Quantum Computing
Project Portfolio - Circuit Design

Deepansha Singh

Quantum Computing: Creating and Simulating Quantum Circuits

Deepansha Singh
deepansha.singh@gmail.com
West Windsor-Plainsboro High School South

Introduction:

Classical computers operate on bits: 0s and 1s. We have been using classical computers and supercomputers to solve many problems from different fields. However, even with such powerful computers, we are still not able to solve some problems in an efficient manner because of the constraints that come with classical computers.

We are now approaching a new tech era - the new tech disruption is no longer Artificial Intelligence or Machine Learning but instead Quantum Computing and Quantum ML/AI. Soon, quantum computers will be at our fingertips and we will be able to access much more powerful machines that can solve problems we deem impossible today. Quantum computing is a new form of computing at the intersection of mathematics (linear algebra), theoretical computer science, and quantum physics. Because of these new concepts and principles such as superposition and entanglement, bits operate on different properties altogether.

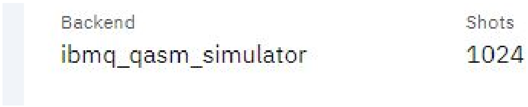
After watching the series Star Trek at a young age, I was fascinated with teleportation. When I came across the video “Einstein’s Spooky Action at a Distance,” I was first exposed to quantum computing and wanted to dive deeper into this field when I discovered that using these advanced concepts could soon make my vision of teleportation a reality. Ever since then, apart from excelling academically in the computer science and mathematics areas, I have created and simulated my own quantum circuits using IBM Quantum Experience. Just the thought of having access to a quantum computer has been very exciting to me as I’ve always wanted to work with them. I also co-founded the research platform QuantFlow with my friend, where we wrote a review paper in quantum computing highlighting the research breakthroughs and created arithmetic circuits using IBM Q Experience. Apart from immersing myself in the quantum computing realm, in the past, I have worked on data science+statistics and algorithms projects.

Abstract:

I’ve been utilizing IBM Quantum Experience platform to create and simulate quantum circuits. Using the circuit compose or programming through Python, I create the quantum circuits. There are both visualization tools to debug the circuit as well as a tool to edit the code and watch the circuit change graphically when I’m composing these quantum circuits. Then, I run experiments on IBM Q systems and simulators available. I can also run these

experiments on a real quantum computers. Please find information available at this link- https://www.ibm.com/quantum-computing/technology/experience/.

Simulator I used to run my circuit experiments & #shots / runs-



Backend	Shots
ibmq_qasm_simulator	1024

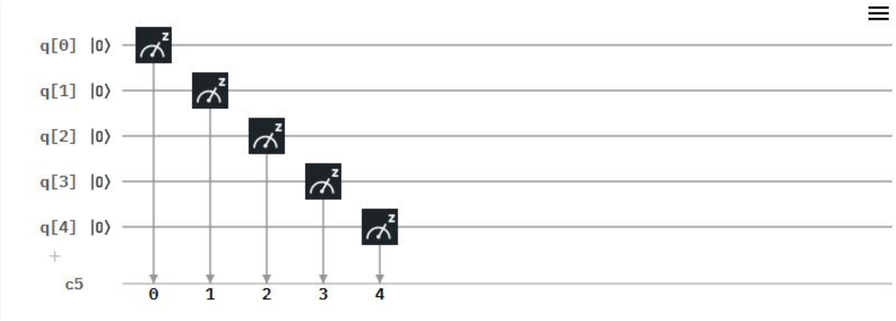
As stated in the introduction, quantum computing operates on linear algebra concepts, where vectors and matrices are used as the basis for the advanced mathematics operations. A vector in quantum computing denotes the state of qubit, which is a binary state and matrices transform vectors into new states through matrix multiplication, where the state of the qubit (vector quantity) is altered when it passes through the gate (mathematically represented as a gate).

I am including all of the gates I used for my quantum circuits and their corresponding matrices. The functions of each of these gates will be explained in great detail through my experiments and simulations of quantum circuits in the following pages of this document.

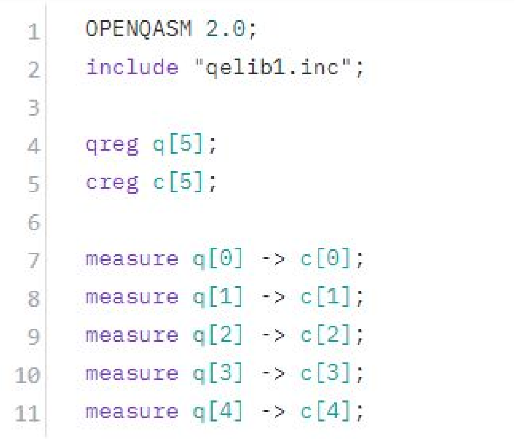
1) Measurement Gate
2) Pauli X Gate
3) Pauli Y Gate
4) Pauli Z Gate
5) Hadamard Gate
6) Controlled NOT (CX) Gate
7) CCX Gate
8) Controlled Hadamard Gate
9) Swap Gate
10) Rx Gate
11) Ry Gate
12) Rz Gate
13) U1 Gate
14) U2 Gate
15) U3 Gate
16) Controlled U1 Gate
17) Controlled U3 Gate
18) If Operation Gate
19) S Gate
20) Inverse S Gate
21) T Gate
22) Inverse T Gate

My Experiments: Some of My Quantum Circuits I have simulated (with code and output) using IBM Quantum Experience platform

Circuit 1- Simple measurement gate returns output for each of the 5 qubits in the circuit

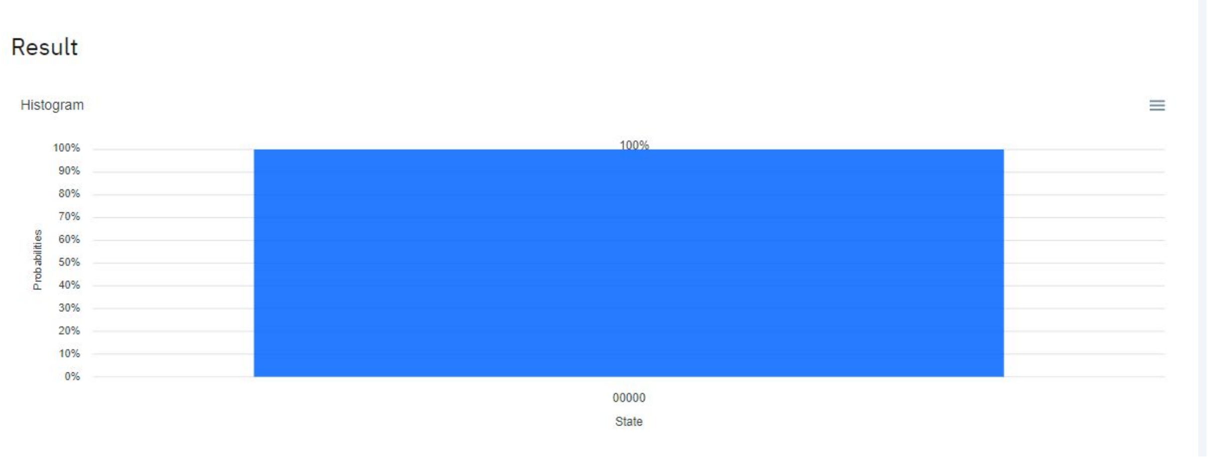


Code-

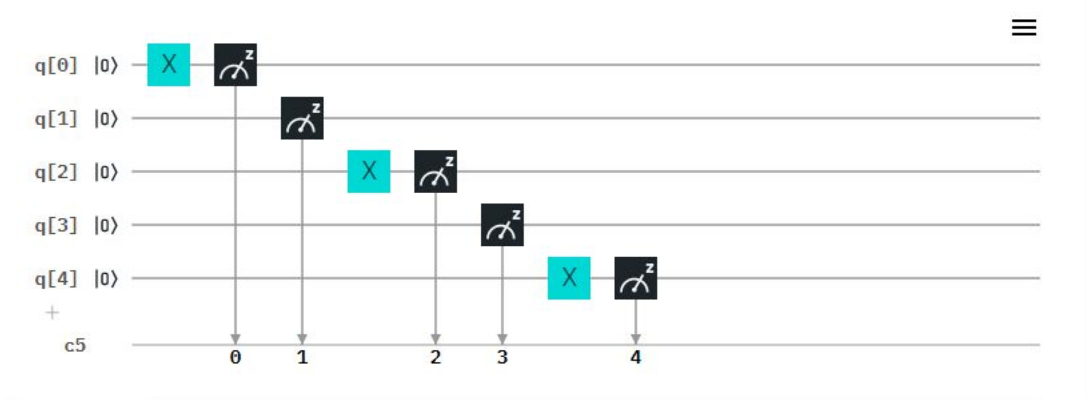


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

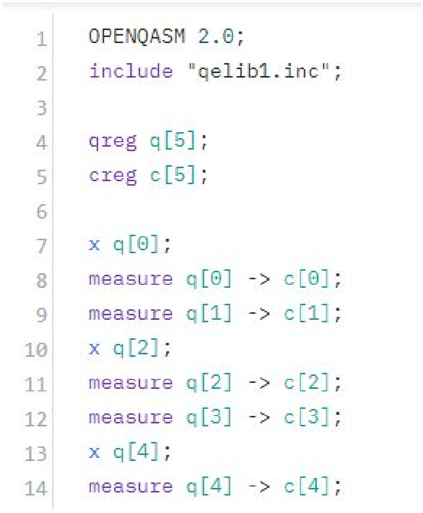
Output- Each of the 5 qubit's states will always be |0> (probability = 100%) because we didn't perform any special gate which will transform the output.



<u>Circuit 2</u>- Performs Pauli X gate on qubits 0, 2, 4 (Rotates vector along x axis of the Bloch sphere which is analogous to classical NOT gate since it simply operates with negation ⇒ if original state is 0, it will be flipped to 1 else if original state is 1 ⇒ it will be flipped to 0)

