

Assignment 04: Superposition and entanglement

Objectives

- You will be able to understand the basics of quantum theory including superposition and entanglement
- You will learn how to create superposition and how it can be used in practice
- You will know how to entangle qubits by using the Quantum Composer

Requirements

- Notebook or Desktop Computer with access to the internet
- Access to IBM Quantum Lab (<https://lab.quantum-computing.ibm.com/>)
- Access to IBM Quantum Composer (<https://quantum-computing.ibm.com/composer>)

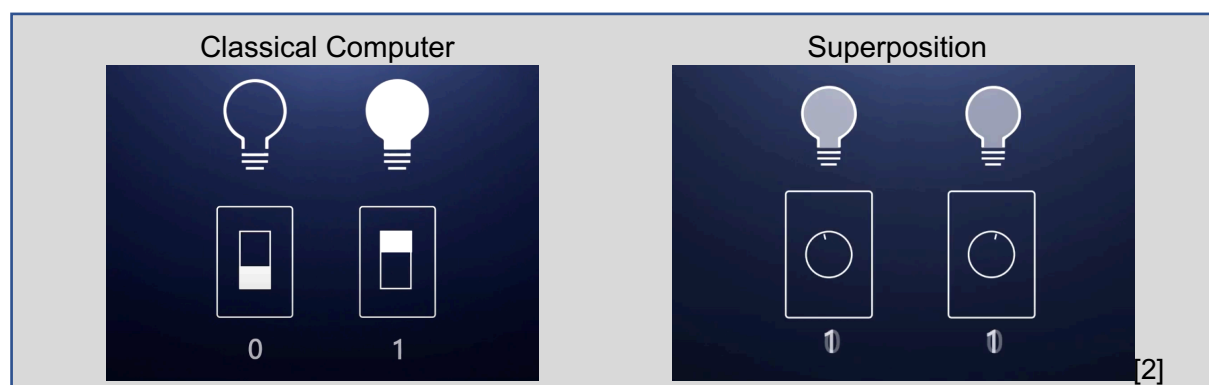
Solution Steps

Quantum objects do not follow the traditional rules of physics and thus often contradict to our tangible real-world experiences. Quantum Computers leverage several quantum mechanical phenomena to create states that scale exponentially with number of qubits.

1. Superposition

Classical Computers manipulate 1s and 0s to generate computation (CPU) while Quantum Computers use quantum bits or qubits (QPU) [1]. Just like Classical Computers, Quantum Computers use 1s and 0s, but qubits have a third state called “superposition” that allows them to represent a 1 or a 0 with some probability at the same time.

While you can imagine a Classical Computer switching from 0 to 1 as a light bulb turning from off to on, superposition of a Quantum Computer can be illustrated as a dimmer of a light bulb which enables a spectrum between a dark and bright light bulb [2].



Let's go back to our navigation case of assignment 2. We found out that the blue route is the most efficient route while the green route is the shortest route.

Similar to our approach a Classical Computer would calculate the efficiency of every single route separately. Superposition allows the Quantum Computer to calculate the efficiency of all potential routes simultaneously. Due to this approach the Quantum Computer comes up with the right solution faster compared to the Classical Computer. Furthermore, superposition enables the Quantum Computer to incorporate real time traffic changes of all routes in its' calculations to make the optimal decision just in time.



To illustrate the power of superposition let's play an interactive Quantum Coin Game.

You and your friend Taylor cannot agree whether you are going to take the blue route or the green route to get to the Time Out Market in Lisbon. While Taylor wants to take the green route as it is the shortest one, you try to convince Taylor to take the blue route because of its' energy efficiency.

To simplify your decision you are playing a coin game.

Here are the basic rules:

You start and you may either turn the coin by applying a X-gate/NOT-gate or leave it as it is by applying an Id-gate (Identity-gate). The moves are hidden and hence not revealed to Taylor. In the next step Taylor may also turn the coin or leave it as it is. You have the third and final move. Now the coin gets revealed. If it shows "Heads", you win; if it shows "Tails", Taylor wins. "Heads" is encoded by "0", "Tails" encoded by "1".

