Comparing Quantum Computing Platforms

Siddh Agarwal[1], Gio Abou Jaoude[2], Avery Leider[2(✉)], and Charles C. Tappert[2]

[1] Hicksville High School, Hicksville, NY, USA
sba17@protonmail.com
[2] Seidenberg School of CSIS, Pace University, New York, USA
{ga97026n,al43110n}@pace.edu, cctappert@gmail.com

Abstract. This research compares and contrasts two commonly available quantum computing platforms available today to academic researchers: the IBM Q-Experience and the University of Maryland's IonQ. Hands-on testing utilized the implementation of a simple two qubit circuit and tested the Pauli X, Y, and Z single-qubit gates as well as the CNOT 2+ qubit gate and compared the results, as well as the user experience. The user experience and the interface must be straightforward to help the user's understanding when planning quantum computing training for new knowledge workers in this exciting new field. Additionally, we demonstrate how a quantum computer's results, when the output is read in the classical computer, loses some of its information, since the quantum computer is operating in more dimensions than the classical computer can interpret. This is demonstrated with the ZX and XZ gates which appear to give the same result; however, using the mathematics of matrix notation, the phase difference between the two answers is revealed in their vectors, which are 180° apart.

Keywords: IBM Q-experience · IonQ · Quantum computing pedagogy · Quantum information · Amazon braket services

1 Introduction

When educators learn a new technology, it is always joined with thoughts of how to teach it to the next learner. IBM's Q-Experience quantum computer is clearly in the lead for researchers to use in education, as the company was the first to develop a universal quantum computer in 2017, and in January 2019, IBM opened up their quantum computers to worldwide research for free on the Internet. IBM offers free tutorials and enough free time on its simulators and real quantum computers for an educator to include this today as a lab component when they teach quantum computing.

IBM's competitors are working to catch up with them and replicate their success. However, the design of their hardware is not the same as IBM's approach, and so their interface is different because it is specific to their company's

K. Arai (Ed.): FICC 2022, LNNS 438, pp. 423–441, 2022.
https://doi.org/10.1007/978-3-030-98012-2_32

hardware. This research study reports the experience we had running quantum computing circuits on the quantum computers of IBM and IonQ, a competitor to IBM, to compare them in order to help future researchers make informed decisions of which quantum computer to use.

The IBM Q-Experience is simple and easy to use. To obtain the free resources and tutorials, you simply need to register and obtain an IBM ID. The free resources are a certain amount of computing time that refreshes every 24 h, and all of the free time you want looking at the site. There is a link to the open source platform used for programming code, called QISKit (Quantum Information Science Kit). QISKit is written in Python, and utilizes Jupyter notebooks. Valuable visualizations are offered, so you can drag and drop gates into quantum circuits without any programming at all. If programming is preferred, Python can be used via QISKit, as well as OpenQASM, IBM's assembly language for quantum computing. Both of these options are available alongside the drag-and-drop circuit composer, so you can switch between them if you want to. The single qubit gates can be seen in a Q-sphere visualization that changes dynamically as you move the gates around. IBM has many different simulators and quantum computers, which are free to use within the time limit.

The IonQ is more complex to use. First, you must establish an account at Microsoft Azure or Amazon Web Services (AWS). We created a Braket account at AWS, which requires a credit card. Only a small amount of time, one hour is offered for free, on the TN1 and SV1 simulators, and after that, it bills your credit card. The local simulator appears to be free to use, but a small fee is charged to use Amazon's online Jupyter notebook service, SageMaker, as shown in Fig. 1.

▾ SageMaker		$0.14
▾ US East (N. Virginia)		$0.14
Amazon SageMaker CreateVolume-Gp2		$0.14
$0.14 per GB-Mo of Notebook Instance ML storage	1.004 GB-Mo	$0.14
Amazon SageMaker RunInstance		$0.00
$0.00 for Notebk:ml.t3.medium per hour under monthly free tier	149.327 Hrs	$0.00

Fig. 1. AWS charges for notebooks

The quantum programming is also done in Jupyter notebooks, and the Braket developer guide is mostly concerned with getting connected to the system. It assumes that you already understand quantum computing, and does not offer free tutorials. We found that there was significant time lost to running the Jupyter notebook on the real machine. This was because of billing and metering of the AWS system that allows you access to the IonQ computer. For the circuits we ran on the IonQ machine, we were charged $0.01 per shot and $0.30 per task, as shown in Fig. 2.