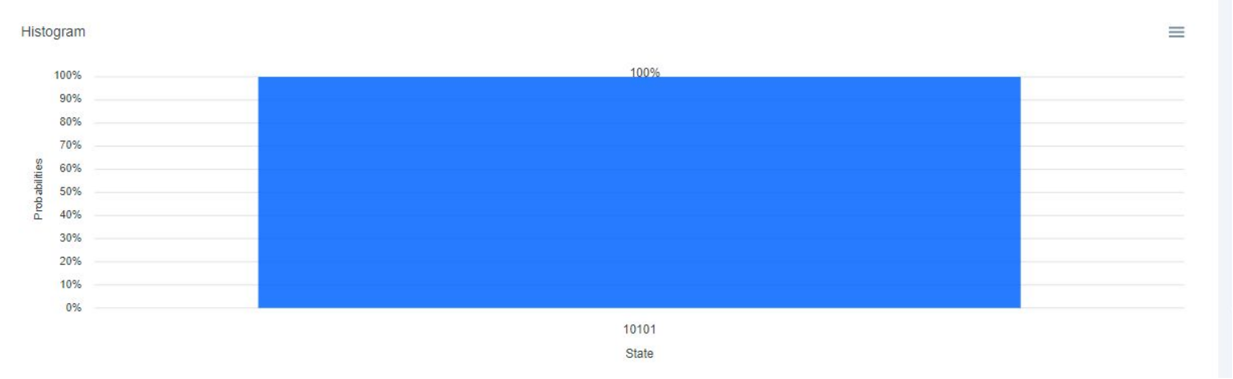


Code-

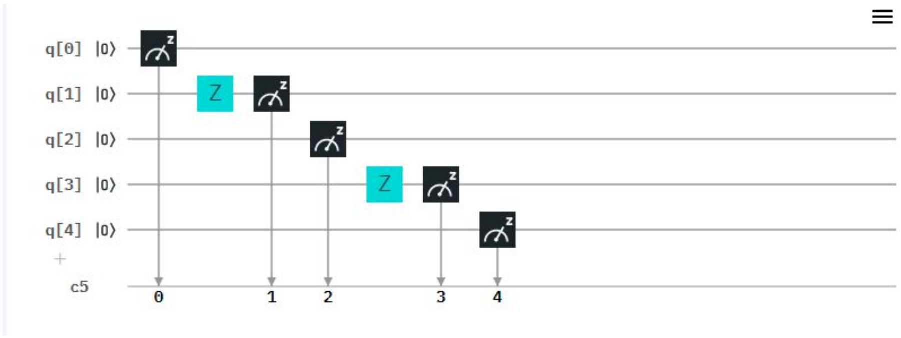


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

x q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
x q[2];
measure q[2] -> c[2];
measure q[3] -> c[3];
x q[4];
measure q[4] -> c[4];
```

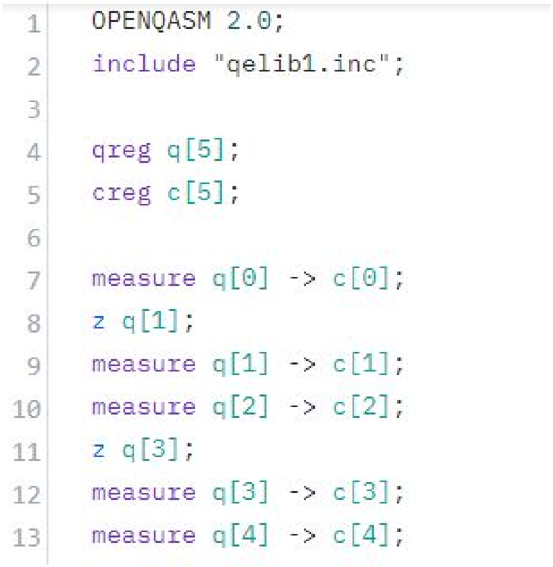
Output- The states for qubits 0, 2, and 4 will be flipped because Pauli X gate is applied on them.



Circuit 3 - Performs the Pauli Z gate on qubits 1, 3 (Rotates vector along the z axis of the Bloch sphere)

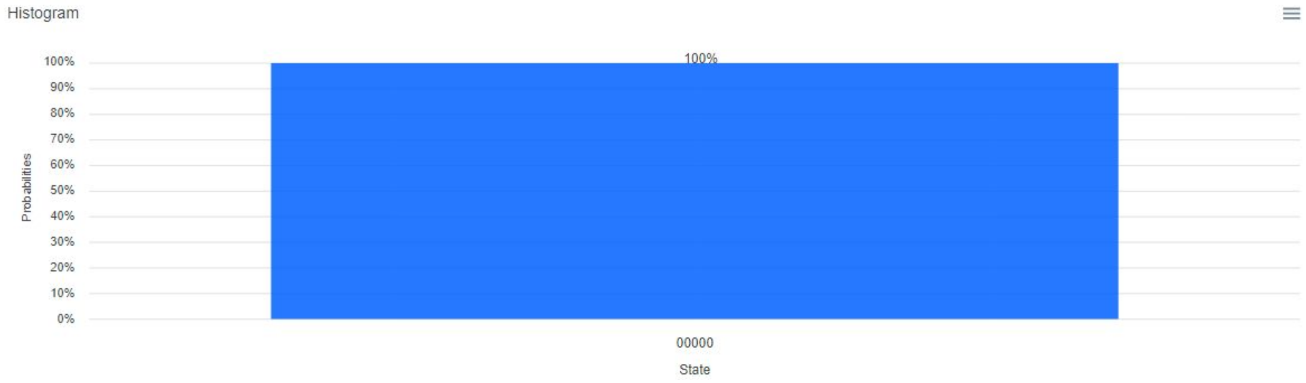


Code-

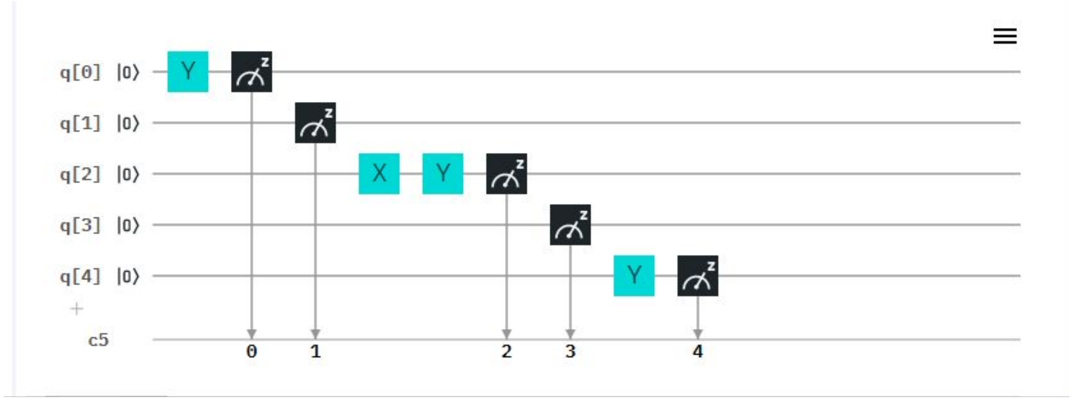


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

measure q[0] -> c[0];
z q[1];
measure q[1] -> c[1];
measure q[2] -> c[2];
z q[3];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

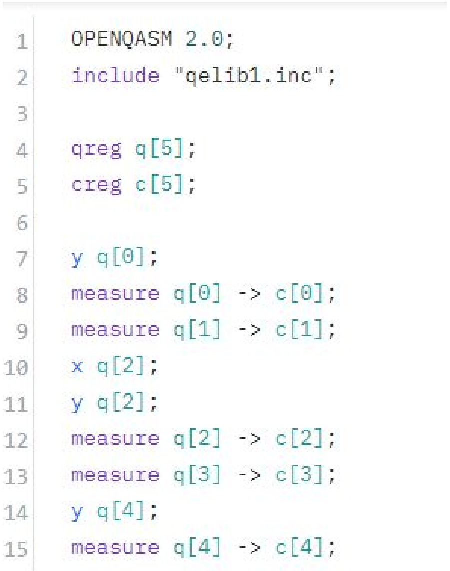
Output- None of the qubit's states will change since graphically they will not move towards either the |0> or |1> state (with current input, rotating about z axis doesn't make them closer to either state).



Circuit 4 - Performs Pauli Y gate on qubits , which rotates the qubit along the y axis by 180 degrees (this is equivalent to multiplying the states of the qubits by the imaginary number i

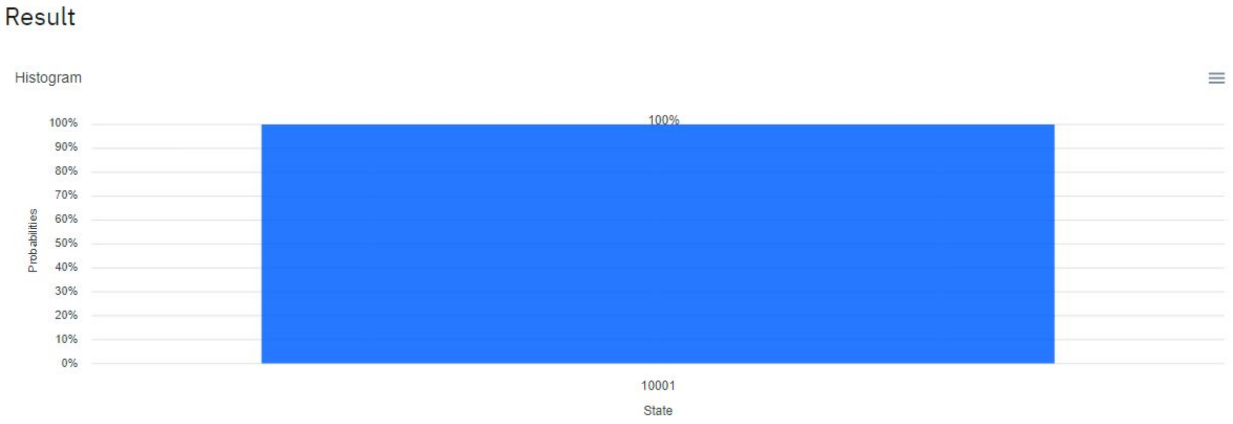


Code-

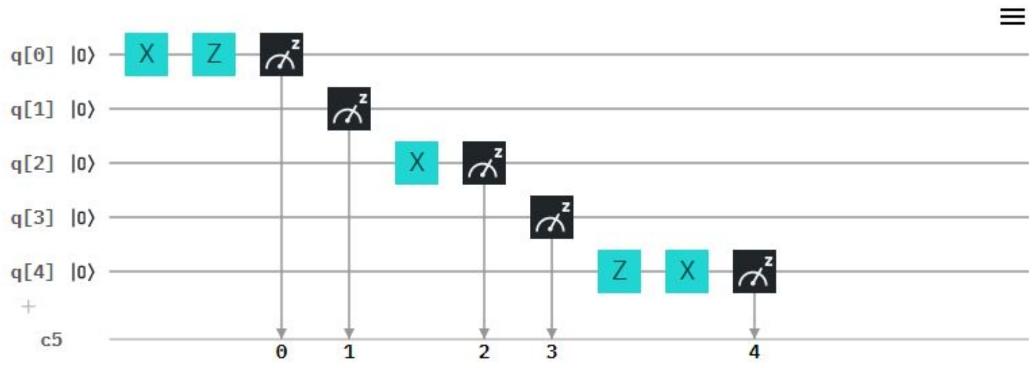


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

y q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
x q[2];
y q[2];
measure q[2] -> c[2];
measure q[3] -> c[3];
y q[4];
measure q[4] -> c[4];
```

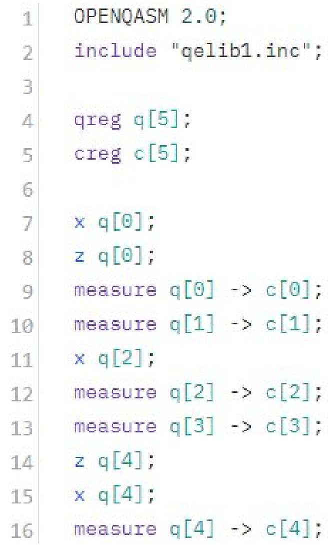
Output- Only qubits 0 and 5 will change state since rotating about the y axis by 180 degrees will always lead them to have the |1> state.



<u>Circuit 5</u> -Combination of Pauli X and Z gates on the 5 qubits in the circuit

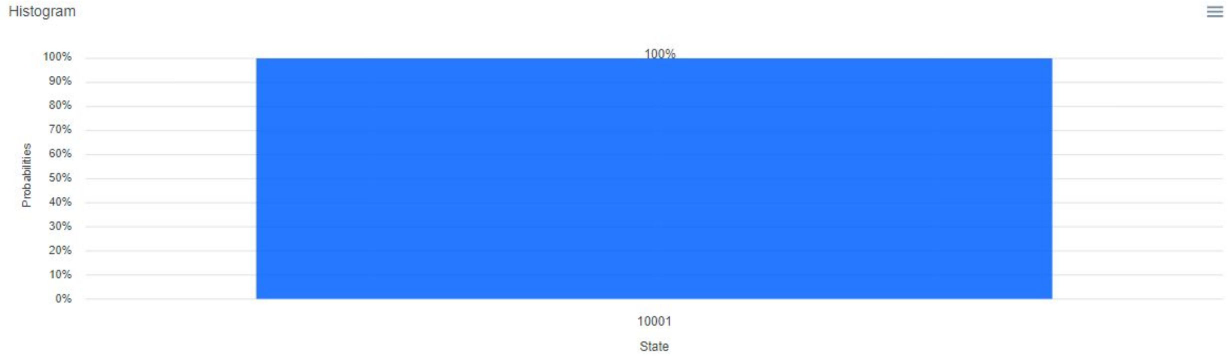


Code-

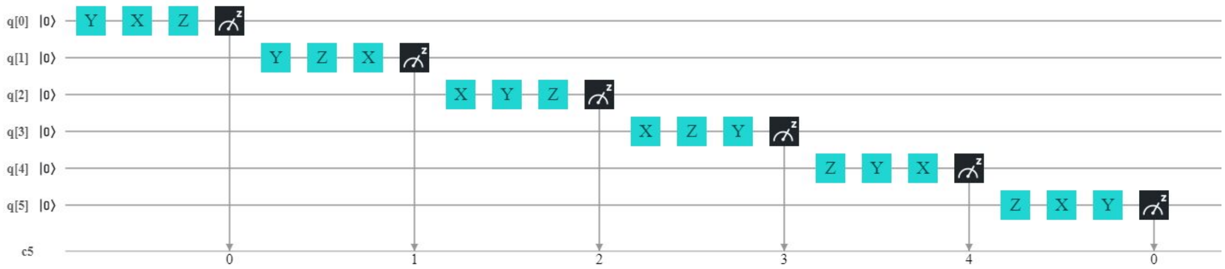


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

x q[0];
z q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
x q[2];
measure q[2] -> c[2];
measure q[3] -> c[3];
z q[4];
x q[4];
measure q[4] -> c[4];
```

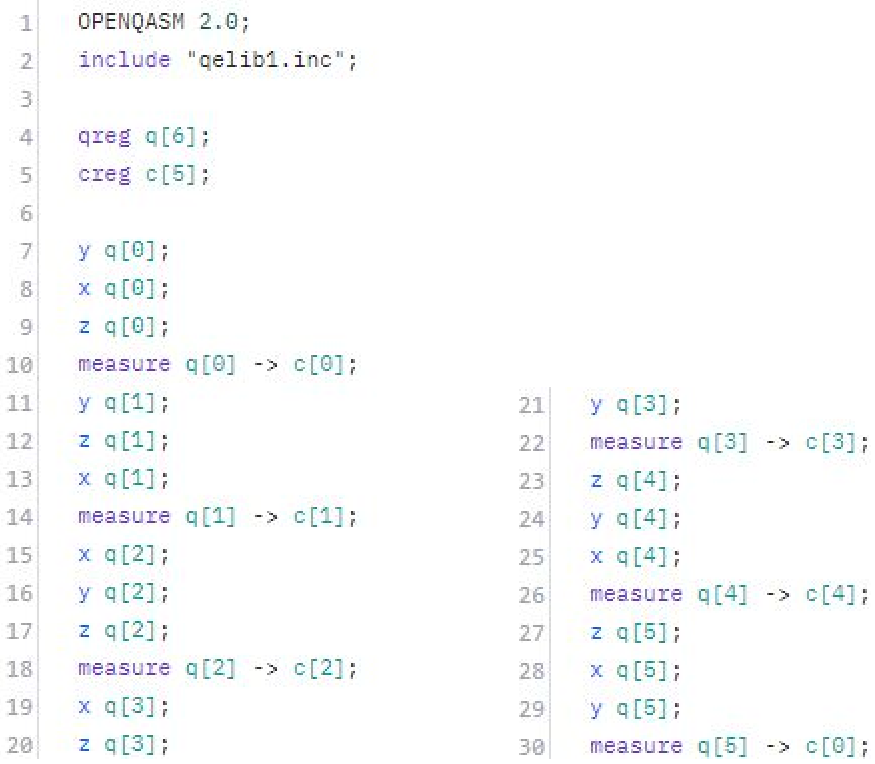
Output- Only qubits 0 and 5 will change their states since the Pauli Z gate will output |-1> when the input is |1>; however, the negative sign is for the direction and will not be denoted when outputting the state.



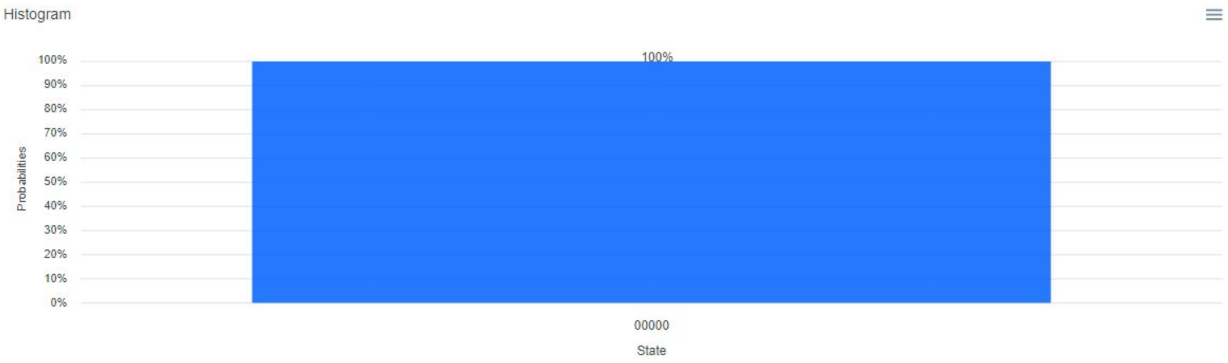
Circuit 6 -Combination of Pauli X, Y, and Z gates on the 6 qubits in the circuit



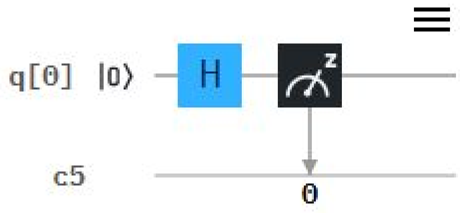
Code-


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[6];
creg c[5];

y q[0];
x q[0];
z q[0];
measure q[0] -> c[0];
y q[1];
z q[1];
x q[1];
measure q[1] -> c[1];
x q[2];
y q[2];
z q[2];
measure q[2] -> c[2];
x q[3];
z q[3];
y q[3];
measure q[3] -> c[3];
z q[4];
y q[4];
x q[4];
measure q[4] -> c[4];
z q[5];
x q[5];
y q[5];
measure q[5] -> c[0];
```

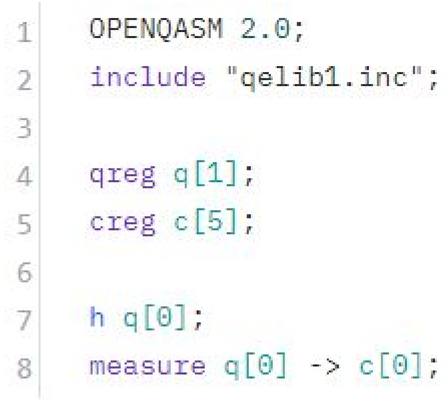
Output-



<u>Circuit 7</u> - Applying Hadamard gate on qubit 0 which will put the qubit in superposition

