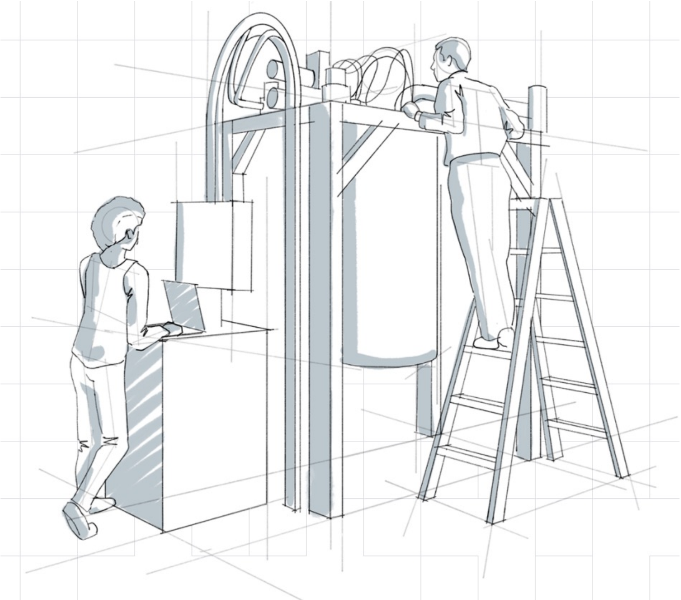
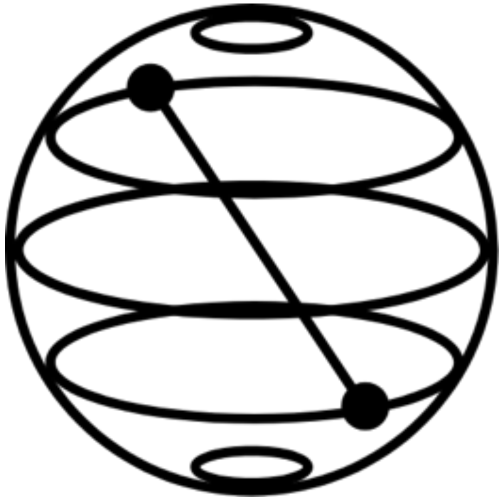
Telecom Valley | SophiaConf

100% OPEN SOURCE - Conférences et Workshops

IBM

First steps into IBM Quantum Computing





IBM Client Center Montpellier

JM Torres | torresjm@fr.ibm.com

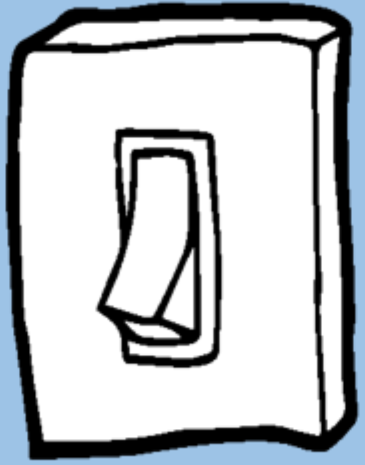
June 2021

Part 1

Guided tour of the IBM Quantum devices,

and Quantum « Hello World! »