To be able to play the game we need to open a new file in IBM Quantum Lab and import the basic functions that are required for the game.

```
Assignment 5_Superposit x
from qiskit import ClassicalRegister, QuantumRegister, QuantumCircuit;
from qiskit import execute, BasicAer;
from qiskit.tools.visualization import plot_histogram, circuit_drawer;
from ipywidgets import interact
```

Execute the following functions that define the possible moves for Taylor and you and the order of the moves.

```
Assignment 5_Superposit x
def Move_Y1(move_Y1): global moveY1; moveY1=move_Y1;
def Move_T1(move_T1): global moveT1; moveT1=move_T1;
def Move_Y2(move_Y2): global moveY2; moveY2=move_Y2;
```

The function “who_wins” defines that you are the winner if the coin shows 0, indicating “Heads”, while Taylor wins when the coin shows 1 (“Tails”). Unfortunately copying the following code sometimes changes the formatting of the function, which causes errors in executing the code. Therefore, please make sure that all code lines are exactly displayed as in the following code box. To indent the lines please use the tab button.

```

Assignment 5_Superposit x
+ > < < > < > < > Code < >
Qiskit v0.35.0 (ipykernel)

def who_wins(counts):
    if len(counts)==1 :
        print('The winner is', 'you' if ("0" in counts) else 'Taylor')
    else:
        count0=counts["0"];
        count1=counts["1"];
        print('The coin is in superposition of |0> and |1>');
        print('You win with probability', "%.1f%%" % (100.*count0/(count0+count1)));
        print('Taylor wins with probability', "%.1f%%" % (100.*count1/(count0+count1)));
    return()

```

Ready to play?

Round 1:

In the first round both players can only choose between a X-gate to turn the coin and an Id-gate to leave the coin as it is.

The following code provides a drop-down menu where you can decide your first move. In this example an Id-gate was chosen to be your first move. After selecting the move in the drop down menu, you do not have to re-run the code, just make your selection in the drop down menu and continue with the next code box.

```

Assignment 5_Superposit x
+ > < < > < > < > Code < >
Qiskit v0.35.0 (ipykernel)

interact(Move_Y1, move_Y1={'id Gate':0,'X Gate':1});

move_Y1 id Gate

```

Just like before, please make the first move for Taylor.

```

Assignment 5_Superposit x
+ > < < > < > < > Code < >
Qiskit v0.35.0 (ipykernel)

interact(Move_T1, move_T1={'id Gate':0,'X Gate':1});

move_T1 id Gate

```

Now you make the second move. In this case a X-gate was chosen.

```

Assignment 5_Superposit x
+ > < < > < > < > Code < >
Qiskit v0.35.0 (ipykernel)

interact(Move_Y2, move_Y2={'id Gate':0,'X Gate':1});

move_Y2 X Gate

```

The following code creates a Quantum Circuit (qc) based on one qubit (q) and a classical bit (c) to measure the qubit. Depending on the moves of Taylor and you, the code applies the gates chosen in the previous three steps and measures the outcome.

```
Assignment 5_Superposit x
+ > % □ □ ▶ □ ↺ ↻ Code ⌵ ⌂ Qiskit v0.35.0 (ipykernel) ⚙
```

```
q = QuantumRegister(1);
c = ClassicalRegister(1);
qc = QuantumCircuit(q, c);
backend = BasicAer.get_backend('qasm_simulator');

qc.id(q[0]) if (moveY1 == 0) else qc.x(q[0]);
qc.id(q[0]) if (moveT1 == 0) else qc.x(q[0]);
qc.id(q[0]) if (moveY2 == 0) else qc.x(q[0]);

qc.measure(q, c);
qc.draw(output='mpl')
```

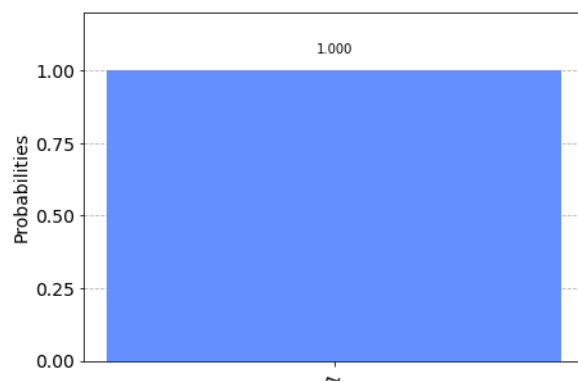
To know who won the game please execute the following code.

```
Assignment 5_Superposit x
+ > % □ □ ▶ □ ↺ ↻ Code ⌵ ⌂ Qiskit v0.35.0 (ipykernel) ⚙
```

```
job = execute(qc, backend, shots=2000);
result = job.result();
counts = result.get_counts(qc);
print(counts);
who_wins(counts);
plot_histogram(counts)
```