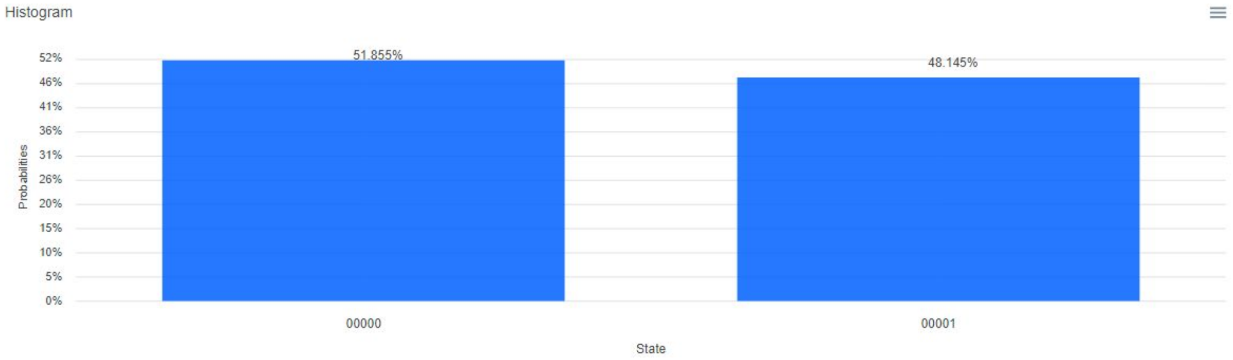


Code-

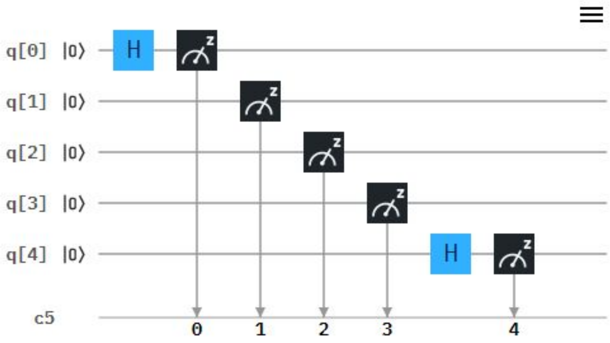


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

h q[0];
measure q[0] -> c[0];
```

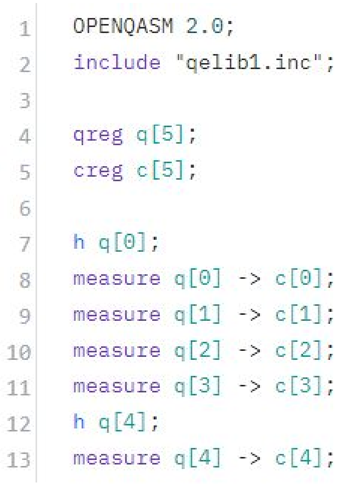
Output- The hadamard gate uses a very popular quantum physics principle called superposition, where the qubit's state will be in two states at once: both |1> and |0>. It will have 50% probability for both states and be in a more fluid, dynamic state.



<u>Circuit 8</u>-Applying the Hadamard gate on qubits 0 and 4, which puts the qubits in superposition ⇒ this is a famous quantum computing concept

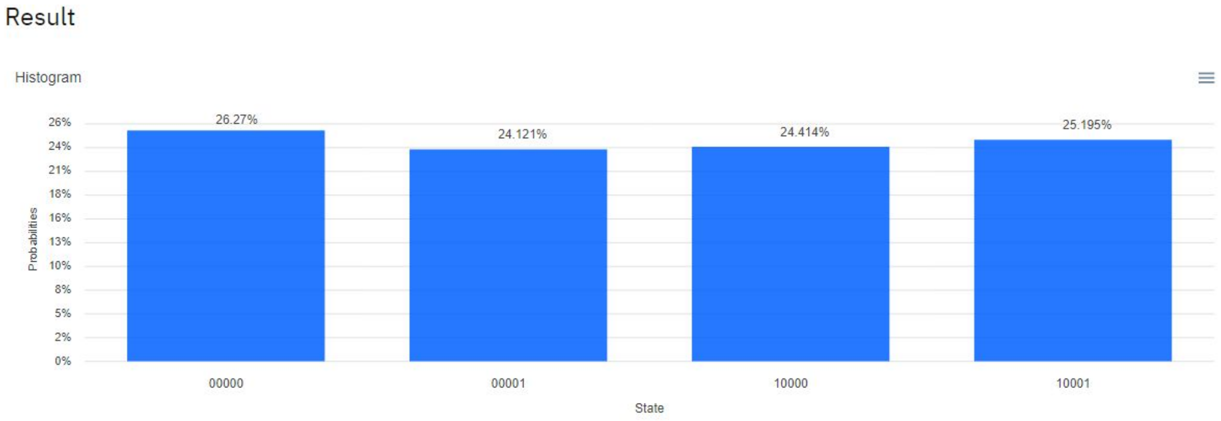


Code-

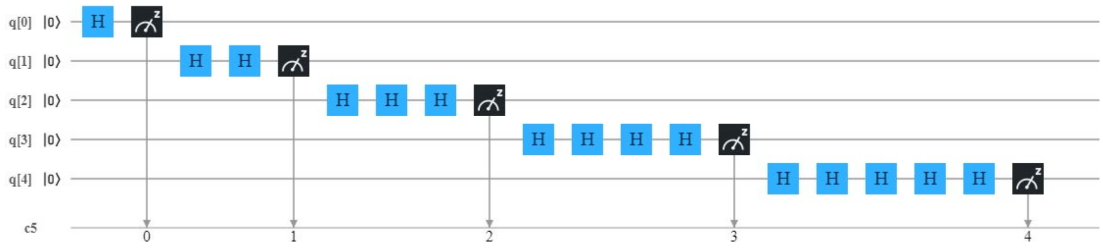


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

h q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
h q[4];
measure q[4] -> c[4];
```

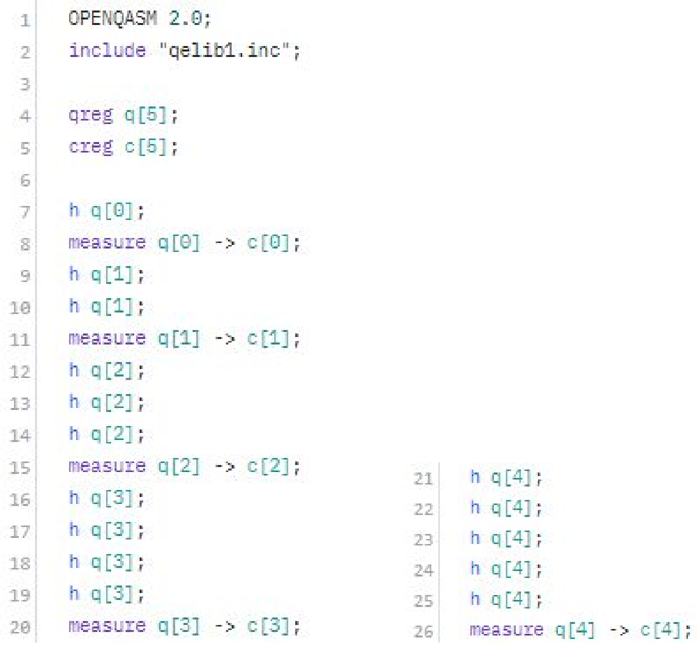
Output- Both qubits 0 and 4 will be in superposition. Thus, they both will have 50% probability of being in either state |0> and |1>. Thus, there are 4 possibilities, each with approximately 25%.



Circuit 9 - Applies Hadamard gate multiple times of different qubits



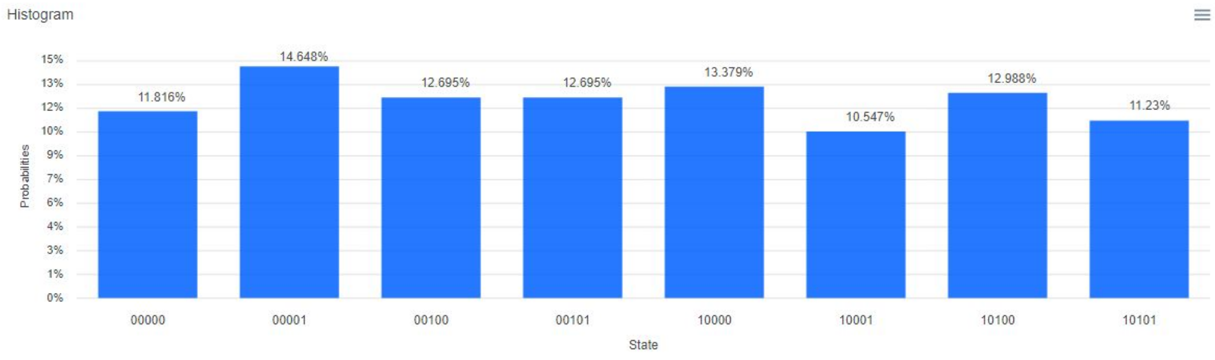
Code-



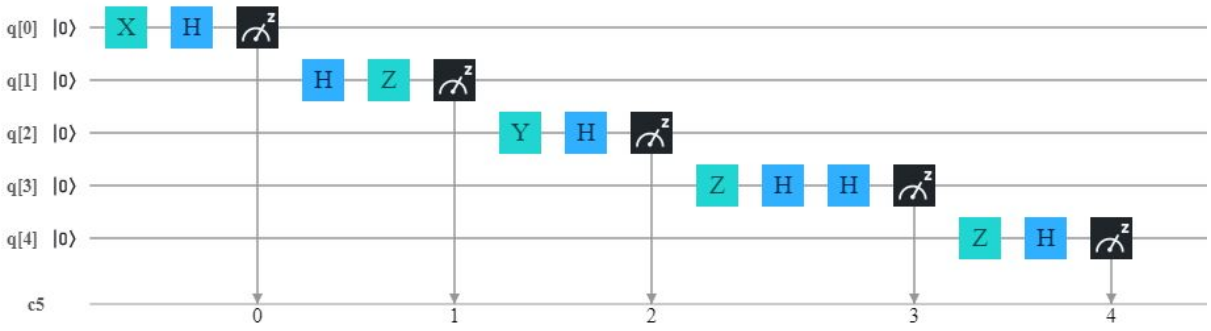
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

h q[0];
measure q[0] -> c[0];
h q[1];
h q[1];
measure q[1] -> c[1];
h q[2];
h q[2];
h q[2];
measure q[2] -> c[2];
h q[3];
h q[3];
h q[3];
h q[3];
measure q[3] -> c[3];
h q[4];
h q[4];
h q[4];
h q[4];
h q[4];
measure q[4] -> c[4];
```

Output- Similar to the previous example, each qubit which goes through the hadamard gate an odd number of times will be in superposition and if it goes through an even number of times it will be in its original state (by the hadamard gate property). Thus, qubits 0, 2, and 4 each have the possibility of being in either state |0> or |1> because of the superposition property and so, there are 8 cases each with approximately 12%.

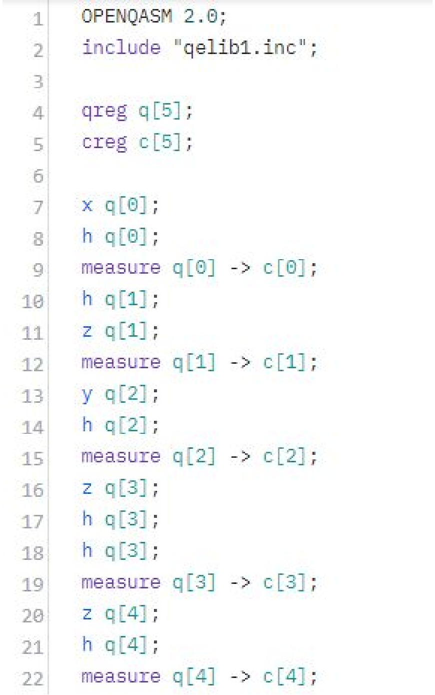
Result



<u>Circuit 10</u> - Applies the Hadamard and Pauli X, Y, Z gates in conjunction to 5 qubits

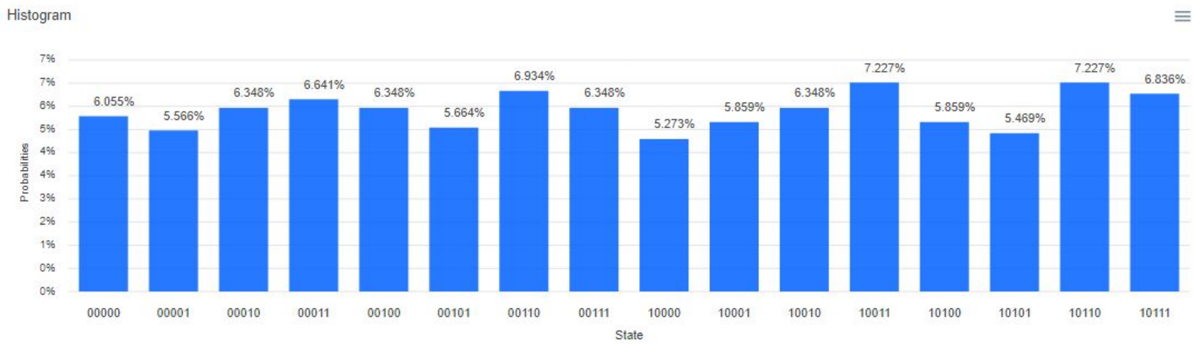


Code-

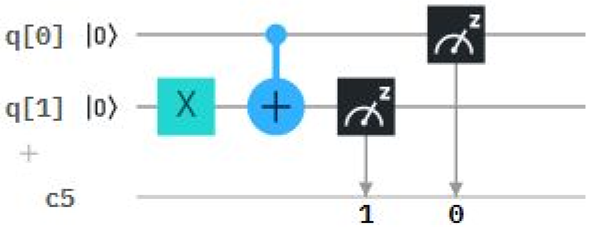


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

x q[0];
h q[0];
measure q[0] -> c[0];
h q[1];
z q[1];
measure q[1] -> c[1];
y q[2];
h q[2];
measure q[2] -> c[2];
z q[3];
h q[3];
h q[3];
measure q[3] -> c[3];
z q[4];
h q[4];
measure q[4] -> c[4];
```

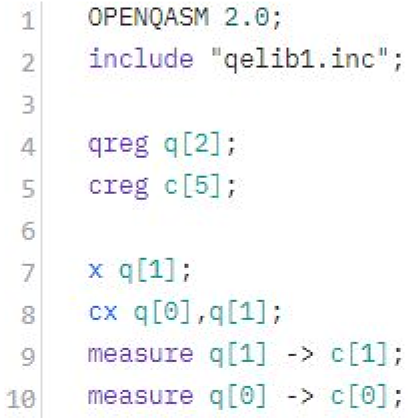
Output- In this case, there will be a total of 16 cases. Every qubit which has the hadamard gate applied an odd number of times will be in superposition.



Circuit 11 - Apply the Controlled NOT gate (CX) gate; Unlike the other gates explored in previous quantum circuits in this doc, this gate operates on two qubits (control and target qubits) ⇒ this gate performs the Pauli X gate on the target qubit ONLY if control bit is in $|1>$ state



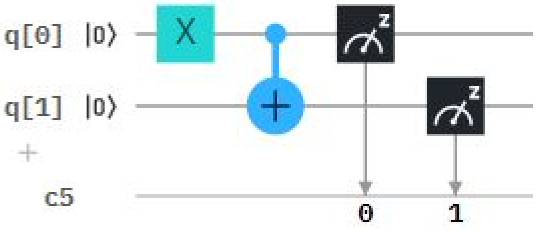
Code-



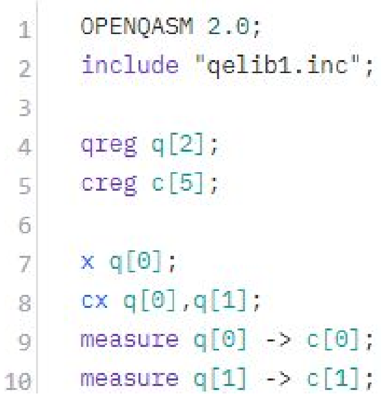
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[5];