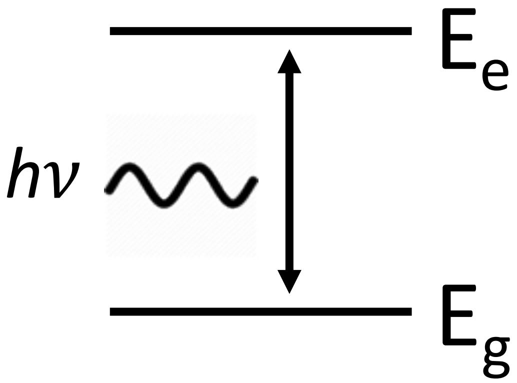
0



1

classical bit

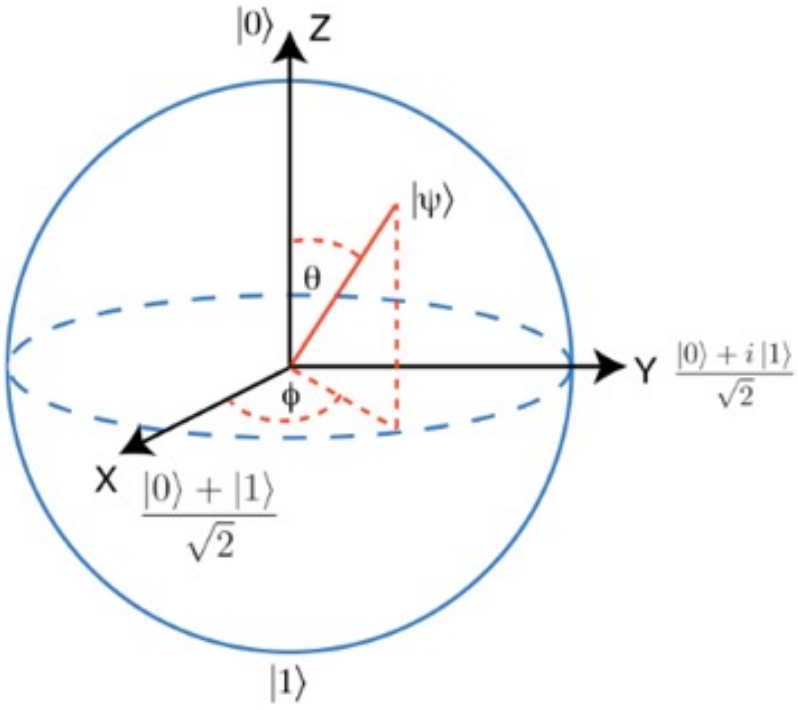
qubit : quantum bit



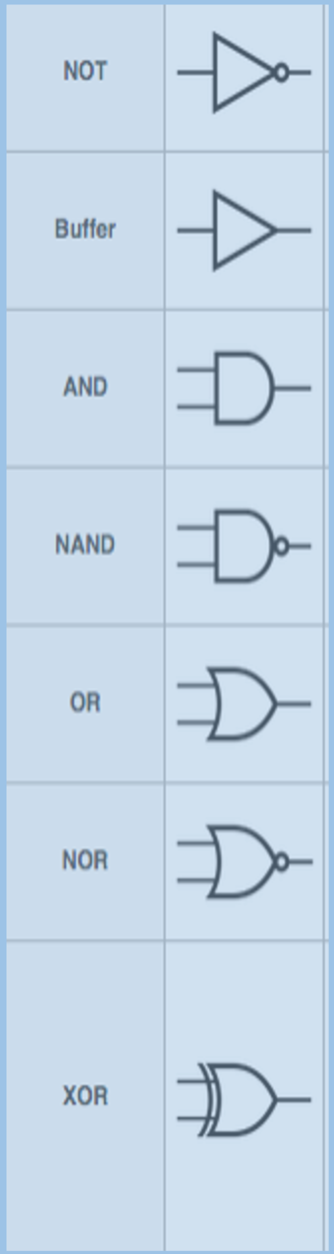
$|e\rangle \sim |1\rangle$

$|g\rangle \sim |0\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$



The Bloch sphere



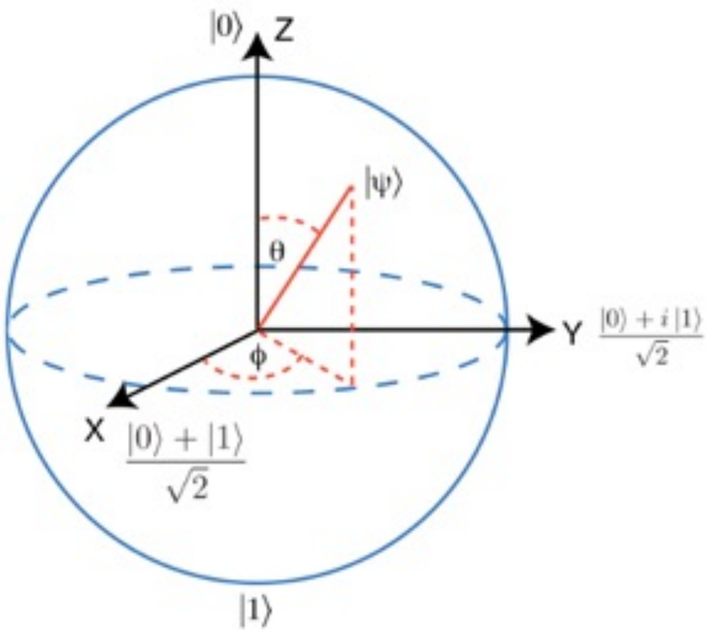
Controlling a qubit

« PAULI » Operators

rotation around x axis	⊕	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\mathrm{qc.x(qr[n])}$	RX	$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
rotation around y axis	Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\mathrm{qc.y(qr[n])}$	RY	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
rotation around z axis	Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\mathrm{qc.z(qr[n])}$	RZ	$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$
Identity	I	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\mathrm{qc.id(qr[n])}$		

Bloch Sphere

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle$$



superposition (X+Z) Hadamard gate | H | $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ | $\mathrm{qc.h(qr[n])}$

CNOT : flips target qubit according to control qubit state.

