  - For example, in quantum machine learning a Hadamard-like transformation S_p is used to store the patterns

$$S_p = \begin{pmatrix} \sqrt{\frac{p-1}{p}} & \frac{1}{\sqrt{p}} \\ \frac{-1}{\sqrt{p}} & \sqrt{\frac{p-1}{p}} \end{pmatrix}$$

  - *and can be represented by the parametrized U gate with $\phi = \pi$, $\lambda = \pi$*

$$U(\theta, \pi, \pi) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$U(\theta, \pi, \pi) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$\sin\left(\frac{\theta}{2}\right) = \frac{1}{\sqrt{p}}$$

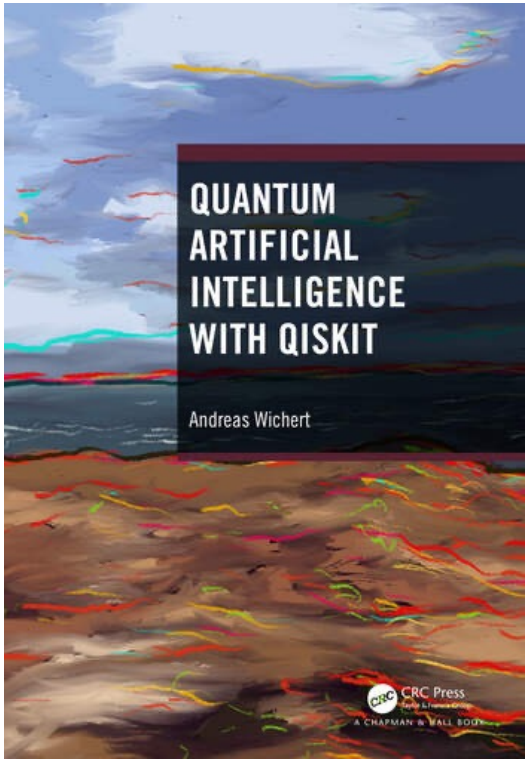
$$\theta = \arcsin\left(\frac{1}{\sqrt{p}}\right) \cdot 2$$

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{p-1}{\sqrt{p}}}$$

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

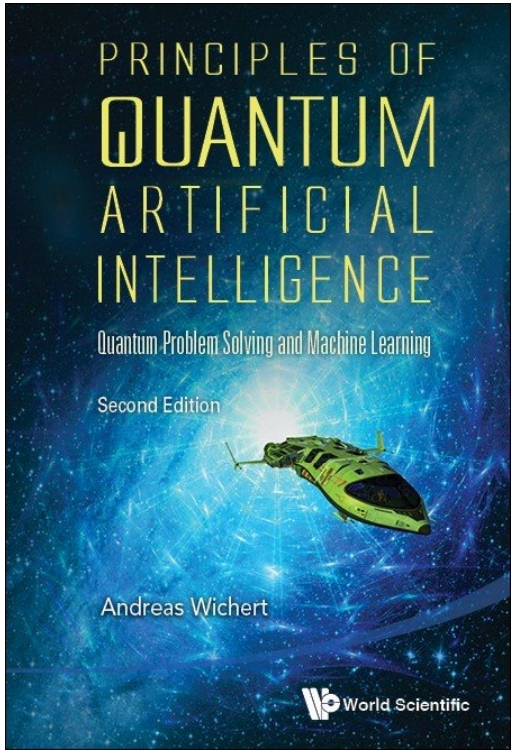
Quantum Circuits

- The depth of a quantum circuit indicates the quantum gates that can be executed in parallel to compute the entire circuit
  - The depth is related to the time to execute the circuit.
  - Indicated with the command *qc.size()*.
- The defined quantum circuit is converted to the target quantum computer for execution (if not simulated).
- This process is called **transpilation** contrary to the compilation task in classical programing languages.
  - It is a translation problem determined by the available set of operations and *the hardware topology*.
- Finding an optimal decomposition of a quantum circuit into the present quantum gates is a complex problem.
  - For example, almost all quantum gates that operate on *n* qubits require an **exponential** number of *2* qubits gates by a **naive** implementation



- Chapter 3

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



- Chapter 8

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