x q[1];
cx q[0],q[1];
measure q[1] -> c[1];
measure q[0] -> c[0];
```

Output- We applied the X gate on qubit 1, transforming the state into |1>. However, we didn't transform the state of the control qubit (which is qubit 0 and it's still in state |0>), so the state for qubit 0 will stay the same and only qubit's 1 state will change to |1>.

<u>Circuit 12</u> - Apply the Pauli X gate on the control qubit instead of the target qubit with the CX gate ⇒ will lead to the CX gate altering the state of qubit 1

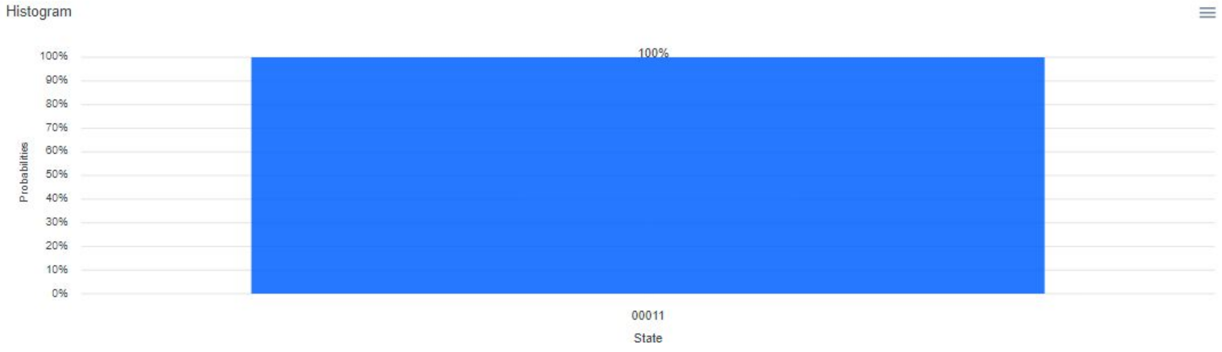


Code-

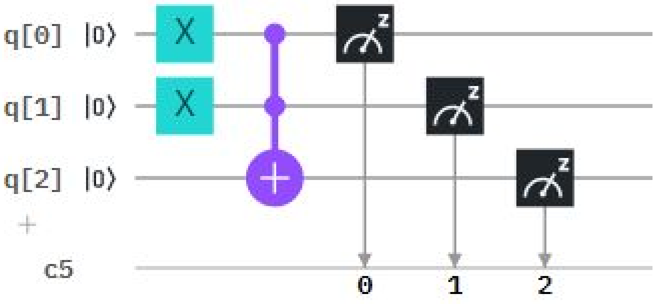


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[5];

x q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

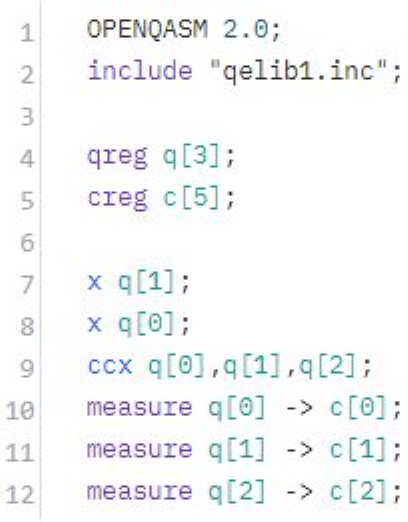
Output - In this case, both of the qubits 0 and 1 will change their state. Because qubit 0 goes through the Pauli X gate, it will transform to |1>, and the state for qubit 1 will also change because CX gate is applied.



<u>Circuit 13</u>- Apply the CCX gate; Unlike the other gates explored in previous quantum circuits in this doc, this gate operates on three qubits (control and target qubits) ⇒ this gate performs the Pauli X gate on the target qubit ONLY if BOTH control bits are in $|1>$ state



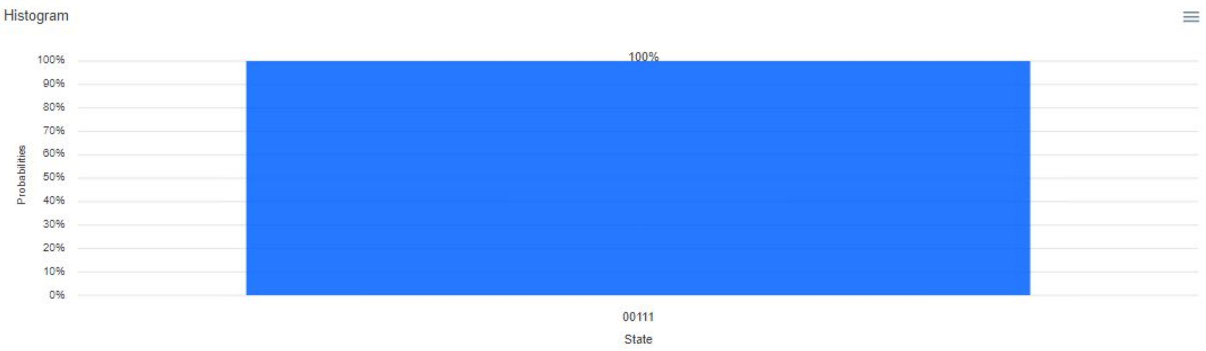
Code-



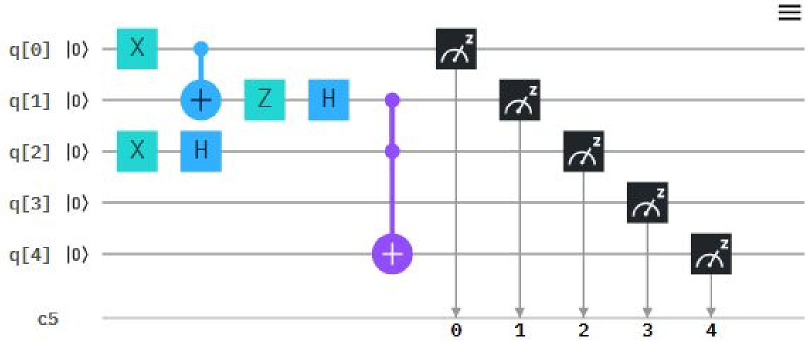
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[5];

x q[1];
x q[0];
ccx q[0],q[1],q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Output- The state for qubits 0, 1, and 2 will change. Qubits 0 and 1 will both have state |1> because they go through the Pauli X gate and qubit 2's state will go to |1> because of CCX.

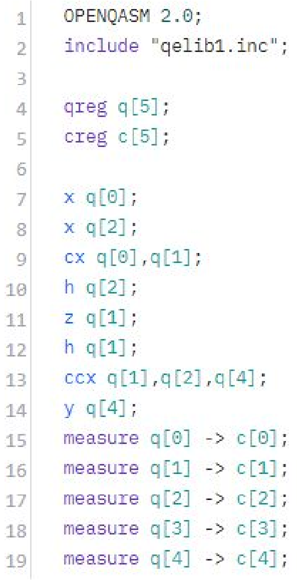
Result



Circuit 14- Applies the CX, CCX, Hadamard, and Pauli X, Y, and Z gates to the 5 qubits

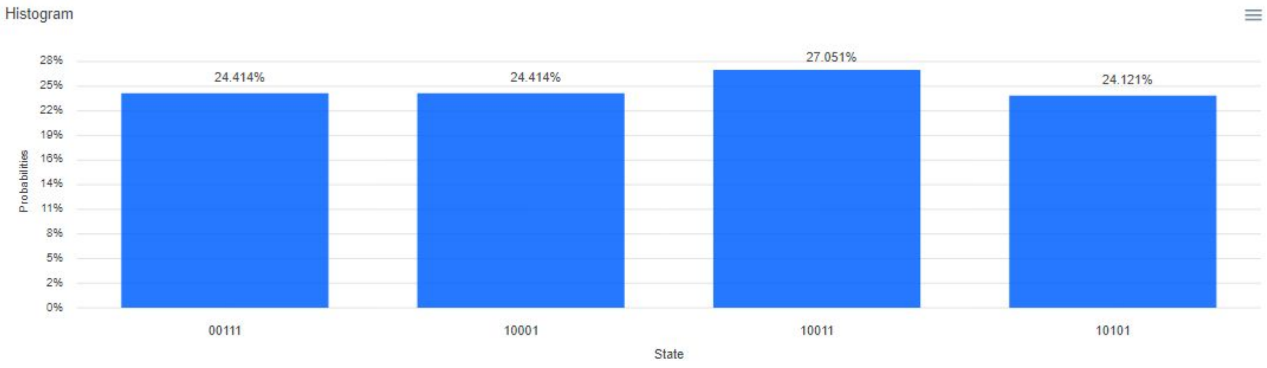


Code-

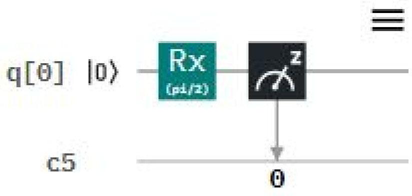


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

x q[0];
x q[2];
cx q[0],q[1];
h q[2];
z q[1];
h q[1];
ccx q[1],q[2],q[4];
y q[4];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

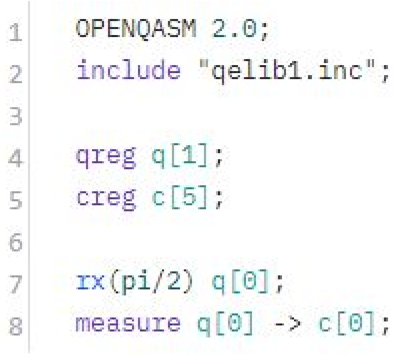
Output- Qubit 0's state will change to |1>. Qubit 1's state will be in superposition, where it could either be in state |0> or |1>. Similarly, qubit 2's state will also be in superposition. Qubit 4 will only be |1> when both qubit 1 and 2 are in state |1> because of the CCX gate.



<u>Circuit 15</u> - Applies the R gate (Rx) which will rotate the qubit along the x axis by the inputted angle (in radians) ⇒ changes the directionality through the rotation on the Bloch sphere

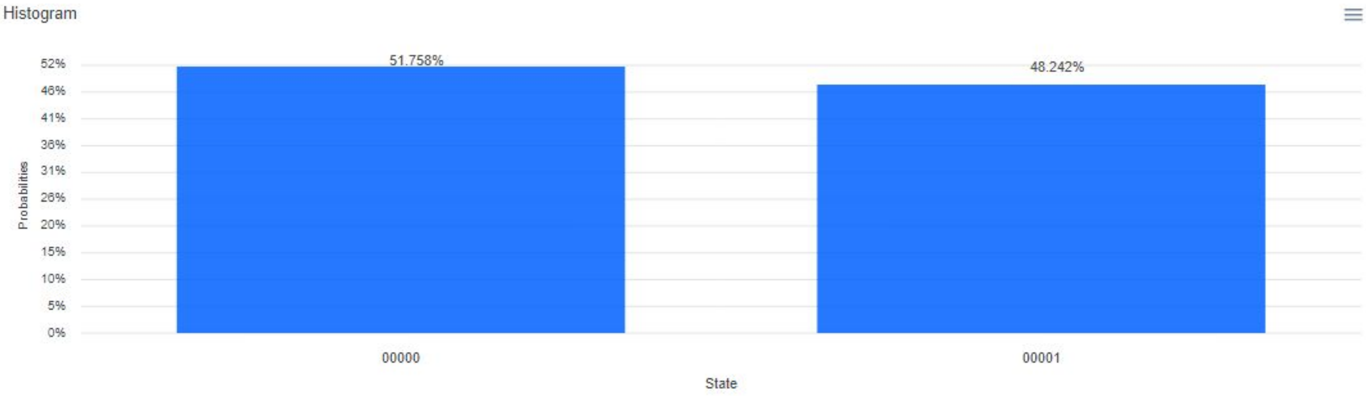


Code-

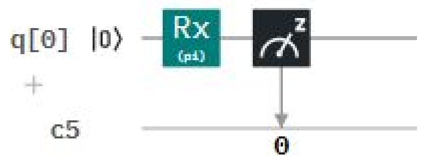


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

rx(pi/2) q[0];
measure q[0] -> c[0];
```

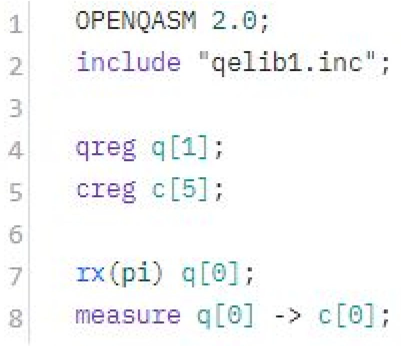
Output- Qubit 0 will rotate $\pi/2$ radians because of the Rx gate. Graphically, it will be in the middle of both states |0> and |1>, so there will be 50% probability of it being in either state.



<u>Circuit 16</u> - Applies the R gate (Rx) which will rotate the qubit along the x axis by π degrees (in radians) ⇒ changes the directionality through the rotation on the Bloch sphere

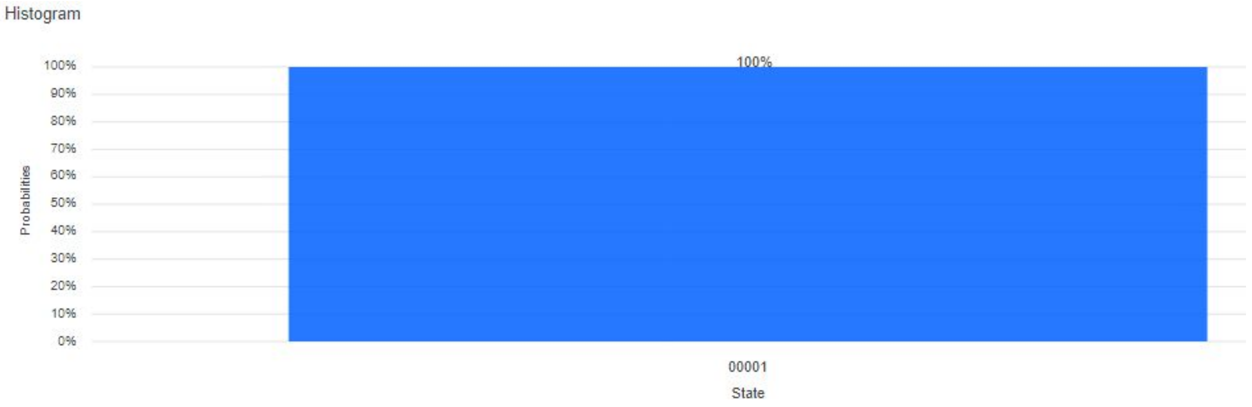


Code-

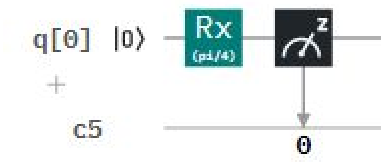


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

rx(pi) q[0];
measure q[0] -> c[0];
```

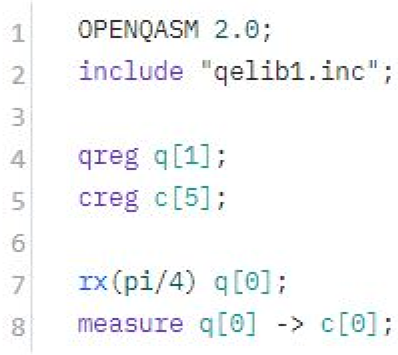
Output- Qubit 0 will rotate π radians because of the Rx gate. Graphically, it's state will change to |1> because of the rotation, so the probability is 100%.



<u>Circuit 17</u>- Applies the R gate (Rx) which will rotate the qubit along the x axis by $\pi/4$ degrees (in radians) ⇒ changes the directionality through the rotation on the Bloch sphere

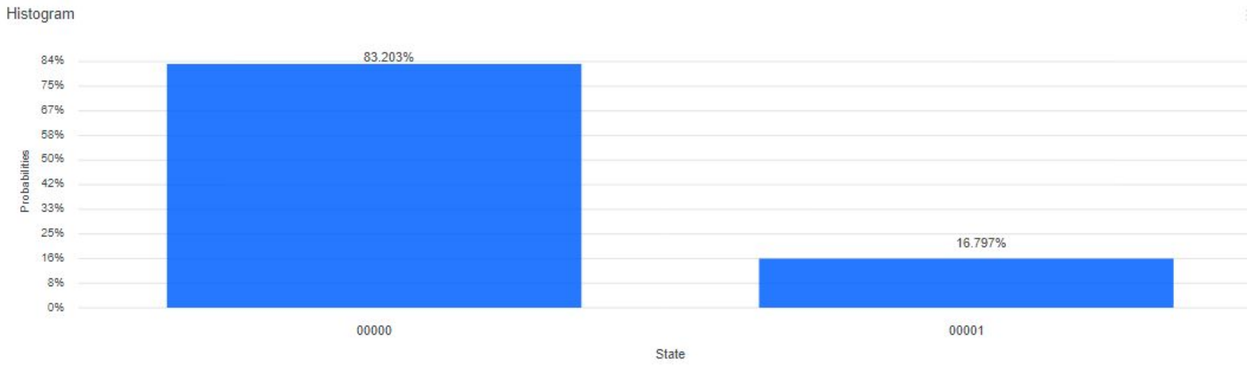


Code-

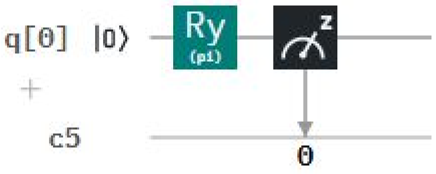


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

rx(pi/4) q[0];
measure q[0] -> c[0];
```

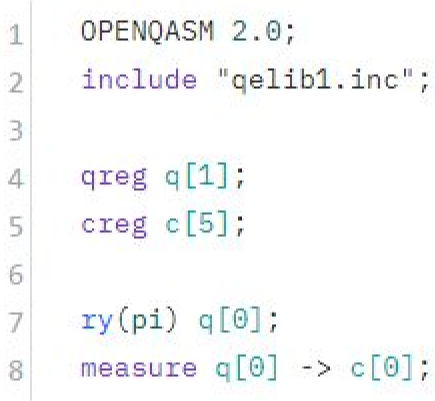
Output- Qubit 0 will rotate $\pi/4$ radians because of the Rx gate. Graphically, it's state will be closer to |0> because of the rotation, so the probability is closer to 100%, but not exactly 100%, since there is a slight rotation.