Based on the moves Taylor won the game and therefore you would take the green route to the Time Out Market.

```
{'1': 2000}
The winner is Taylor
```



However, as you know that the blue route would lead to lower CO2 emissions you can convince Taylor to play a second round of the Coin Game. This time you want to use the power of superposition to always win the game.

Round 2:

Besides applying X-gates and Id-gates, you have the chance to also apply Hadamard-gates.

Please continue to work in the IBM Quantum Lab file you created for round 1 of this game. You make again the first move and apply an Id-, X- or H-gate. In this example we decided to apply a H-gate in the first move.

```
Assignment 5_Superposit x
+ % □ ▶ □ ↺ ↻ Code ▼ ⓘ Qiskit v0.35.0 (ipykernel) ⓘ
```

```
interact(Move_Y1, move_Y1={'id Gate':0, 'X Gate':1, 'H Gate':2});
```

move_Y1 H Gate ▼

Now it's time for Taylor to make the next move. Like in the first round, Taylor can only choose between an Id-gate or X-gate.

```
Assignment 5_Superposit x
+ % □ ▶ □ ↺ ↻ Code ▼ ⓘ Qiskit v0.35.0 (ipykernel) ⓘ
```

```
interact(Move_T1, move_T1={'id Gate':0, 'X Gate':1});
```

move_T1 X Gate ▼

Please decide which gate should be applied for your second move.

```
Assignment 5_Superposit x
+ % □ ▶ □ ↺ ↻ Code ▼ ⓘ Qiskit v0.35.0 (ipykernel) ⓘ
```

```
interact(Move_Y2, move_Y2={'id Gate':0, 'X Gate':1, 'H Gate':2});
```

move_Y2 id Gate ▼

Analogously to the first round the following code creates a quantum circuit (qc) with one qubit (q) and a classical bit (c) to measure the qubit. The gates that were chosen in the previous three steps are then applied to the qubit.

```
Assignment 5_Superposit x
+ % □ ▶ □ ↺ ↻ Code ▼ ⓘ Qiskit v0.35.0 (ipykernel) ⓘ
```

```
q = QuantumRegister(1);
c = ClassicalRegister(1);
qc = QuantumCircuit(q, c);
backend = BasicAer.get_backend('qasm_simulator');

if moveY1 == 0 : qc.id(q[0]);
elif moveY1 == 1 : qc.x(q[0]);
elif moveY1 == 2 : qc.h(q[0]);

if moveT1 == 0 : qc.id(q[0]);
elif moveT1 == 1 : qc.x(q[0]);

if moveY2 == 0 : qc.id(q[0]);
elif moveY2 == 1 : qc.x(q[0]);
elif moveY2 == 2 : qc.h(q[0]);

qc.measure(q, c);
qc.draw(output='mpl')
```

To get to know the chances of winning for you and Taylor please execute the following code.

```
Assignment 5_Superposit x
+ > % □ □ ▶ □ ↺ ↻ Code ▼ ⓘ Qiskit v0.35.0 (ipykernel) ⓘ

job = execute(qc, backend, shots=200);
result = job.result();
counts = result.get_counts(qc);

print(counts);
who_wins(counts);

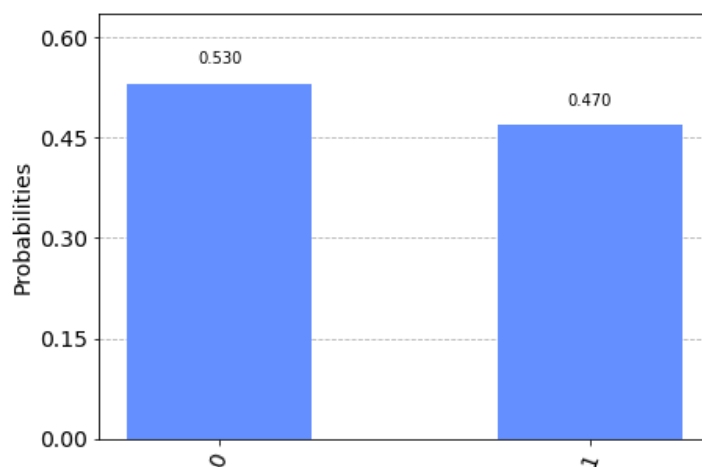
plot_histogram(counts)
```