AWS Service Charges		$20.73
▾ Braket		$20.60
▾ US East (N. Virginia)		$20.60
Amazon Braket CompleteTask		$20.00
$0.01 per-shot for IonQ-IonQdevice in US East (N. Virginia) Region	2,000.000 Quantum-Shot	$20.00
Amazon Braket Task		$0.60
$0.30 per-task for IonQ-IonQdevice in US East (N. Virginia) Region	2.000 Quantum-Task	$0.60

Fig. 2. IonQ charges for running our circuit

The IonQ results are read in zeroes and ones; however, the most significant bit is not on the left-hand side, as in the IBM Q-Experience. Instead, the most significant bit is on the right-hand side of the results.

Amazon Web Services offers many different quantum computers using the Braket system, which can all be programmed the same way in Jupyter notebooks. In the code, only one value has to be changed to run it on a different machine. They do not include the IBM Q-Experience.

The quantum computing devices available now on AWS Braket Portal are the D-Wave (Advantage_system1.1, DW_2000Q_6), the Rigetti (Aspen-8, Aspen-9), the IonQ, and two Amazon Braket simulators (the TN1 tensor network and the SV1 state vector).

We also describe the theory of the circuit we are using as an example, and what mathematical operation each quantum gate in the circuit is doing. The theory is proven in mathematics, and then proven again when the results appear as expected after the quantum circuit is run. Quantum computers are capable of more than what is visible in the results, because there is a bottleneck in the information sent as output from the quantum computer into the classical computer so the output results can be read. The output must be only binary zeros and ones.

The quantum circuit we ran on each of the quantum computers is displayed in Fig. 5. It has two qubits, which become entangled. Then, one qubit is put through a gate (one of the simple one-qubit Pauli gates, X, Y or Z; "U" represents that the gate will be one or two of these Pauli gates that can be represented as a Hermitian matrix). Following this, we de-entangle the two qubits, measure both qubits, and report the results as they appear in the classical registers. We then replace "U" with "X" to test the circuit. Then we replace "U" with "Y" to test the circuit. Then we replace "U" with "Z" to test the circuit. This simple circuit allows us to test the different platforms and find out if they give us the expected results, as well as become aware of any additional observable information to compare the platforms.

1.1 Summary of Introduction

The rest of this paper is composed of the Literature Review section, which identifies the sources of information used in this research; the Basic Concepts section, which is a review of all quantum computing concepts needed to understand this paper; the Methodology, which shows the approach used to compare the two

quantum computing platforms; the Findings, which shows the results of using this methodology on the two systems under study; and our final section, Future Work.

2 Literature Review

This specific circuit that was selected to use for our Methodology is well known in quantum computing, and was used because of its familiarity. Many have tested a perceived error made by quantum researcher Michael Nielsen, who used the same quantum circuit in his YouTube video "Superdense Coding: How to send two bits using one qubit" [6], as part of his video series "Quantum Computing for the Determined" [7].

Studying Nielsen's video, there were several comments made by viewers that said that he made an error, in which he said "XZ" when he meant to say "ZX" when describing the gate combinations that were operating on the first qubit, qubit 0. The difference is shown in Fig. 3 and in Fig. 4. Nielsen acknowledged these comments and agreed that he had made the error.

However, in Lewis Westfall's "Superdense Coding Step by Step" [10], in which Nielsen's presentation of the circuit is exactly replicated and explained without adjusting for the error, there was no difference in the expected output from the two gate combination. As seen in Fig. 4, testing "XZ" and "ZX" both result in an answer of "11" and inclusion in the full circuit. This causes a question of whether use of the "XZ" and "ZX" gates give the same output in this scenario, and are thus interchangeable.