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

measurement

measures quantum state in quantum register into classical register (0/1)

More operators are available from qiskit (S, T, swap, cswap, ccx, cz, …)

quantum operators :

H operator (Hadamard)

$$|0\rangle \xrightarrow{\boxed{H}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

creates equal superposition of states $|0\rangle$ and $|1\rangle$

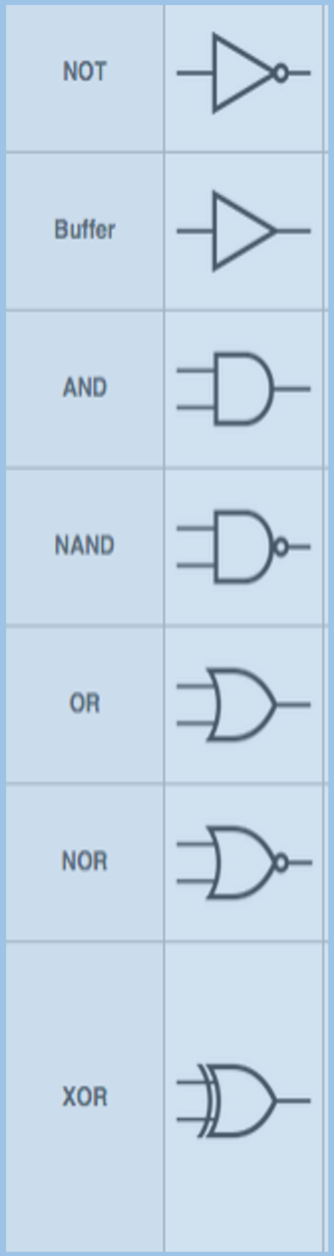
Control-Not operation

controler

target

target qubit state is flipped if and only if the control qubit is in state $|1\rangle$

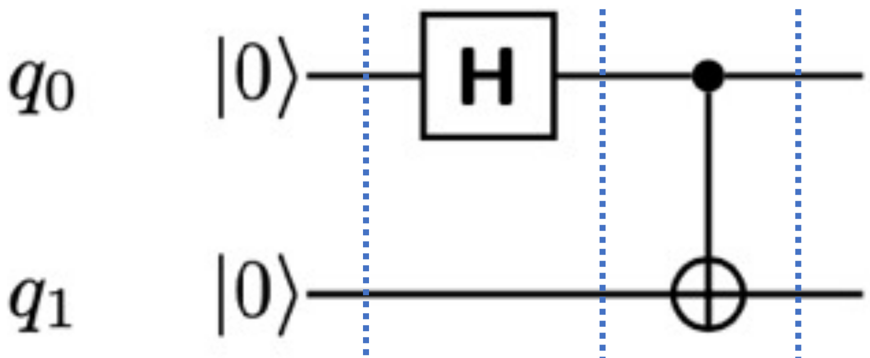
creates quantum entanglement of two qubits



classical operators

Hello World!

print('Hello World!')



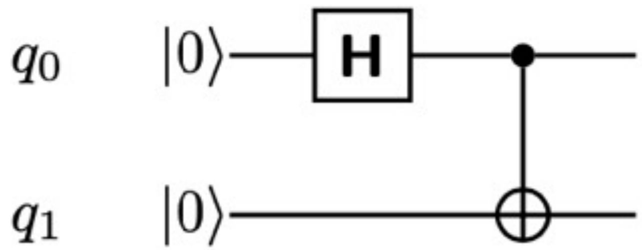
$$|00\rangle$$

$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Hello World! example

Hadamard gate applied to q_0, then Control-Not applied to q_1, controlled by q_0



This produces the « Bell-State »

With words :

System starts in $|00\rangle$ (both q_0 and q_1 in state $|0\rangle$).