<u>Circuit 18</u>- Applies the R gate (Ry) which will rotate the qubit along the y axis by $\pi/2$ degrees (in radians) ⇒ changes the directionality of the qubit through the rotation on the Bloch sphere

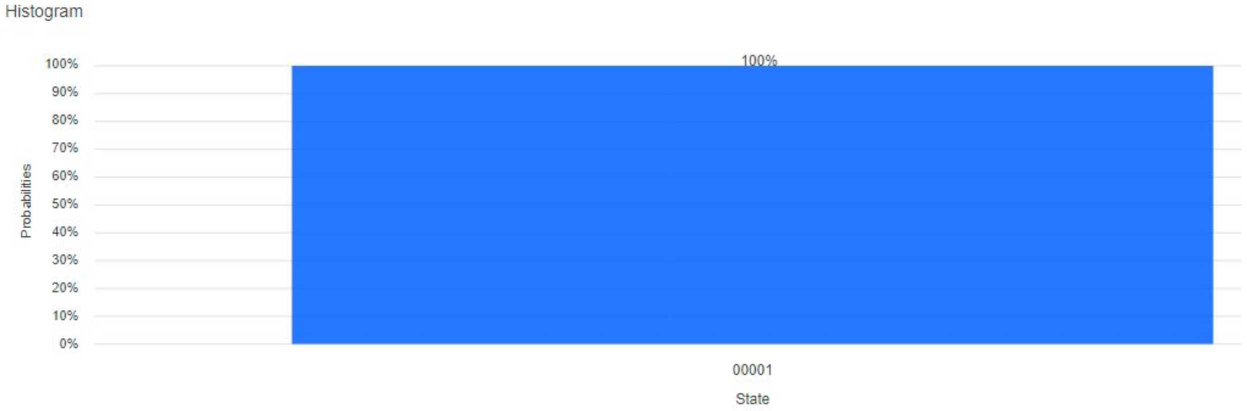


Code-

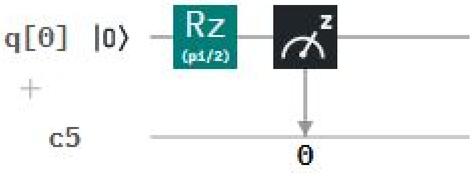


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

ry(pi) q[0];
measure q[0] -> c[0];
```

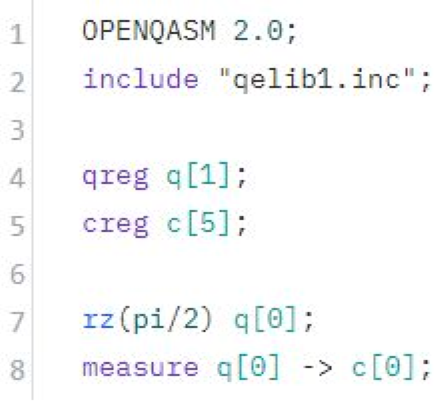
Output- Qubit 0 will rotate π radians because of the Ry gate. Graphically, it's state will be |1> because of the rotation, so the probability is 100%.



<u>Circuit 19</u>- Applies the R gate (Rz) which will rotate the qubit along the z axis by $\pi/2$ degrees (in radians) ⇒ changes the directionality of the qubit through the rotation on the Bloch sphere

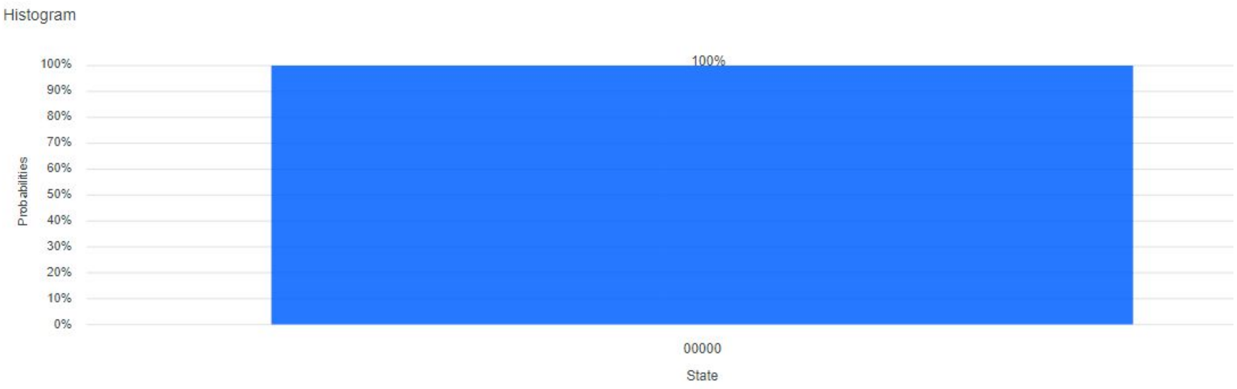


Code-

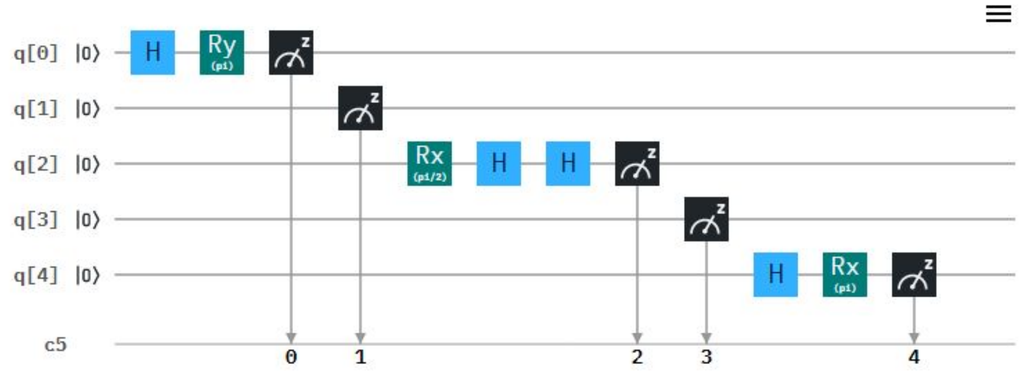


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[5];

rz(pi/2) q[0];
measure q[0] -> c[0];
```

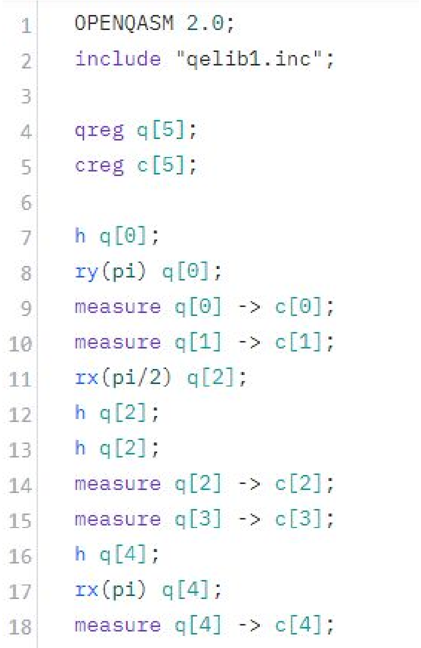
Output- Qubit 0 will rotate $\pi/2$ radians because of the Rz gate. Graphically, it's state will be |0> because when it's rotated about the z-axis, it's distance with respect to |0> and |1> states won't change.



Circuit 20 - Combination of R gates and Hadamard gates on the 5 qubits of the circuit

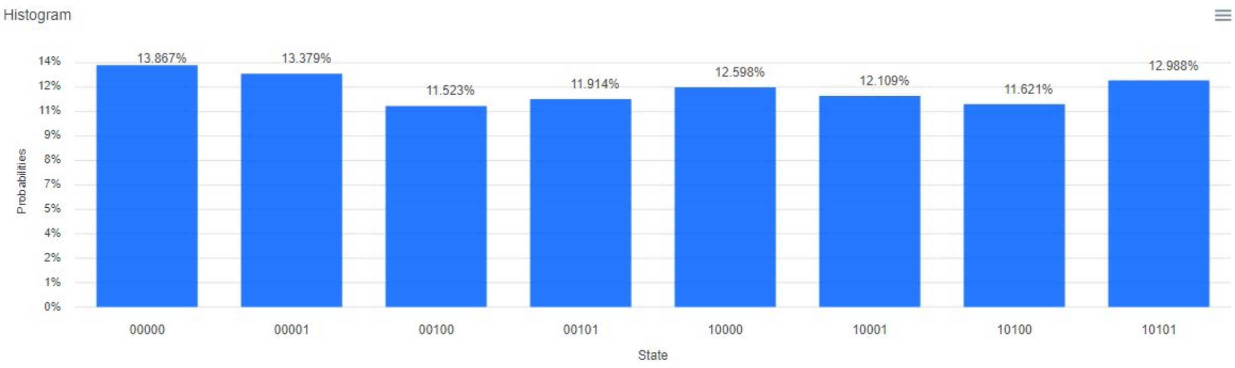


Code-

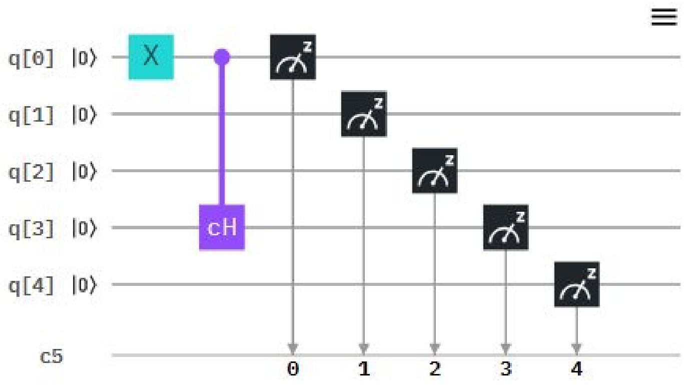


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

h q[0];
ry(pi) q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
rx(pi/2) q[2];
h q[2];
h q[2];
measure q[2] -> c[2];
measure q[3] -> c[3];
h q[4];
rx(pi) q[4];
measure q[4] -> c[4];
```

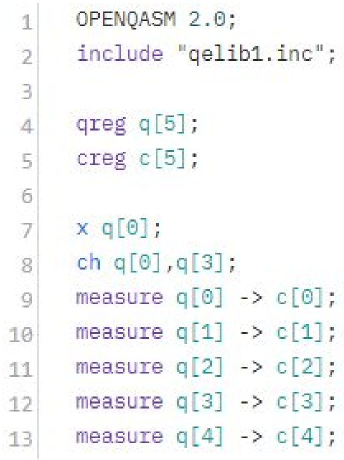
Output- Qubit’s 0 and 4 will be in superposition with 50% probability of being in either state |0> or |1> because of Hadamard. Qubit 1 will be rotated about the x-axis $\pi/2$ radians, which will result in it being midway between both states, so it will have 50% probability for each state. Thus, there are a total of 8 possibilities with approx same probabilities.



Circuit 21 - Applying the controlled hadamard gate, which is similar to the controlled XNot gate. Given two qubits (control and target qubits), the hadamard operation is performed on the target qubit whenever the control qubit is in the $|1>$

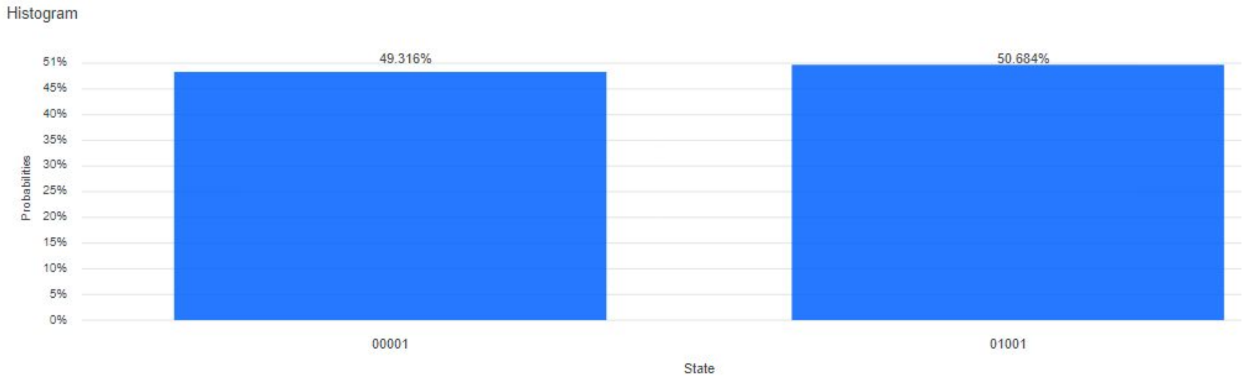


Code-

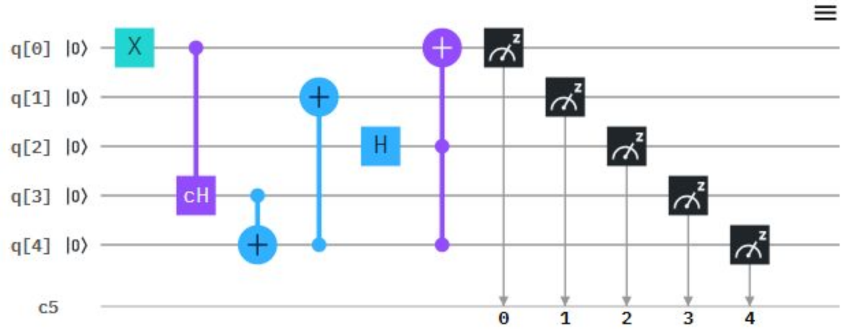


```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

x q[0];
ch q[0],q[3];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

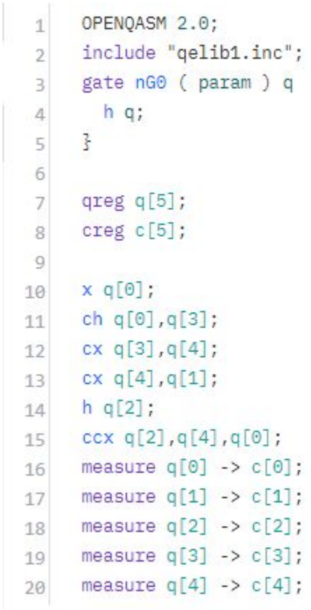
Output- Applying the x gate has no effect on the probabilities since it is simply flipping the state from |0> to |1>. The hadamard gate will then lead the probabilities resulting in 50% each.



Circuit 22 - Applying the controlled gates (controlled Hadamard, controlled-X, CCX, controlled-Y, controlled-Z, etc.) in conjunction with the 5 qubits in the circuit

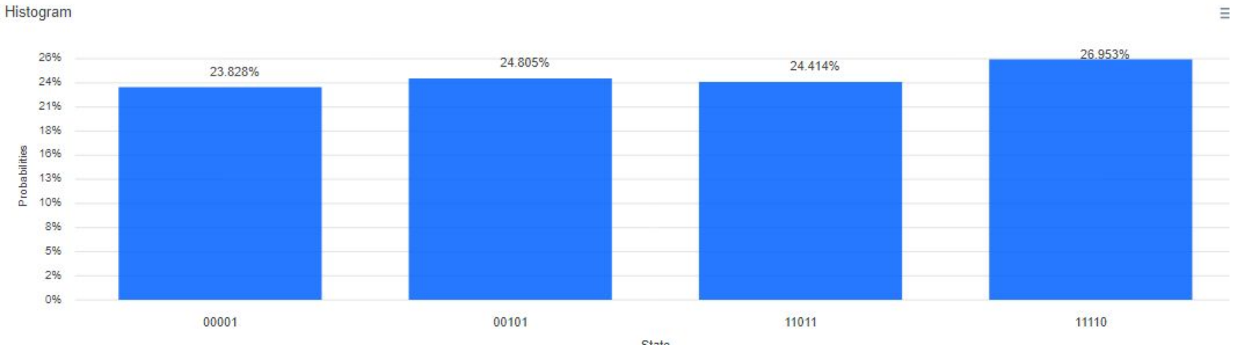


Code-

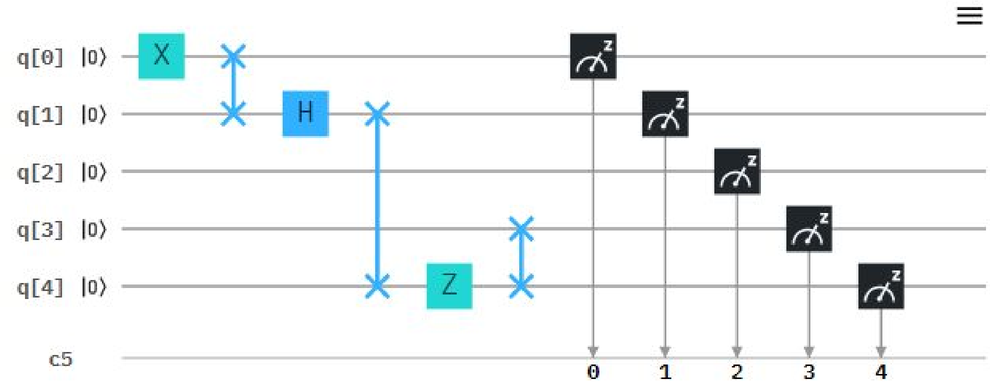


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[5];
creg c[5];

x q[0];
ch q[0],q[3];
cx q[3],q[4];
cx q[4],q[1];
h q[2];
ccx q[2],q[4],q[0];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

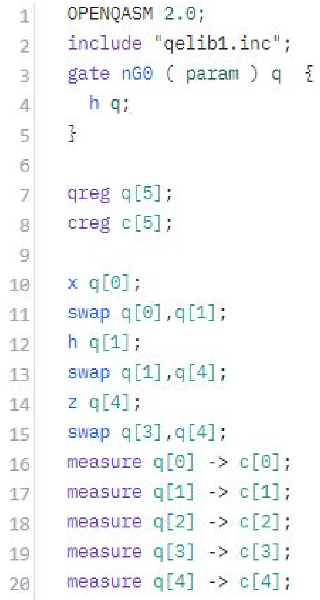
Output- Because of the cH gate, qubit 3 will be in superposition. Qubit 4 will not always be 1 because its control qubits (qubit 3) won't always be in the |1> state 50% of the time. Qubit 0's state also depends on qubit 2 and qubit 3 because of the CCX gate, so it too, will not always be in the |1> state. Thus, there are a total of 4 possibilities each with approx 25% prob.