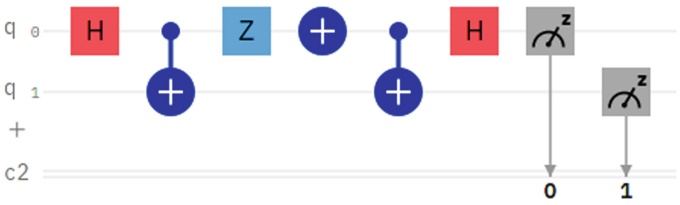

Fig. 3. Circuit using XZ gate on qubit 0

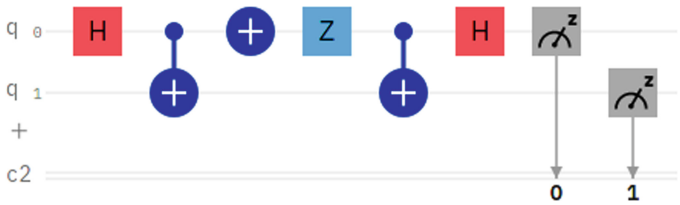

Fig. 4. Circuit using ZX gate on qubit 0

However, there is a difference, which in our Findings section of this paper can be observed in the IBM, but not the IonQ machine. The IBM Q-Experience quantum computer provides a visualization of the phase showing the difference as shown in Fig. 20 and 21. Also, one can see from the matrix multiplication that there is indeed a difference.

$$XZ => \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} => \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Using matrix multiplication, we can see that the order of the gates after transforming the ket keeps the direction, but changes the phase angle.

$$ZX => \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} => \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

In our quantum circuit, in place of Nielsen's X, Y, Z and ZX gates, we first place a U gate in that location as a placeholder. The U gate represents a gate that uses a unitary operation, and the reasoning for quantum circuit complexity of a unitary operation is closely related to the problem of finding minimal length paths in a particular curved geometry. This demonstrates the quantum computer's ability to calculate in more powerful ways than our current classical computers [3].

3 Basic Concepts

In quantum computing there are some basic concepts which are explained here. Matrix multiplication and linear algebra are used to understand and create circuits of gates. These circuits are for quantum bits, called "qubits" to be operated on to give results. Dirac bra-ket notation [2] is used to describe the vectors of zero ket and one ket. Kets can be explained in matrix form, and then multiplied by gates, also described in matrix form, to prove what the results will be of the gate operations on the quantum bits, or qubits. After doing the circuit in math, we then do it in the quantum computer to prove our results.

3.1 The Basis States of Ground State and Active State

The mathematical symbol for "zero ket," is used to express a vector used in quantum mechanics that represents a qubit in its *ground* state:

$$|0\rangle$$

Zero ket can be expressed as a 2 row by 1 column matrix:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The mathematical symbol for "one ket," is used to express a vector used in quantum mechanics that represents a qubit in its *active* state:

$$|1\rangle$$

One ket can be expressed as a 2 row by 1 column matrix:

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

3.2 Matrices of the Operators of the Gates X, Y and Z

The operations of the X, Y, and Z gates are expressed by using the 2×2 matrix of the gate, multiplying it by the 2×1 matrix of the basis state (one or zero ket meaning ground state or active state) and getting the result.

The X Gate. An X gate, called the "flip gate" because it always flips the qubit 180° about the x-axis, so a one-ket becomes a zero-ket and vice-versa. The X gate is a 2 row by 2 column matrix:

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

When we multiply the zero-ket 2×1 matrix by the X gate 2×2 matrix and then reduce it back to a 2×1, we see we have changed it to a one-ket. And vice-versa, we flip the state of the qubit from one-ket to zero-ket by applying the X gate:

$$|1\rangle \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We make a shorthand notation here, where instead of writing out the 2×2 matrix of the X gate, we simply write X in front of the one-ket or zero-ket matrix to show we are multiplying the X gate matrix times the one-ket matrix or the zero-ket matrix:

$$X \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The Y Gate. The Y gate is a 2 row by 2 column matrix:

$$Y = \begin{pmatrix} 1 & -i \\ i & 0 \end{pmatrix}$$

Multiply the zero-ket 2×1 matrix (we use shorthand, here, writing "Y" outside of the zero-ket 2×1 matrix to show we will multiply Y by it) and then reduce it back to a 2×1, we see we have changed it and note that the negative imaginary number has been changed to a positive i:

$$Y \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix}$$

Multiply the one-ket by the Y gate and we get these results:

$$Y \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix}$$

The Z Gate. The Z gate is a 2 row $\times$ 2 column matrix:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The Z gate multiplied by the zero-ket gives us these results:

$$Z \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

When the Z gate is multiplied by the one-ket, it gives this result:

$$Z \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

3.3 Is It a Hermitian? The Hermitian Test for Reversibility

The matrices of the X, Y and Z gates are Hermitian. A French mathematician, Charles Hermite, developed this test, named after him, and it is also called the self-adjoint matrix. This is a test for pure reversibility in a matrix that is a square. Square means the matrix is in the set of $n \times n$ matrices. To test if a square matrix is Hermitian, there are two steps. First step is to take the complex conjugate of the matrix. Second step is to transpose it. If the result of the two steps looks exactly like the original matrix, that matrix is Hermitian.

In quantum computing, a Hermitian gate is a gate with a Hermitian matrix which keeps the ket pointing on the unit sphere. The significance of the Hermitian gate is that this means that the operation of the gate is completely reversible in one step. The power and danger of quantum computing's ability to crack RSA encryption [9], the most commonly used cipher method for bank-to-bank financial transactions worldwide, is because of its ability to reverse circuits. RSA counts on the difficulty of factoring large prime numbers with current classical computers, who operate using circuits that are not reversible, and therefore take too long to find the factors of a six hundred digit number. Quantum computers are able to calculate so effectively with their reversible circuits, that in just a few years, as soon as

they are stable with enough qubits, they will crack RSA in moments. Another significant user of RSA cipher methods are military communications. Ability to crack this and see all of the communications in plain text and clear voice of their adversary gives great advantage to the military force who has this technology. Another significant characteristic of the Hermitian matrix is that Hermitian matrices have real eigenvalues whose eigenvectors form a unitary basis, a characteristic used in the mathematics of quantum computing. In quantum computing, the Hilbert space is used to predict by calculations the output of operations, and all of the matrices used in the Hilbert space are Hermitian.

The first step, the complex conjugate, means to look at all the complex numbers in the matrix (complex numbers $\mathbb{C}$ contain both a real number part $\mathbb{R}$ and an imaginary number part i) and change the negative complex number signs to positive complex number signs, and vice-versa. When written out, the number value is written first, as a variable name, such as Y, then afterwards, the complex conjugate should have a prime tic mark on that same name, such as Y', or, sometimes it is marked with a star, such as $Y*$, to show it has been changed. In this case, we have changed it to its complex conjugate.

This is reversible, one can go from Y' back to Y in the same way and end up back where we started. Why when we take the complex conjugate, do we not change the negative or positive sign on the 1? Because 1 is not a complex number $\mathbb{C}$. Why do we change the negative to positive sign, or, the positive to negative sign, on the i numbers? Because they are complex numbers, revealed by the fact that they are i, which is the imaginary part of a complex number.

For example, the complex conjugate of the Y gate is Y':

$$Y = \begin{pmatrix} 0 & -i \\ i & 1 \end{pmatrix} \sim comConj \sim Y' \begin{pmatrix} 0 & i \\ -i & 1 \end{pmatrix}$$

The reason the word *complex* is used is to show how to handle imaginary numbers, which is to include them, even if i is equal to 0 or 1 when doing the complex conjugate include them as i: for example, the complex conjugate of $a + bi => a - bi$. If we said that we were taking the conjugate without any imaginary numbers, then it would be $a + b => a - b$. In quantum computing, we accommodate imaginary numbers for a specific reason.

The second step is the transpose. This means to see the square matrix as if it were a page of transparent paper. Take the upper right corner of the square matrix and flip it over so that the upper right corner is now the bottom left corner.