Then q_0 goes through Hadamard and gets into equal superposition of $|0\rangle$ and $|1\rangle$.

After q_0 controls q_1, the state of q_1 is in a superposition of $|0\rangle$ & $|1\rangle$, (q_1 stays at $|0\rangle$ when q_0 is $|0\rangle$, and q_1 goes $|1\rangle$ when q_0 is $|1\rangle$).

So : both q_0 and q_1 are in $|0\rangle$ (state $|00\rangle$) or both q_0 and q_1 are in $|1\rangle$ (state $|11\rangle$).

Our system is in equal superposition of $|00\rangle$ and $|11\rangle$.

The two qubits are entangled: if you measure one of the qubits, you immediately know the state of the other.

In between :

System starts in $|00\rangle$,then :

$$H|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle+|10\rangle)$$

Applying CNOT: left part of the sum stays as is, right term goes to $|11\rangle$ resulting state is $\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$.

One can easily prove there are no $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta}$ such that:

$$(\boldsymbol{\alpha}|0\rangle+\boldsymbol{\beta}|1\rangle)\otimes(\boldsymbol{\gamma}|0\rangle+\boldsymbol{\delta}|1\rangle)=\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$$

So, the resulting state is not the product of two quantum states, instead this is an entangled state.

With maths :

Stage 1 (H on q0) :

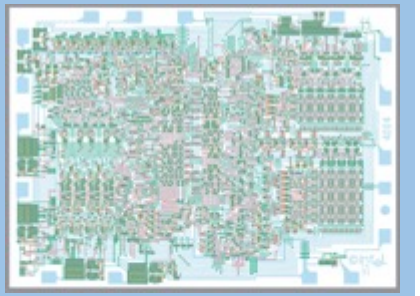
$$(H\otimes I)|00\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$$

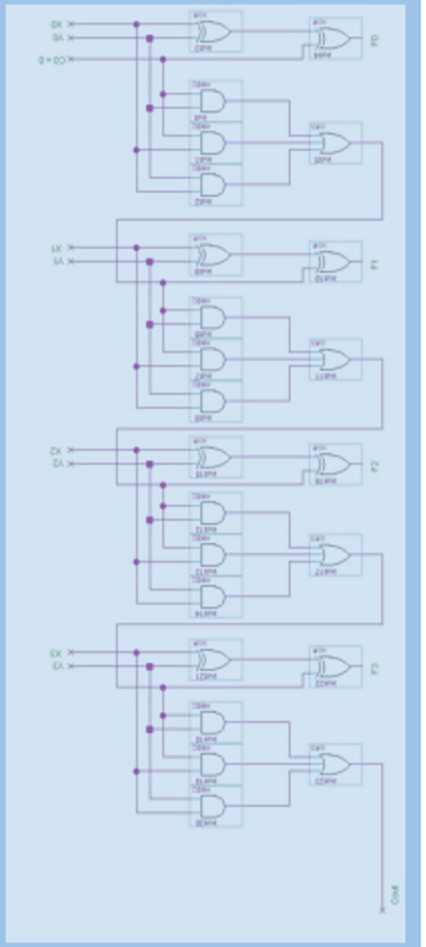
Stage 2: CNOT(0,1)

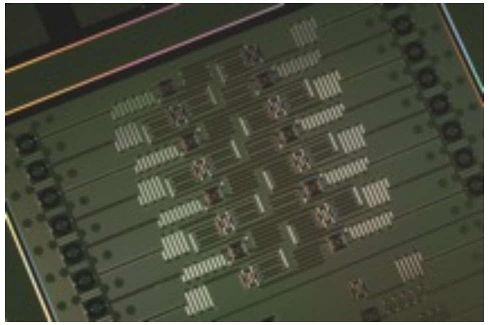
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \times \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$$