The strategy that was used in this exemplary code can be summarized as follows:

- You: H-gate
- Taylor: X-gate
- You: Id-gate

As depicted in the following graph this strategy leads to an almost 50/50 chance between you and Taylor of winning the coin game due to superposition.

```
{'0': 106, '1': 94}
The coin is in superposition of  $|0\rangle$  and  $|1\rangle$ 
You win with probability 53.0%
Taylor wins with probability 47.0%
```



Task: Identify a strategy that always allows you to win the coin toss game by using superposition (15 min.)

Now it is your turn! With the knowledge you gained about gates and superposition please try different moves to identify a strategy that allows you to always win the game and therefore ensures that you take the route with the lowest CO2 emissions. For changing the strategy you only need to change the selections in the drop down menu, without re-running the corresponding code box. To check who wins the game with the new strategy only re-run the code boxes marked in red color.

Hint: Whenever you have problems to come up with the right strategy by trial and error, try to think about all possible combinations first and then try each of those strategies. You can also use a decision tree to visualize the potential strategies. In total there are 18 different strategies and two of them enable you to always win the game.

2. Entanglement

Another special behavior of qubits is the so-called entanglement. The topic of quantum entanglement is at the heart of the disparity between classical and quantum physics: entanglement is a primary feature of quantum mechanics lacking in classical mechanics. As we never see entanglement in the real world, Einstein also referred to it as “spooky action at a distance” [3].

When two qubits, qubit A and qubit B are entangled, they are basically paired up so that they represent the same wave function, and hence yield the same outcome if they are measured [4]. This indicates that the quantum states of the entangled qubits cannot be described independently from each other, even though the qubits might be separated by a large distance [5].

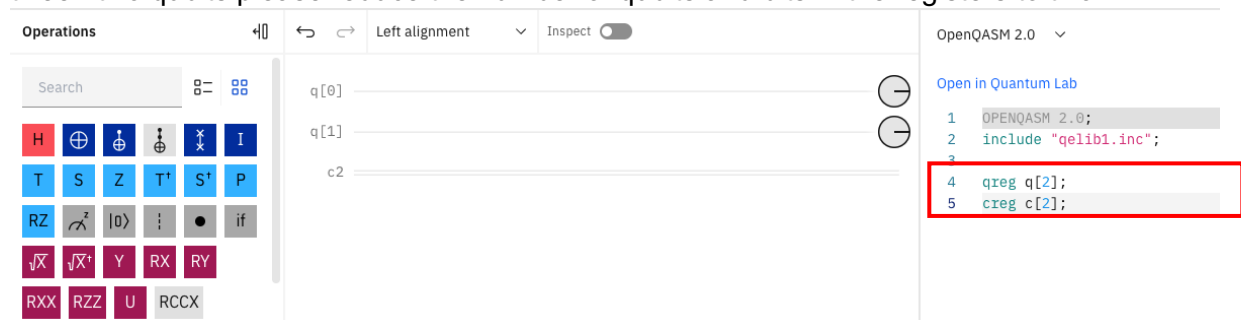
Hint: Think of entanglement as a bonding between twins, where each can “sense” the other.

To get more insights about how entanglement works and why it can change the internet we use to know nowadays by enabling a quantum internet please watch the following video:

<https://youtu.be/soywlog1Fdk>

In the next exercise we are going to learn how to entangle qubits using the Quantum Composer. <https://quantum-computing.ibm.com/composer>