For example, the transpose of the Y' returns us to the original matrix of Y. This means that the Y matrix passes the 2-steps of the Hermitian.

$$Y' = \begin{pmatrix} 0 & i \\ -i & 1 \end{pmatrix} \sim transpose \sim Y = \begin{pmatrix} 0 & i \\ -i & 1 \end{pmatrix}$$

For the X gate, we take the first step, which is the complex conjugate, which changes nothing, because the X gate contains no complex numbers that need to have their negative or positive numbers changed. T

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \sim comConj \sim X' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Then for step two, which is the transpose, the upper right corner and lower left corner switch places as we turn it over. Again, there is no change. We have returned to our original matrix of X. The X gate is Hermitian.

$$X' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \sim transpose \sim X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

For the Z gate, we take the first step, which is the which is the complex conjugate, which changes nothing, because the Z gate contains no complex numbers that need to have their negative or positive numbers changed.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \sim comConj \sim Z' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Then for step two, which is the transpose, the upper right corner and lower left corner switch places as we turn it over. Again, there is no change. We have returned to our original matrix of Z. The Z gate is Hermitian.

$$Z' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \sim transpose \sim Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

3.4 Summary of Literature Review

This review of the literature and of all of the basic mathematical quantum computing concepts needed to understand this paper shows two significant gaps: we found no comparisons of one quantum computer versus another in terms of the user experience, especially for a new student learning quantum computing. Additionally, we found a gap in documented methodology in the literature to use to evaluate that user experience from one quantum computer to another.

4 Methodology

4.1 Using the IBM Q-Experience

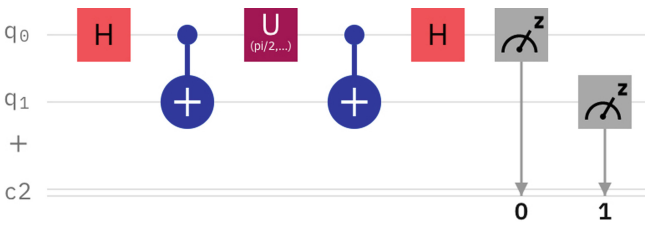

Fig. 5. Michael Nielsen's circuit using U gate

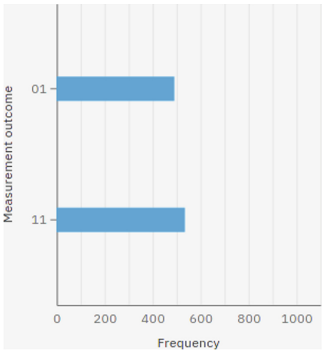

Fig. 6. Histogram results of our U gate are 50% 01 and 50% 11

The results of running the U gate as shown in the Michael Nielen's circuit shown in Fig. 5 are shown in the histogram in Fig. 6, which is a 50% probability of getting the answer 01, and a 50% probability of getting the answer 11. When reading the first answer, 01, that means the qubit 0 resulted in a 1 value after measurement. This binary number is equivalent to 1 in decimal. The measurement of qubit 1 resulted in value 0.

"This representation of the bitstring puts the most significant bit (MSB) on the left, and the least significant bit (LSB) on the right. *This is the standard ordering of binary bitstrings.* We order the qubits in the same way (qubit representing the MSB has index 0), which is why Qiskit uses a non-standard tensor product order" [4].

In other words, binary should be read from right to left in the IBM Q-Experience.

Then we tested each of the single qubit Pauli gates. Figure 7 shows the X gate.

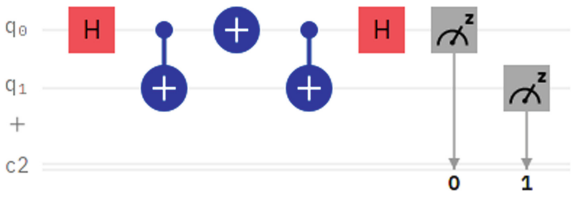

Fig. 7. Our circuit for this test with X gate

When we ran this circuit on the IBM-Q Experience quantum computer simulator, for 1,000 shots, we received these results shown in Fig. 8. The simulator results can give a perfect result, such as 100%, but running the quantum circuits on the real quantum computers always has error in it.

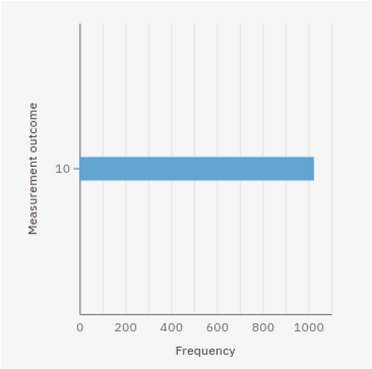

Fig. 8. Histogram results of our X gate are a 100% chance of 10

Figure 9 shows our quantum circuit using the Y gate on qubit 0.