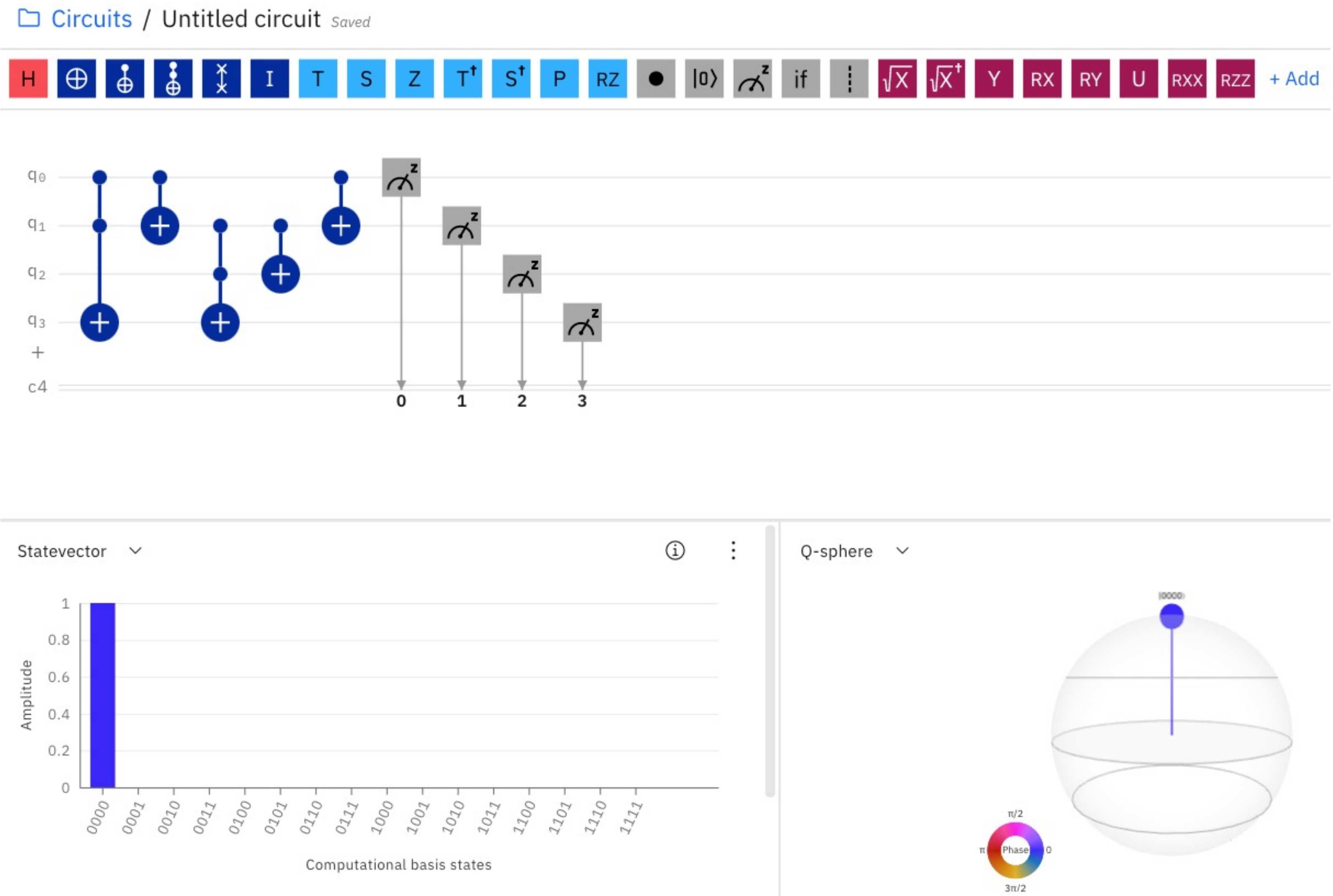
$$= \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$$

Quantum Circuit









print('Hello World!')

Demo : Bell state on a quantum machine

Part 2

using qiskit library to run quantum program with Python.

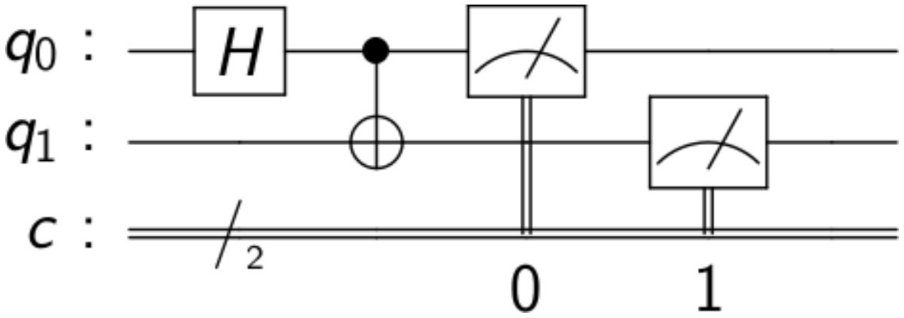
print('Hello World!')