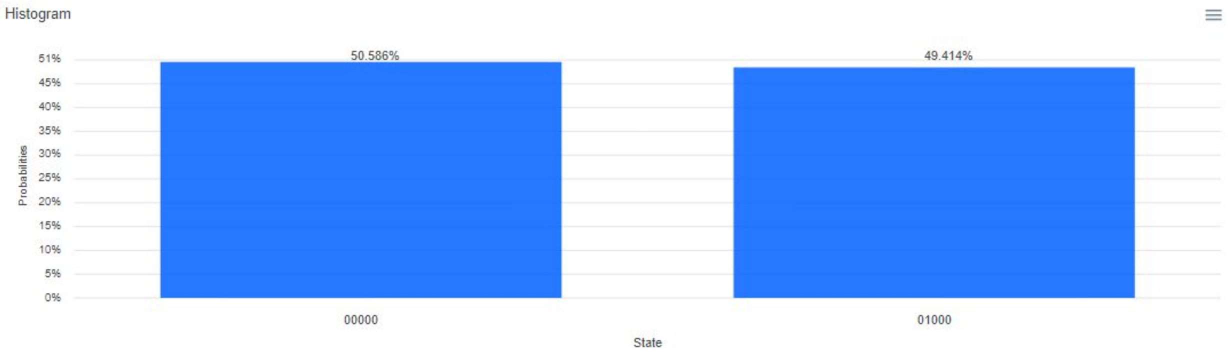
<u>Circuit 23</u> - Applying the swap gate on two qubits in the quantum circuit



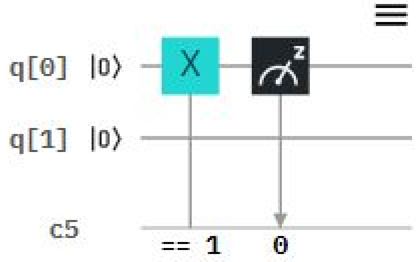
Code-


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[5];
creg c[5];

x q[0];
swap q[0],q[1];
h q[1];
swap q[1],q[4];
z q[4];
swap q[3],q[4];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

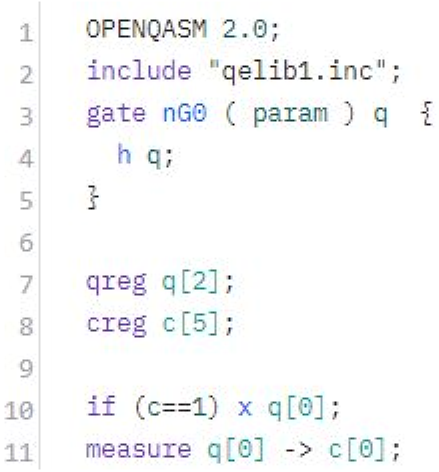
Output- Once qubit 0 changes its state to |1>, it will be swapped with qubit 1. Then, qubit 0 will always have state |0>. After swapping again twice, in the end, qubit 3 is in superposition.



<u>Circuit 24</u> - Using the “if” gate to apply conditions that will change the state of different qubits

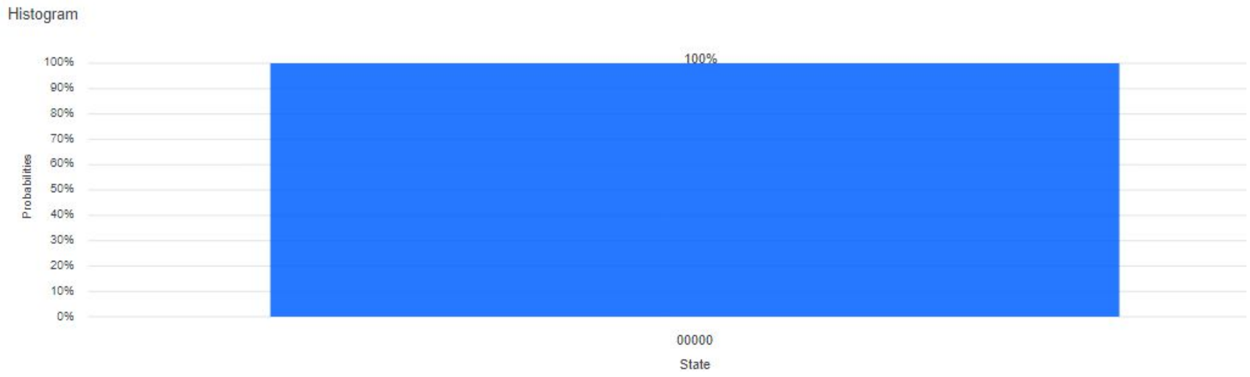


Code-

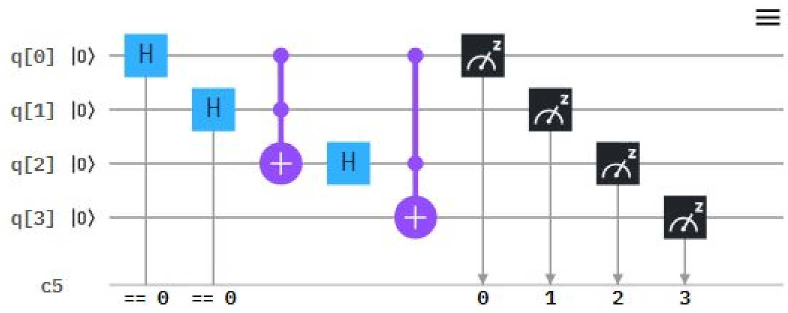


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[2];
creg c[5];

if (c==1) x q[0];
measure q[0] -> c[0];
```

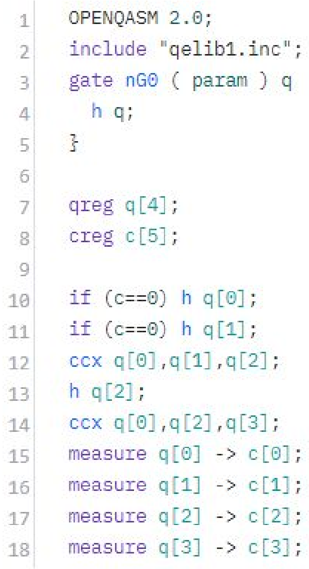
Output- The state of qubit 0 won’t change because it’s state is |0> to begin with, so Pauli X gate won’t be applied.



Circuit 25 - Using the “if” gate to apply conditions that will change the state of different qubits (more complex circuit than Circuit 24) along with hadamard and controlled gates

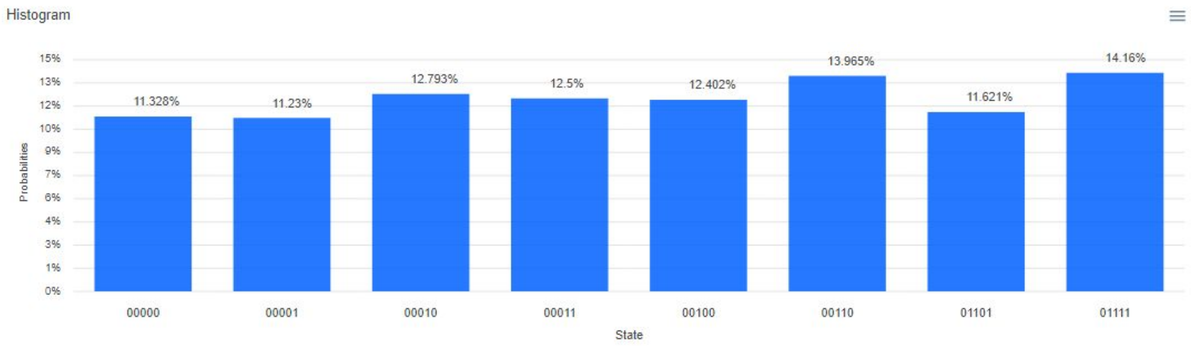


Code-

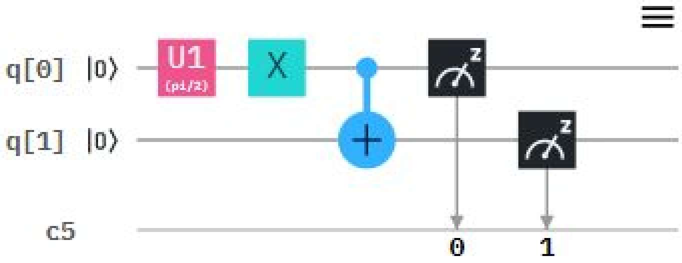


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[4];
creg c[5];

if (c==0) h q[0];
if (c==0) h q[1];
ccx q[0],q[1],q[2];
h q[2];
ccx q[0],q[2],q[3];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
```

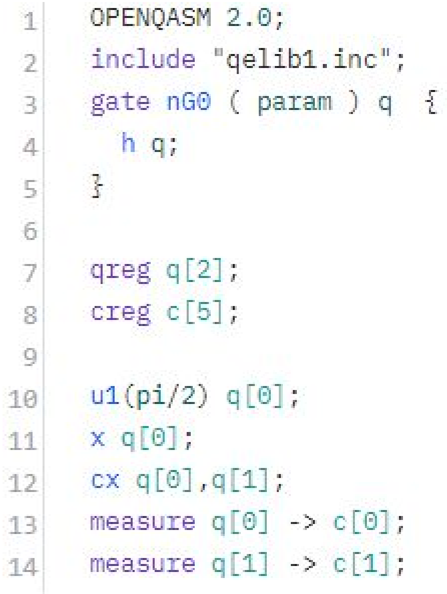
Output- All three qubits - qubits 0, 1, and 2 will be in superposition because of the hadamard, conditional, and CCX gates. Thus, each one has 2 possibilities amounting to a total of 8 possibilities with all being about equivalent probabilities.



<u>Circuit 26</u> - Applying the U1 gate, which performs the equivalent operation of the Rz gate, which rotates the qubit’s state along the z axis of the Bloch sphere by the defined angle

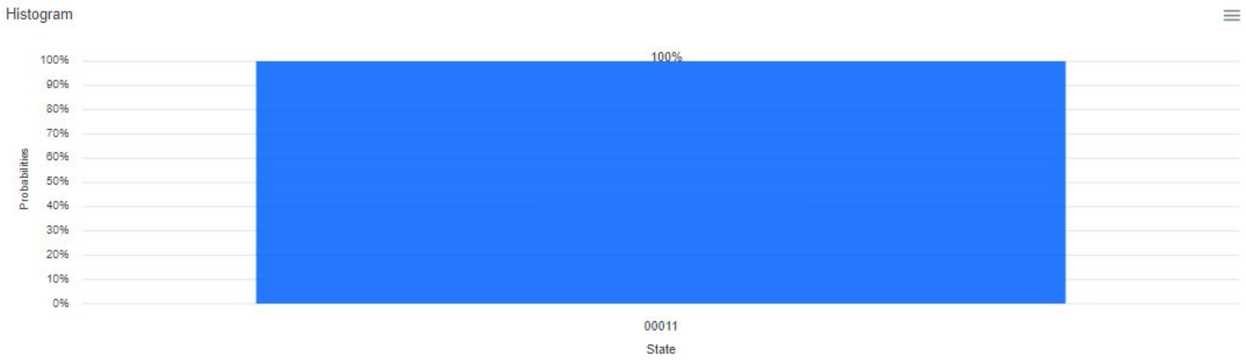


Code-

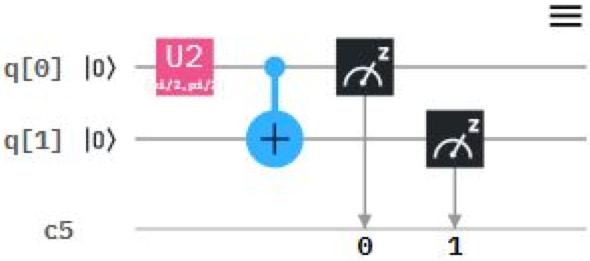


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[2];
creg c[5];

u1(pi/2) q[0];
x q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

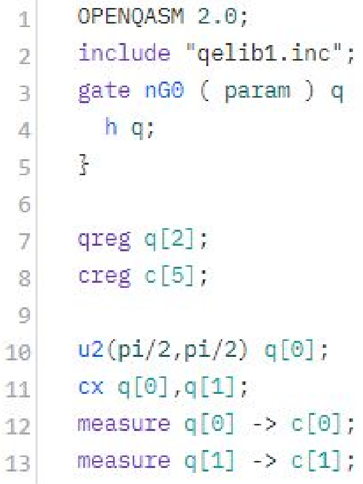
Output- The U1 gate itself won’t change the state of qubit 0, because when the qubit is rotated about the z-axis, it doesn’t get closer to either the |0> or |1> state. However, the Pauli X gate on qubit 0 leads to both qubits 0 and 1 being in the |1> state.



__Circuit 27__ - Applying the U2 gate, which has two parameters and each parameter controls the angle by which the qubit's state will be rotated about the x and z axes

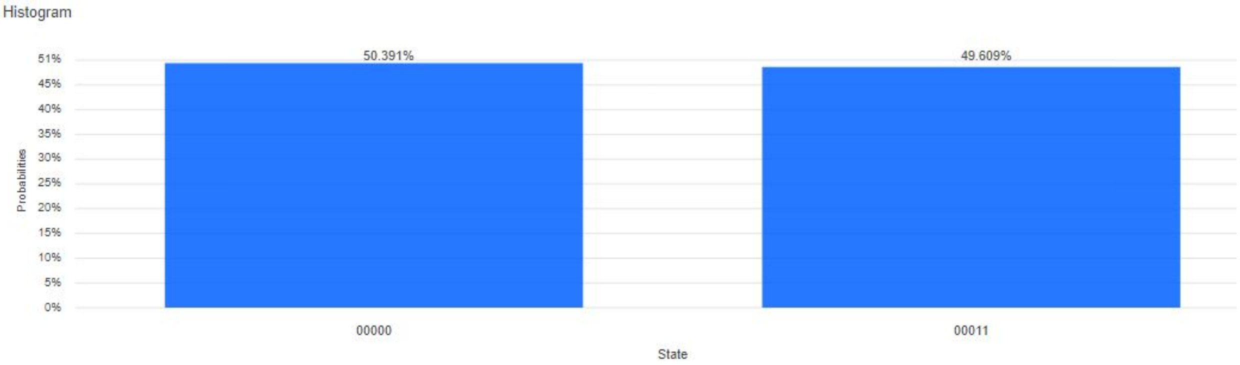


Code-

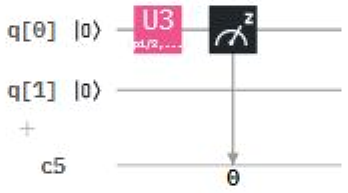


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[2];
creg c[5];

u2(pi/2,pi/2) q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

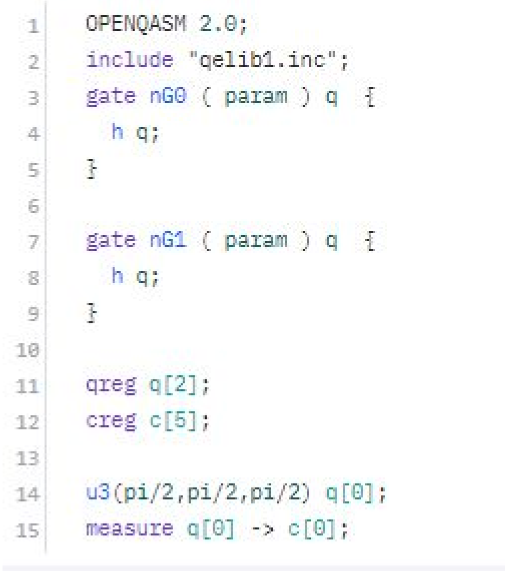
Output- Graphically, rotating the qubit about the x and z axes will lead the qubit's state to being midway between states |0> and |1>. Thus, there is a 50% probability of qubit 0 being in each state. Qubit 1 will only be in state |1> when qubit 0's state is also |1>.