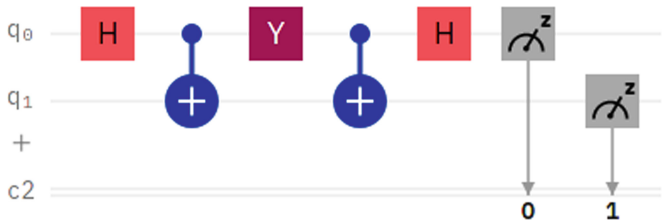

Fig. 9. Our circuit for this test with Y gate

When we ran this circuit on the IBM-Q Experience quantum computer, we received these results shown in Fig. 10.

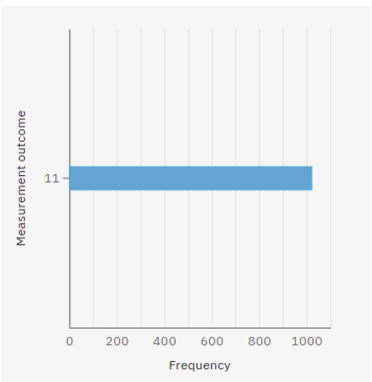

Fig. 10. Using Y gate on qubit 0 results in 100% chance of output 11

Figure 11 shows the circuit using the Z gate on qubit 0.

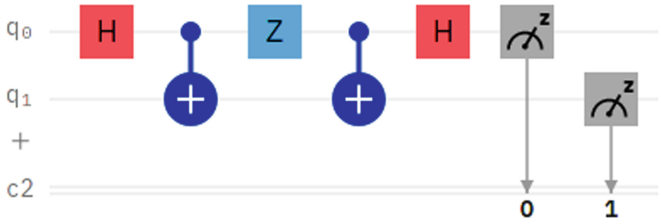

Fig. 11. Our circuit for this test with Z gate

When we ran this circuit in Fig. 11 on the IBM-Q Experience quantum computer, we received these results shown in Fig. 12.

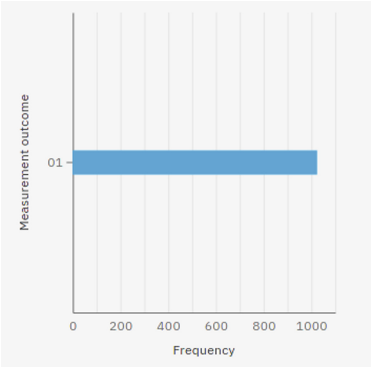

Fig. 12. Results of our Z gate show a 100% probability of output 01

Then we tested the quantum circuit using "XZ" as in Fig. 13 and again using "ZX" as shown Fig. 14 to compare the differences. This is how we found that Michael Nielsen's video on superdense coding did result in the answer '11' for both, however, there was more information that was not accounted for.

Fig. 13. Two gates of XZ on qubit 0 in place of U gate

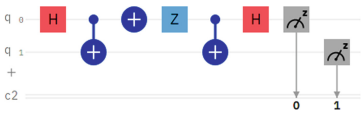

Fig. 14. Two gates of ZX on qubit 0 in place of U gate

4.2 Using the IonQ

Using the IonQ computer, with the Amazon Web Services Braket service, we found the X, Y, and Z gates to work just like the IBM Q-Experience. We found an important difference in the results: IonQ (as well as Amazon Braket's other simulators and quantum computers) deliver the binary results in reverse order than the IBM Q-Experience. As you can see in Fig. 15, the results of the IonQ are read from left to right. So, a result of 01 (the equivalent of 2 in binary) means that qubit 0 had the value 0, and qubit 1 had the value 1 after measurement. 01 should be read as a 10 (the equivalent of 1 in binary). IonQ uses the standard tensor product order.

```
In [4]: # chooses the cloud-based managed real IonQ quantum machine to run your circuit
        device = AwsDevice("arn:aws:braket:::device/qpu/ionq/ionQdevice")

        # execute the circuit
        task = device.run(bell, s3_location, shots=1000)
        # display the results
        print(task.result().measurement_counts)

        Counter({'11': 962, '01': 26, '10': 11, '00': 1})
```

Fig. 15. Reading the bits in reverse order

Also, the IonQ does not give us the visualization of the Q-sphere that helped us see what the difference was between ZX and XZ. It is not visible - the results are both 11 and the important quantum information about the results, that one created a vector that was 180° from the other, is lost.