Programing

In [1]:

```
from qiskit import QuantumCircuit, Aer, execute                 # imports
backend = Aer.get_backend('qasm_simulator')                      # select a device for execution

qc = QuantumCircuit(2,2)                                         # create a quantum circuit having 2 qubits and 2 cbits

qc.h(0)                                                          # buid the circuit by
qc.cx(0,1)                                                       # adding operators on qubits

qc.measure([0,1],[0,1])                                          # use measurement gates to retrieve results

d = execute(qc,backend).result().get_counts()                    # execute qc on backend and get cumulated results into
print(d)                                                         # a dictionnary
```

```
{'00': 491, '11': 533}
```



print('Hello World!')

Historic Quantum Algorithms

Deutsch	1985	$2 \rightarrow 1$
Bernstein-Vazirani	1992	$N \rightarrow 1$
Deutsch-Josza	1992	$2^{N-1} + 1 \rightarrow 1$
Shor	1994	$e^N \rightarrow (\mathrm{LogN})^3$
Grover	1996	$N \rightarrow \sqrt{N}$

More and new ones on quantumalgorithmzoo.org/

qiskit applications modules :

Chemistry

Ground State Energy
Dipole Moment
Excited States

Finance

Portfolio Optimization
Risk Analysis
Pricing

AI

Training
Classification

Optimization

Max
TSP
Graph Partition
Stableset
Clique
Exact Cover
Set Packing
Vertex Cover

Quantum Algorithms

VQE ; QAOA ; Dynamics ; QPE/IQPE ; Amplitude Estimation
Grover ; SVM Q Kernel ; SVM Variational ; Simon ; Deutsch-Josza ; Bernstein-Vazirani

print('Hello World!')