Circuit 28 - Applying the U3 gate, which physically manipulates a qubits using three parameters and each parameter rotates the qubit’s state in the x, y, and z directions

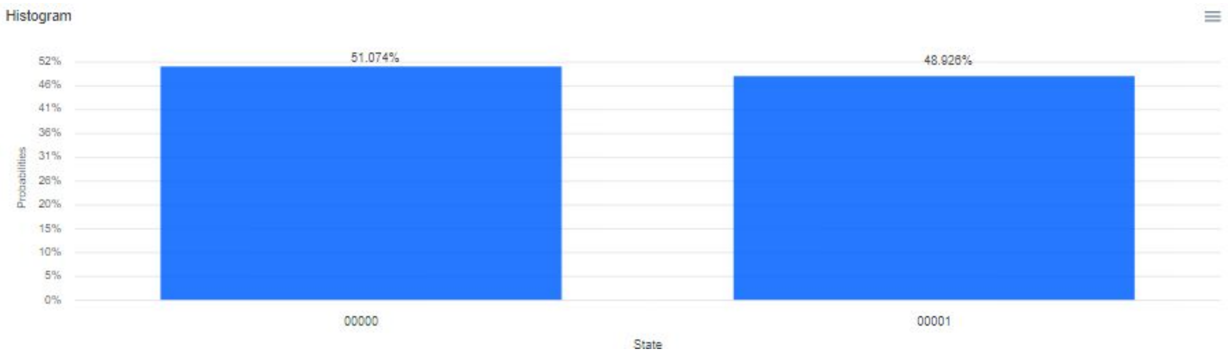


Code-

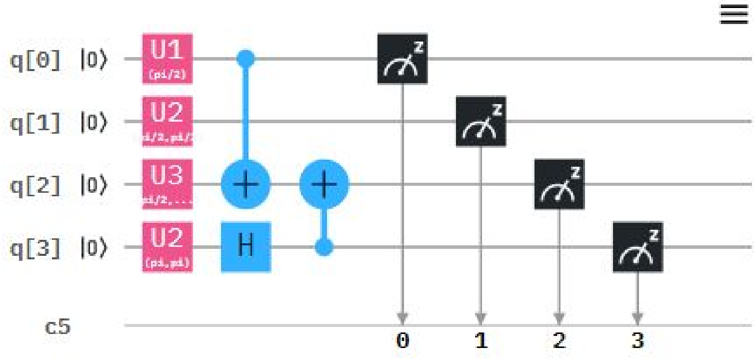


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

gate nG1 ( param ) q  {
  h q;
}

qreg q[2];
creg c[5];

u3(pi/2,pi/2,pi/2) q[0];
measure q[0] -> c[0];
```

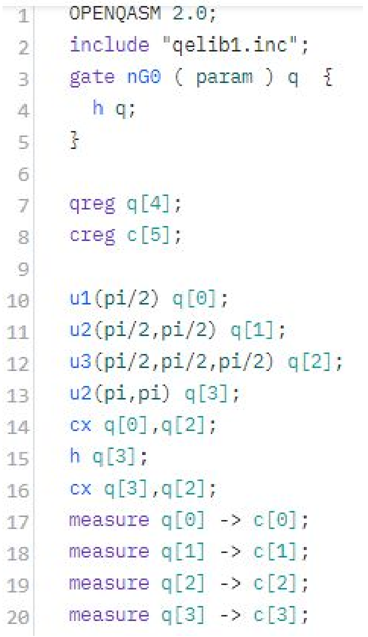
Output- Using the U3 gate will rotate the qubit $\pi/2$ radians with respect to all 3 axes: x, y, and z. This will result in the qubit being midway between the |0> and |1> states, resulting in approx 50% probability for each state.



<u>Circuit 29</u> - Applying the U1, U2, U3 gates in conjunction

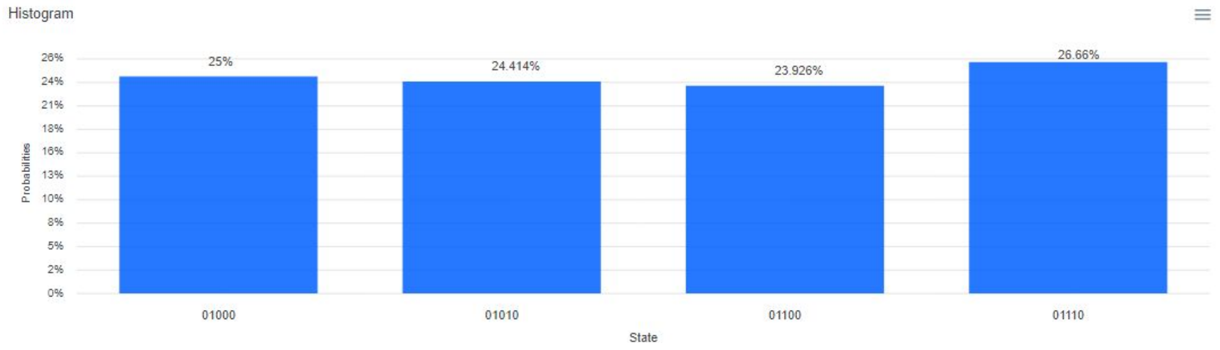


Code-

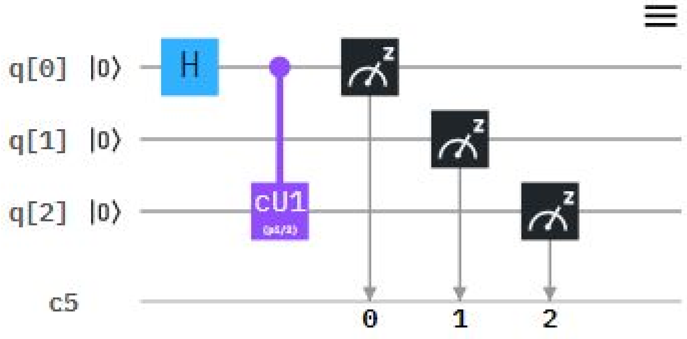


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[4];
creg c[5];

u1(pi/2) q[0];
u2(pi/2,pi/2) q[1];
u3(pi/2,pi/2,pi/2) q[2];
u2(pi,pi) q[3];
cx q[0],q[2];
h q[3];
cx q[3],q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
```

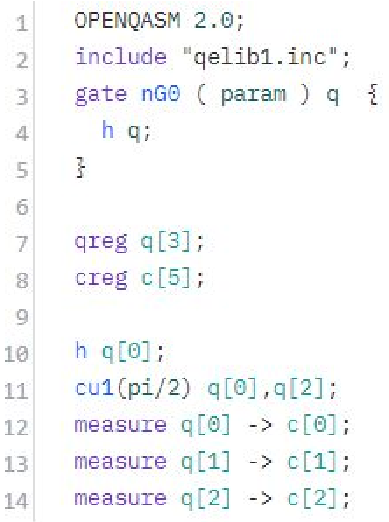
Output- Qubit 3 will be in superposition because of hadamard and U2 gate. Qubit 0 will always be in state |0> because rotating $\pi/2$ radians with respect to the z-axis won't change how close the qubit is to either state. Qubit 1 will also have 50% probability for each state because of U2 gate.



Circuit 30- Applying the controlled-U1 gate, where there is a control qubit and target qubit and the U1 operation is applied on the target qubit whenever the control qubit is in the $|1>$ state

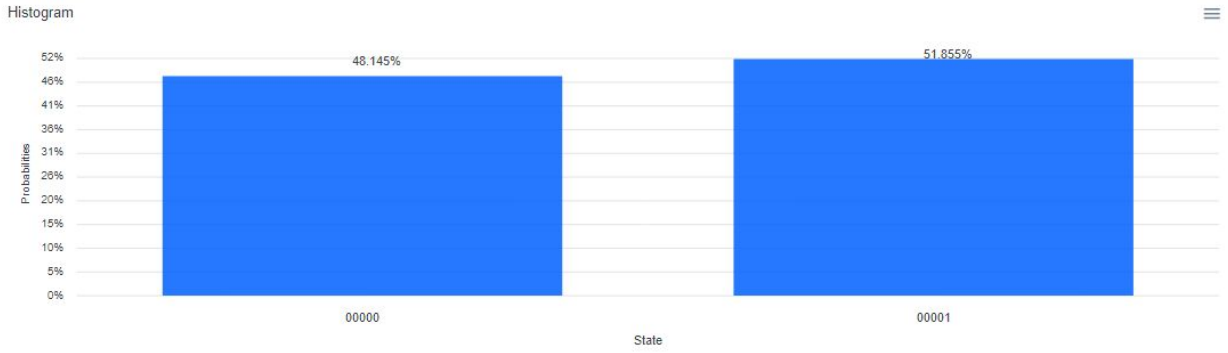


Code-

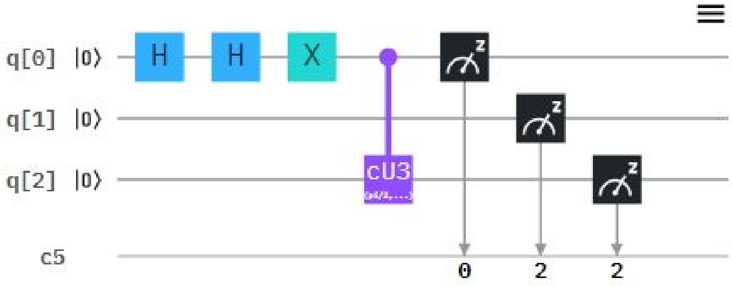


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

h q[0];
cu1(pi/2) q[0],q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

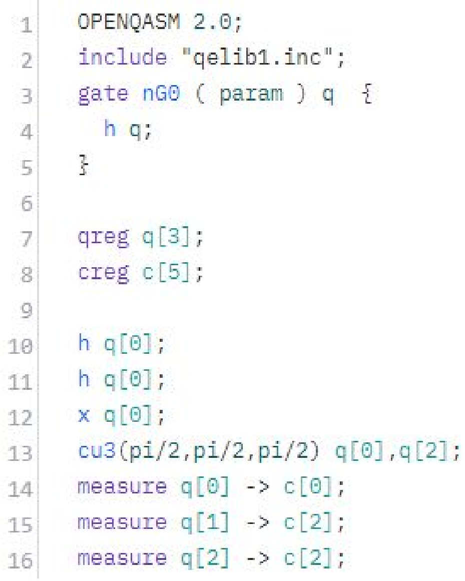
Output- Qubit 0 is in |1> state 50% of the time, so the hadamard gate will be applied to qubit 2 50% of the time. However, qubit 2's state will always remain in |0> state because rotating about z-axis won't change state.



<u>Circuit 31</u>- Applying the controlled-U3 gate, which performs the U3 operation on the target qubit when the control qubit is in state $|1>$



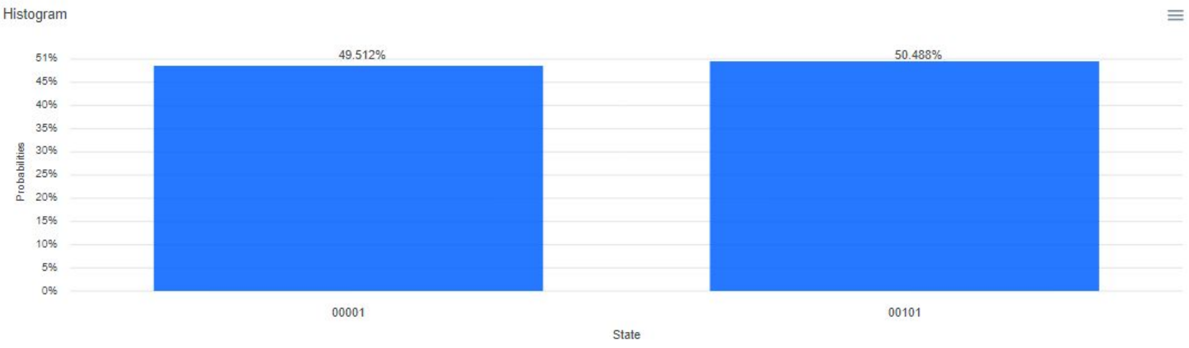
Code-



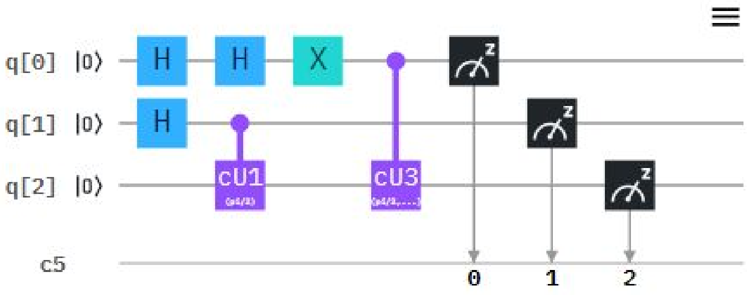
```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

h q[0];
h q[0];
x q[0];
cu3(pi/2,pi/2,pi/2) q[0],q[2];
measure q[0] -> c[0];
measure q[1] -> c[2];
measure q[2] -> c[2];
```

Output- Applying hadamard twice cancels out because of property. The Pauli X gate will change the state to |1>. Then, qubit 0 will always be in |1>. However, because of cU3, qubit 2's state will be midway |0> and |1> so there is a 50% probability for each of the states.

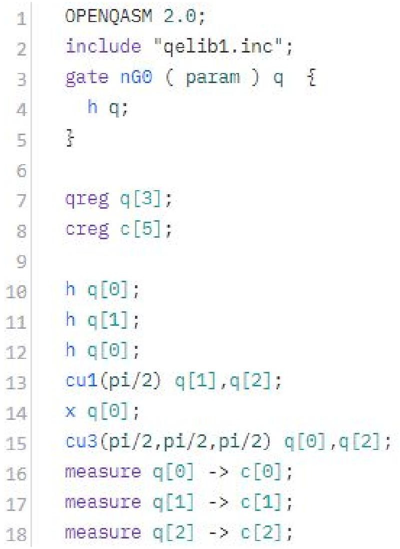
Result



<u>Circuit 32</u>- Applying the controlled U1 and U3 gates in conjunction with other quantum gates

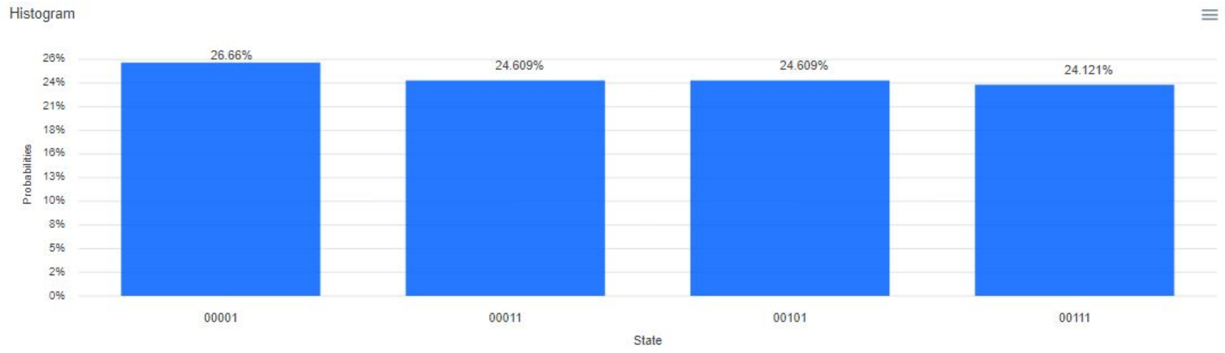


Code-

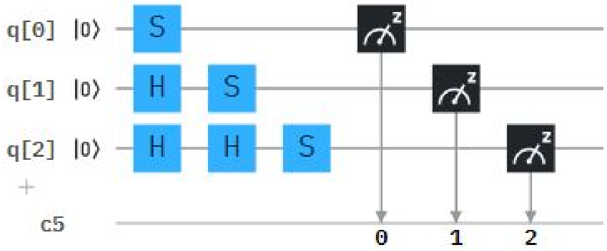


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

h q[0];
h q[1];
h q[0];
cu1(pi/2) q[1],q[2];
x q[0];
cu3(pi/2,pi/2,pi/2) q[0],q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

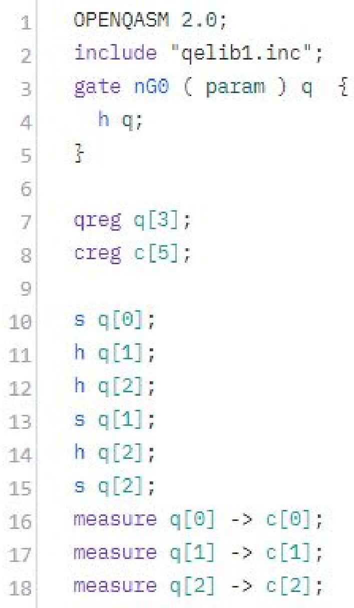
Output- Applying hadamard gate twice won't do anything. Qubit 0 will always be in |1> state because of Pauli X gate. Qubit 1 will always be in superposition because of hadamard gate. Qubit 2 will also have a 50% probability to be in either state |0> or |1> because of the controlled gates.



<u>Circuit 33</u>- Applying the S gate, which gives an output that is equivalent to the Rz gate and the parameter for the angle is $pi/2$. This gate's purpose is to move between the x base and y base.