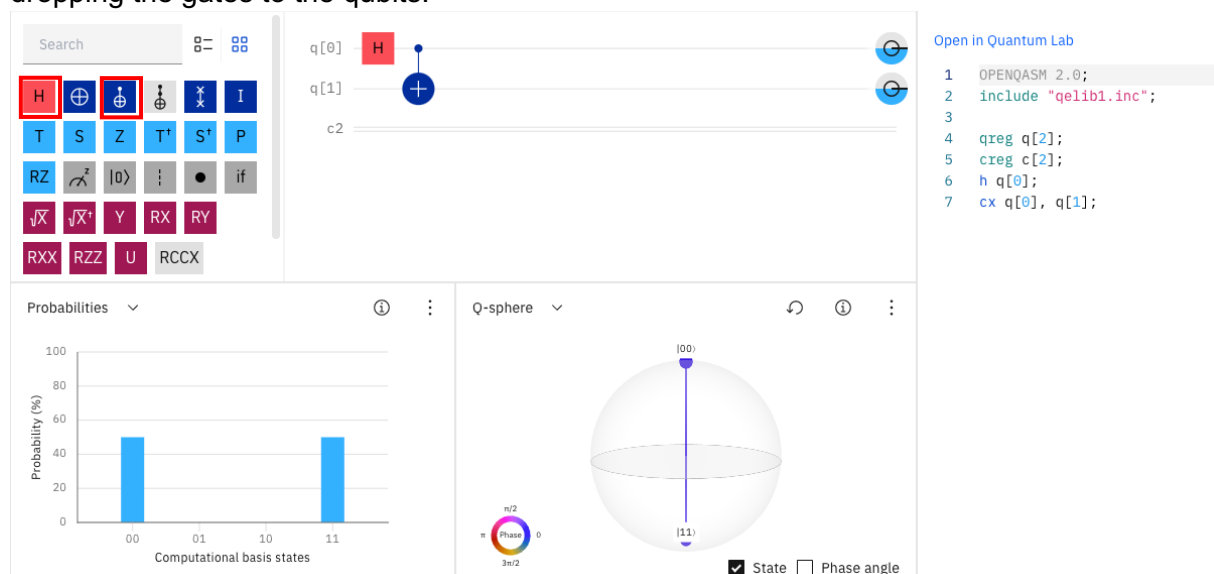
Open the Quantum Composer and create a new file. As we want to create entanglement between two qubits please reduce the number of qubits and bits in the registers to two.

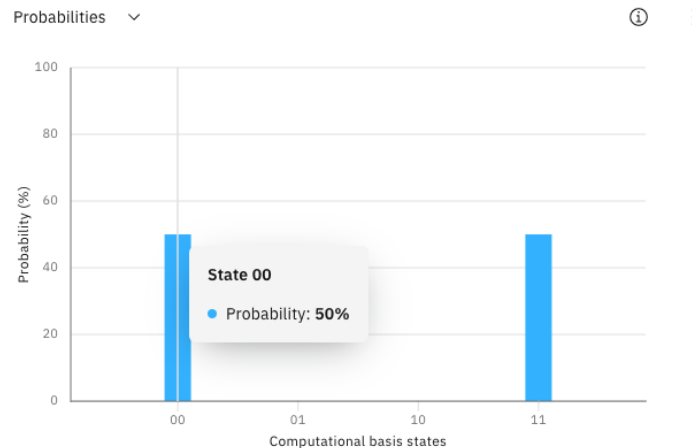


The screenshot shows the Quantum Composer interface. On the left, there's a panel with various quantum gates like H, T, S, Z, T†, S†, P, RZ, |0>, |1>, |00>, |11>, if, √X, √X†, Y, RX, RY, RXX, RZZ, U, and RCCX. The main workspace shows three qubits: q[0], q[1], and c2. On the right, there's a code editor with the following code:

```
1 OPENQASM 2.0;  
2 include "qelib1.inc";  
3  
4 qreg q[2];  
5 creg c[2];
```