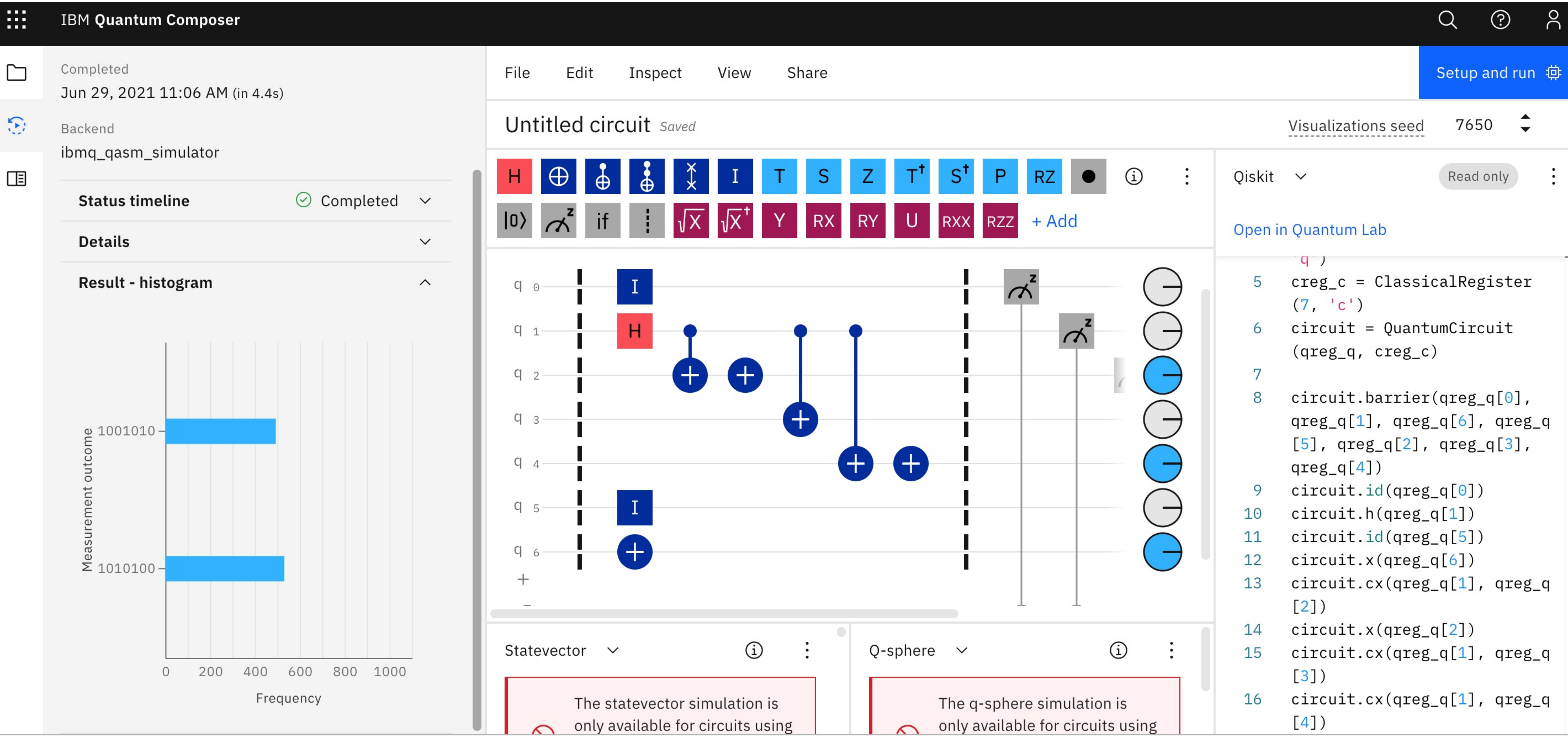
« 3-SAT » using Grover Algorithm demo

Part 3

Try it !

try it !
Superpose your name initials

Letter	binary ASCII value	Letter	binary ASCII value
A	100 0001	N	100 1110
B	100 0010	O	100 1111
C	100 0011	P	101 0000
D	100 0100	Q	101 0001
E	100 0101	R	101 0010
F	100 0110	S	101 0011
G	100 0111	T	101 0100
H	100 1000	U	101 0101
I	100 1001	V	101 0110
J	100 1010	W	101 0111
K	100 1011	X	101 1000
L	100 1100	Y	101 1001
M	100 1101	Z	101 1010


IBM Quantum Composer
Completed
Jun 29, 2021 11:06 AM (in 4.4s)
Backend
ibmq_qasm_simulator
Status timeline
Completed
Details
Result - histogram
Measurement outcome
1001010
1010100
0
200
400
600
800
1000
Frequency
File
Edit
Inspect
View
Share
Setup and run
Untitled circuit
Saved
Visualizations seed
7650
H
I
T
S
Z
T†
S†
P
RZ
|0⟩
if
√X
√X†
Y
RX
RY
U
RXX
RZZ
+ Add
Qiskit
Read only
Open in Quantum Lab
q 0
q 1
q 2
q 3
q 4
q 5
q 6
5 creg_c = ClassicalRegister (7, 'c')
6 circuit = QuantumCircuit (qreg_q, creg_c)
7
8 circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[6], qreg_q [5], qreg_q[2], qreg_q[3], qreg_q[4])
9 circuit.id(qreg_q[0])
10 circuit.h(qreg_q[1])
11 circuit.id(qreg_q[5])
12 circuit.x(qreg_q[6])
13 circuit.cx(qreg_q[1], qreg_q [2])
14 circuit.x(qreg_q[2])
15 circuit.cx(qreg_q[1], qreg_q [3])
16 circuit.cx(qreg_q[1], qreg_q [4])
Statevector
Q-sphere
The statevector simulation is only available for circuits using
The q-sphere simulation is only available for circuits using

In [5]:

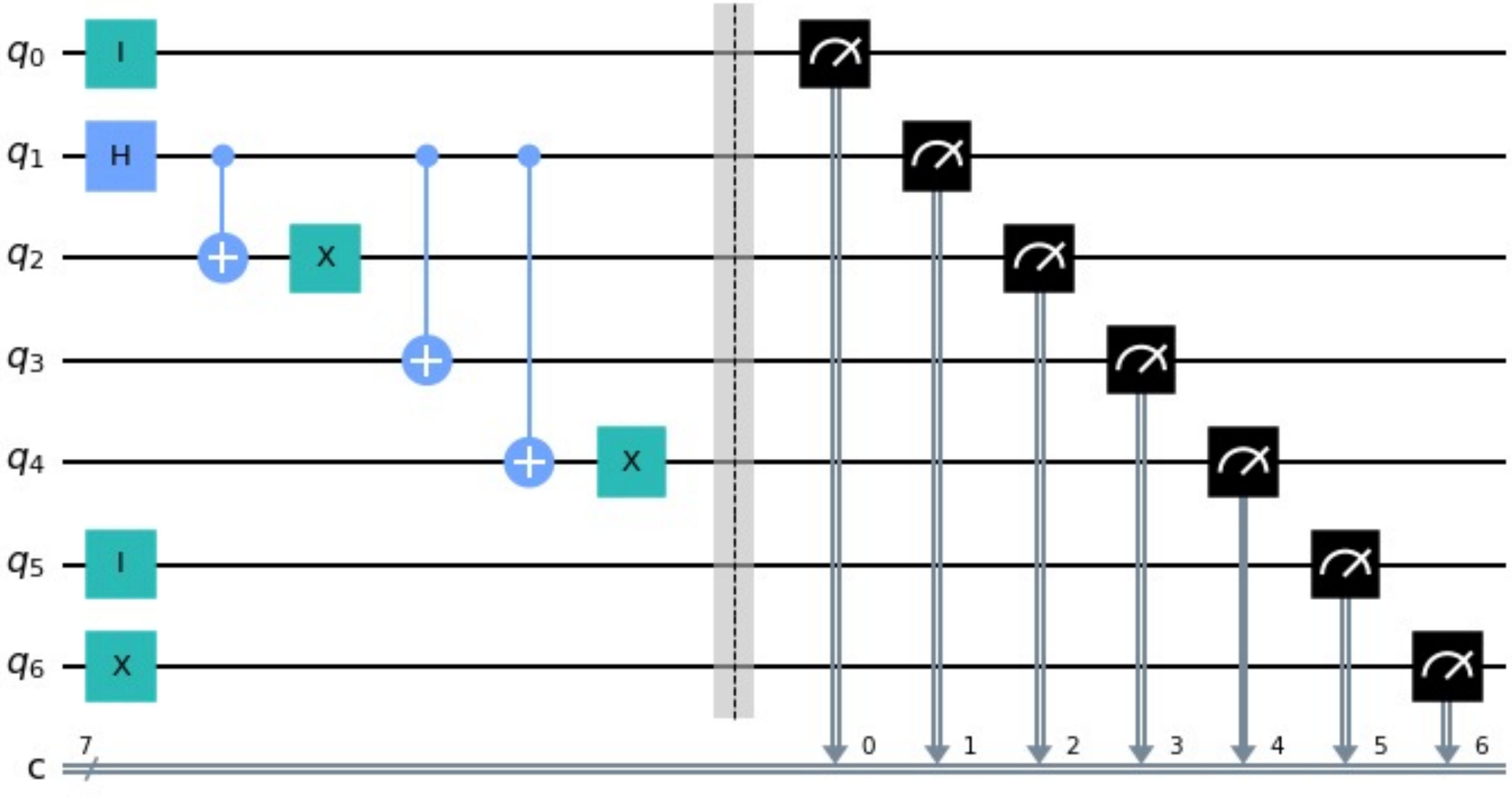
```
import numpy as np
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *

# Loading your IBM Quantum account(s)
provider = IBMQ.load_account()
```

In [6]:

```python
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import execute
from numpy import pi

qreg_q = QuantumRegister(7, 'q')
creg_c = ClassicalRegister(7, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.id(qreg_q[0])
circuit.h(qreg_q[1])
circuit.id(qreg_q[5])
circuit.x(qreg_q[6])
circuit.cx(qreg_q[1], qreg_q[2])
circuit.x(qreg_q[2])
circuit.cx(qreg_q[1], qreg_q[3])
circuit.cx(qreg_q[1], qreg_q[4])
circuit.x(qreg_q[4])
circuit.barrier(qreg_q[4], qreg_q[0], qreg_q[1], qreg_q[2], qreg_q[3], qreg_q[5], qreg_q[6])
circuit.measure(qreg_q[0], creg_c[0])
circuit.measure(qreg_q[1], creg_c[1])
circuit.measure(qreg_q[2], creg_c[2])
circuit.measure(qreg_q[3], creg_c[3])
circuit.measure(qreg_q[4], creg_c[4])
circuit.measure(qreg_q[5], creg_c[5])
circuit.measure(qreg_q[6], creg_c[6])
```

In [7]:

```
circuit.draw()
```



In [19]:

```
provider = IBMQ.get_provider(hub='ibm-q')
backend  = provider.get_backend('ibmq_qasm_simulator')

job = execute(circuit,backend)
results = job.result()

d = (results.get_counts())
plot_histogram(d)
```