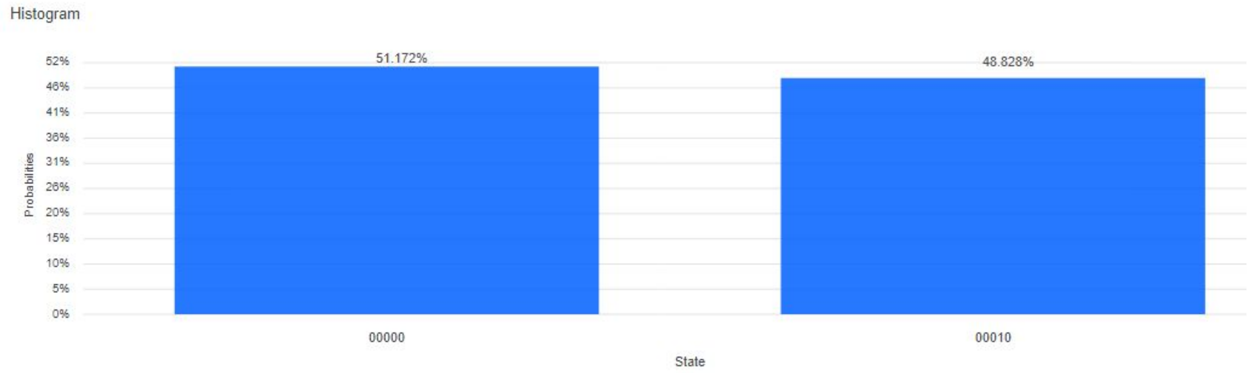


Code-

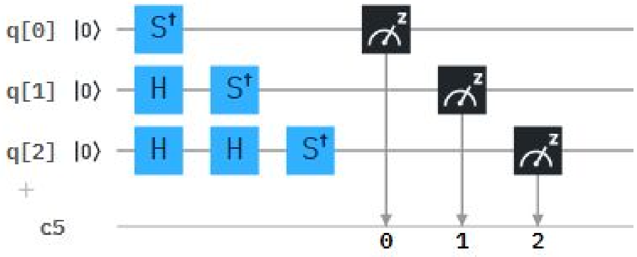


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

s q[0];
h q[1];
h q[2];
s q[1];
h q[2];
s q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

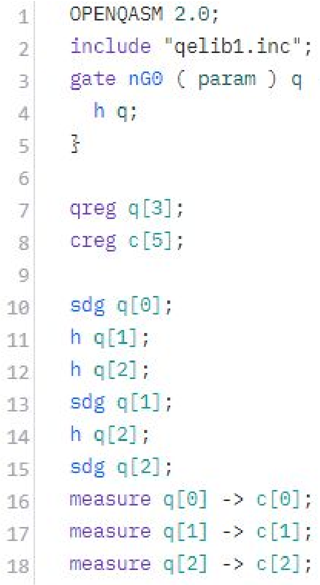
Output- Graphically, Qubit 0 will always be in |0> state because it won't get any closer to |0> when being rotated around z-axis. This is the same case for qubit 2. However, for qubit 1, it will be in superposition and there will be 50% probability that it could be in either state.



Circuit 34- Applying the Sd (inverse of the S) gate, which gives an output that is equivalent to the Rz gate and the parameter for the angle is $-pi/2$. This gate's purpose is to move between the x base and y base.

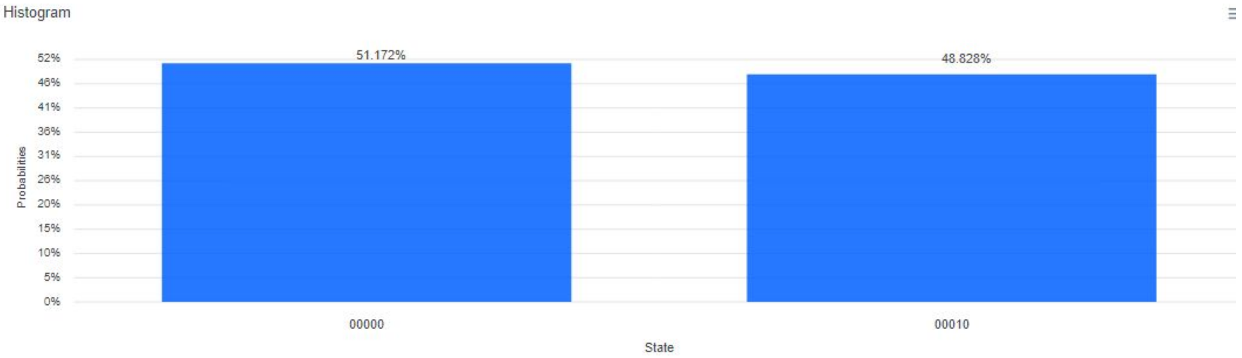


Code-

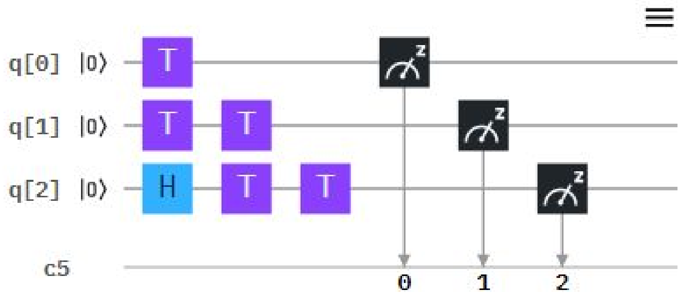


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

sdg q[0];
h q[1];
h q[2];
sdg q[1];
h q[2];
sdg q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

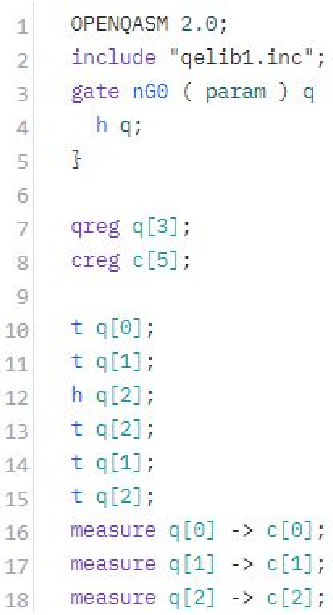
Output- Both qubits 0 and 2 will be in state |0> since rotation about the z-axis $-pi/2$ radians won't lead the qubit to being any closer to either state. Qubit 1, however, because of the hadamard gate will have 50% chance of being in either state |0> or |1>.



Circuit 35- Applying the T gate, which gives an output that is equivalent to the Rz gate and the parameter for the angle is $pi/4$.

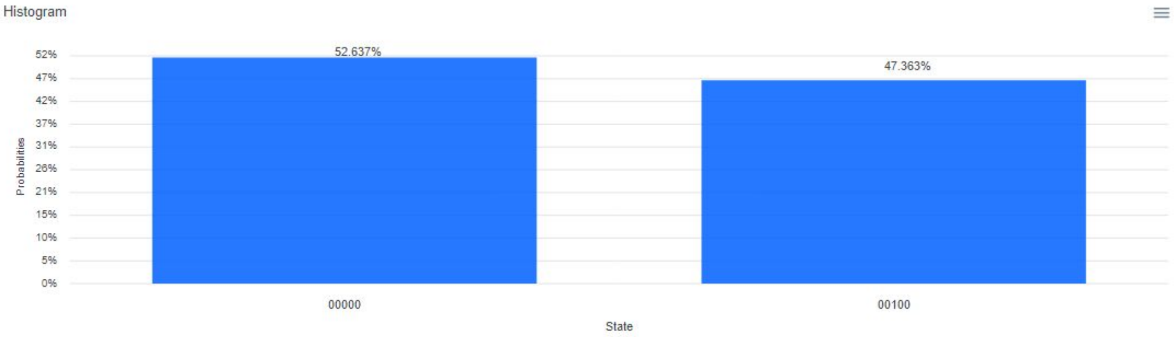


Code-

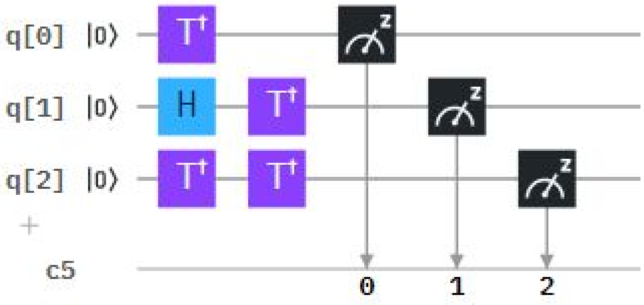


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

t q[0];
t q[1];
h q[2];
t q[2];
t q[1];
t q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

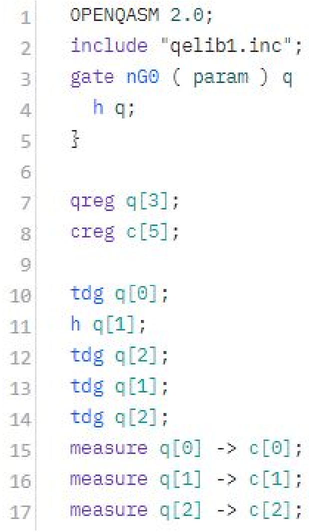
Output- Only qubit 2's state will change because of hadamard gate, so it will have 50% chance of being in either state |0> or |1>. For qubits 0 and 1, the rotation about the z axis will have no affect.



<u>Circuit 36</u> - Applying the Td gate (inverse of the T gate), which gives an output that is equivalent to the Rz gate and the parameter for the angle is $-pi/4$.

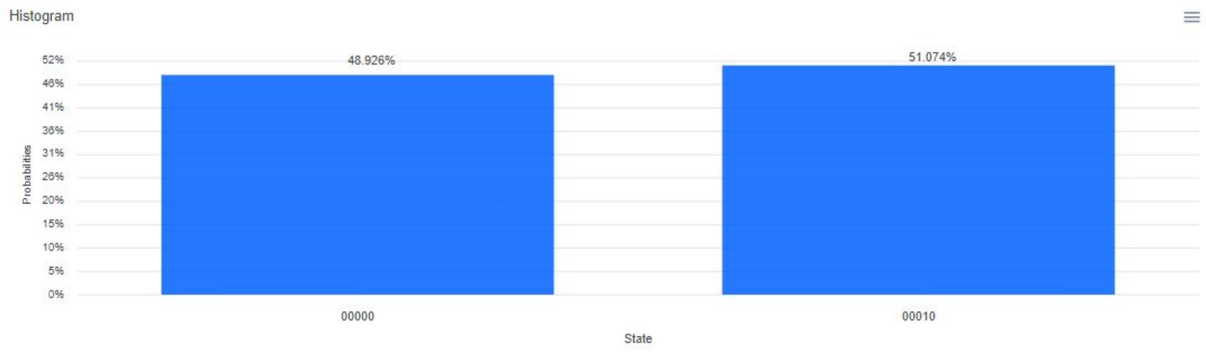


Code-

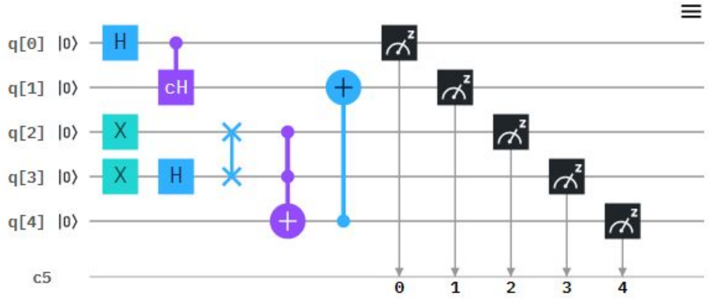


```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

qreg q[3];
creg c[5];

tdg q[0];
h q[1];
tdg q[2];
tdg q[1];
tdg q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

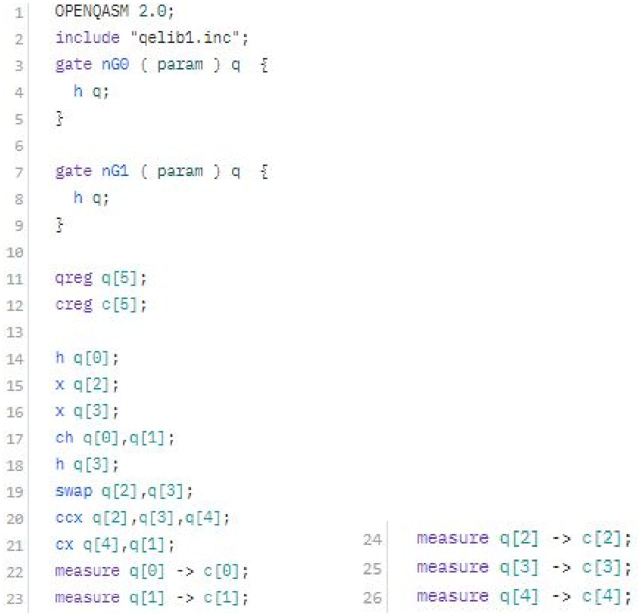
Output- Only qubit 1's state will change because of hadamard gate, so it will have 50% chance of being in either state |0> or |1>. For qubits 0 and 2, the rotation about the z axis will have no affect.



<u>Circuit 37</u> - Applying all of the controlled gates in conjunction

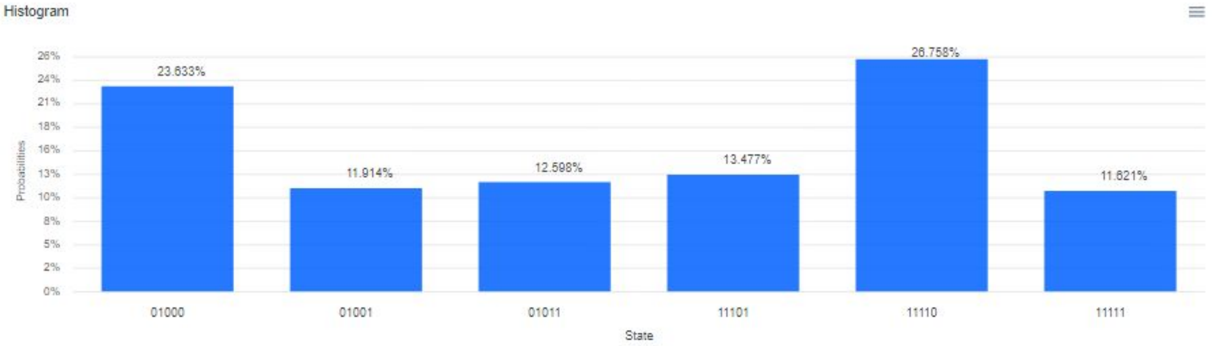


Code-



```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q  {
  h q;
}

gate nG1 ( param ) q  {
  h q;
}

qreg q[5];
creg c[5];

h q[0];
x q[2];
x q[3];
ch q[0],q[1];
h q[3];
swap q[2],q[3];
ccx q[2],q[3],q[4];
cx q[4],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

Output- Qubit 3’s state will always be |1> because of the Pauli X gate and swap gate and qubit 2’s state will be in superposition because of the hadamard and swap gates. Thus, qubit 4 will be in state |1> 50% of the time because of its dependency with qubits 2 and 3. Because of qubit 2, 3, and 4, qubit 1 will not always be in state 1.



Conclusion:

Through these circuits that I created and simulated using IBM Quantum Experience platform, I solidified my quantum computing concepts like superposition and entanglement and understood the functions of each of the gates along with getting the opportunity to get exposure to a real quantum computer at my fingertips which was very thrilling! At the same time, I was able to reinforce the linear algebra concepts I learned through Stanford University Level Online (ULO) Linear Algebra course this past summer. Apart from tinkering and making with this platform, I co-founded the platform QuantFlow, where my friend and I wrote a review paper surveying major breakthroughs in quantum computing realm along with creating circuits on IBM Q Experience. In the upcoming pages of this portfolio, I have attached my review paper and additional computer science programming projects I have worked on in the past!

Next Steps:

There is a very famous analogy posed by Microsoft CEO Satya Nadella that summarizes the power quantum computers have: When trying to find a way out of a corn maze, a classical computer will use a brute force approach, where it will continuously test all of the possible paths until it finds a successful one. However, a quantum computer will simultaneously investigate all of the paths at the same time. This is only possible because of the quantum physics properties that a quantum computer harnesses such as superposition, which enables the qubit to be in multiple states simultaneously, dramatically improving classical computers' efficiency. Because of the immense power quantum computers have, I envision that the classical computers we use daily will soon be replaced by quantum computers - there will be a quantum computer in every home. I am confident I can achieve this through collaboration with my peers and professors on next-generation cutting-edge multidisciplinary quantum computing research and become the "***next Google in quantum computing!***"