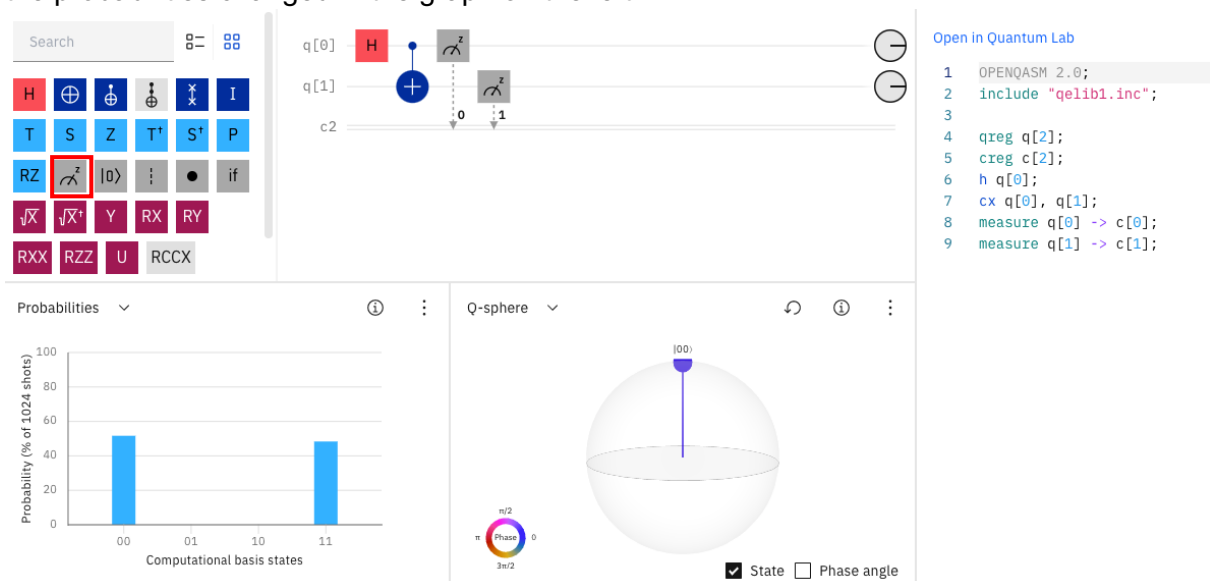
To create entanglement between both qubits we apply a H-gate on q0 and a controlled X-gate, where q0 is the control and q1 is the target qubit. You can apply the gates by dragging and dropping the gates to the qubits.

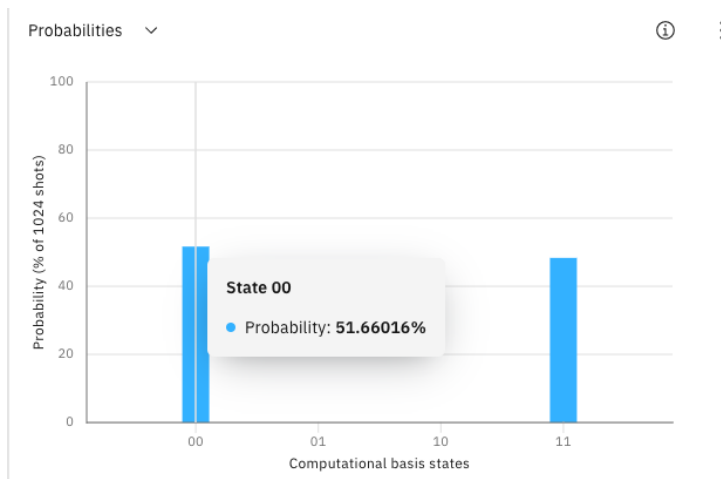




The entanglement leads to the phenomenon that both qubits are in the same state. When you use your cursor to go over the probabilities on the graph of the left-hand side you will notice that the state of both qubits are with exactly 50% probability 0 or 1. This is how the qubits should behave in theory when creating entanglement. However, we sometimes get different results in reality when we try to capture the states of qubits by transferring the states to bits and measuring them.

Let's find out how the theory differs from reality by adding measurement operations to our circuit. Again, please drag and drop the measurement symbol on both qubits and check how the probabilities changed in the graph on the left.



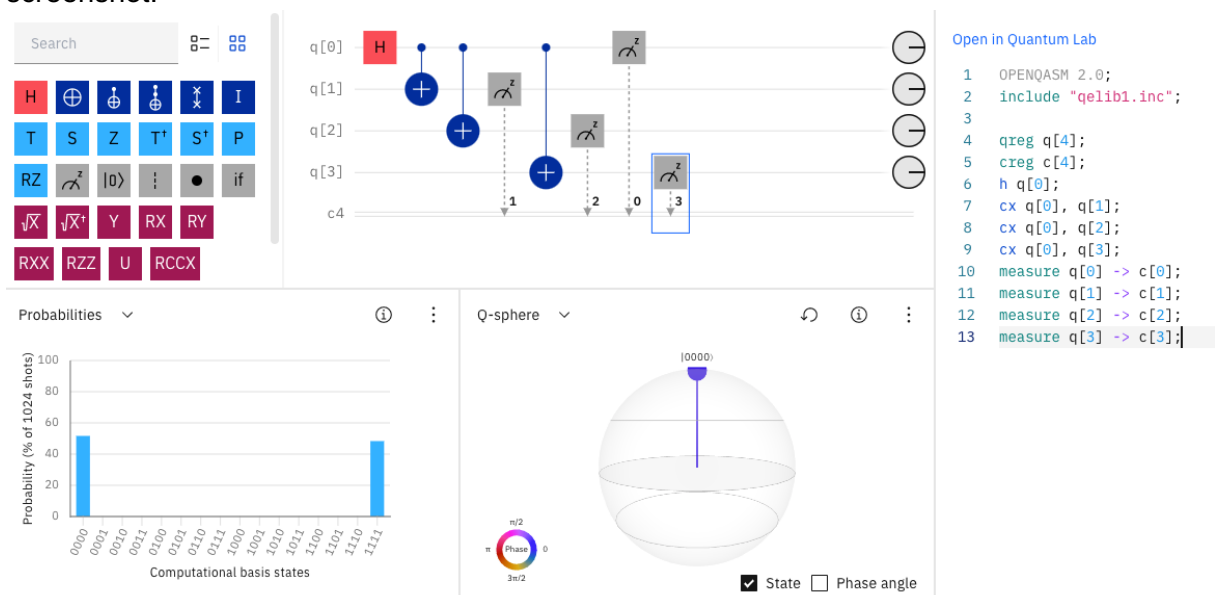


Measuring the states shows that in reality the probability of both states is not always perfectly even distributed. However, as the deviation between theory and reality is quite small, we can conclude that creating entanglement between qubits works quite stable.

Is entanglement limited to two qubits?

Let's find out by repeating the same experiment with a circuit containing four instead of two qubits.

To do so, please adapt your circuit in the Quantum Composer according to the following screenshot.



Again, you may have noticed that the distribution between the states 0 and 1 is not perfectly equal. However, as it is close to 50/50, we can conclude that the entanglement of four qubits is quite stable. Based on these results we can conclude that creating entanglement is not limited to a specific number of qubits.

Useful Resources for Own Research

Create superposition:

<https://quantum-computing.ibm.com/composer/docs/idx/guide/creating-superpositions>

Superposition and entanglement:

<https://www.quantum-inspire.com/kbase/superposition-and-entanglement/>

Quantum Superposition and what that means to Quantum Computation:

<https://becominghuman.ai/quantum-superposition-and-what-that-means-to-quantum-computation-3fbb5a711b9a>

Was macht einen Quantencomputer so mächtig? Teil 1: Qubits, Superposition und weitere Grundlagen:

<https://blog.iao.fraunhofer.de/was-macht-einen-quantencomputer-so-maechtig-teil-1-qubits-superposition-und-weitere-grundlagen/>

Can Quantum Codes Really Be Unbreakable?

<https://www.youtube.com/watch?v=TyomwLbYhig>

Quantum Entanglement: What it is & Why it is important in 2022:

<https://research.aimultiple.com/quantum-computing-entanglement/>

Retrospective

Please answer the following questions:

1. Which gate do we need to enable qubits to be in superposition?
2. What does the principle of superposition imply on quantum states?
3. What principle from quantum mechanics is involved when independent objects share common behaviors even across large distances?
4. Which gates have to be performed on qubits to create entanglement?
5. Is entanglement limited to two particles or qubits?

Sources

[1] <https://techcrunch.com/2022/03/08/quantware-will-build-you-a-custom-25-qubit-quantum-processor-in-30-days/?guccounter=1>

[2] https://www.youtube.com/watch?v=5p2_moQZJWo

[3] <https://www.space.com/31933-quantum-entanglement-action-at-a-distance.html>

[4] Brabazon, Anthony & O'Neill, Michael (2008): Natural Computing in Computational Finance, p. 92

[5] <https://scienceexchange.caltech.edu/topics/quantum-science-explained/entanglement>