The circuits we designed to run on the IonQ machine, are the XZ gate, as shown in Fig. 16, and the ZX gate, as shown in Fig. 17, and one can see immediately that the visualization of the circuit is not as clear to the user as it is with the IBM Q-Experience:

```
T  : |0|1|2|3|4|5|

q0 : -H-C-X-Z-C-H-
        |     |
q1 : ---X-----X---

T  : |0|1|2|3|4|5|
```

Fig. 16. ZX circuit on the IonQ

```
T  : |0|1|2|3|4|5|

q0 : -H-C-Z-X-C-H-
        |     |
q1 : ---X-----X---

T  : |0|1|2|3|4|5|
```

Fig. 17. XZ circuit on the IonQ

The XZ circuit consists of a Hadamard (H) gate on qubit 0, a controlled NOT (CNOT) gate with qubit 0 as the control and qubit 1 as the target, a Z gate followed by an X gate on qubit 0, another CNOT gate with qubit 0 as the control and qubit 1 as the target, and lastly another H gate on qubit 0. The ZX circuit is identical, but with the X and Z gates in the opposite order.

The H gate is a single-qubit gate that is used to put the qubit into superposition, which means that instead of being either a 0 or a 1, it is a combination of these two states. The H gate is reversible, so when used for the second time it takes the qubit out of superposition.

The CNOT gate is a 2+ qubit gate that is used to "flip" the target qubit's state (in our case, qubit 1) only if the control qubit (in our case, qubit 0) has a state of 1.

The Z gate is a single-qubit gate that rotates the qubit 180° about the Z axis, when representing the qubit as a vector on the Felix Bloch sphere [1]. The Bloch sphere is a diagram that is used to represent the state of a qubit and is shown in Fig. 18.

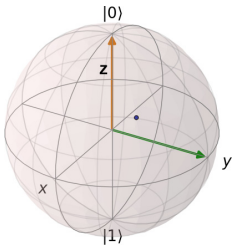

Fig. 18. Diagram of the bloch sphere[8]

The X gate is a single-qubit gate that is similar to the Z gate, but instead rotates the qubit 180° about the X axis.

4.3 Advantages of Proposed Method

In order to compare the two platforms, IonQ via AWS and the IBM Q-Experience, multiple characteristics of each platform were compared. The same

circuits, the XZ and ZX circuits, were tested on both platforms in order to compare the accuracy of our results. The user interface for building and executing circuits was also compared, as well as the price for using each platform. Then, the information received when building and executing circuits on each platform was compared. For example, the Q-Experience provides the measurement outcome graph, the Q-Sphere visualization, and the probability graph, while IonQ provides only the number of shots resulting in each outcome. Finally, we compared any other characteristics of each platform that we found, such as the IBM Q-Experience's unconventional bit string order, which reads right-to-left from the most significant bit to the least significant bit.

This approach finds advantages and disadvantages for both platforms for users of all experience levels, while being cohesive and examining various important aspects of both platforms. This is summarized in Fig. 19.

Platform	IonQ	Q-Experience
Cost	$0.01/shot	Free
Bit order (right to left)	LSB to MSB	MSB to LSB
Visualizations	None	Q-Sphere (phase shift), probability graph, measurement outcome graph
Accuracy (average percent in our tests)	96.6	92.4
Circuit creation tool	Python (Jupyter notebooks)	Graphical drag-and-drop, Python (Jupyter notebooks), OpenQASM

Fig. 19. Summary of advantages and disadvantages

5 Findings

Based on the results of both the IonQ and IBM Q-Experience machines, we found that both the XZ and ZX circuits result in the output 11. While at first this seems to show that the X and Z gates are interchangeable, this is not the case.

The IBM Q-Experience has a diagram of the Q-sphere, which provides us more information than just the results measured by the classical computer. This allows us to see that although the XZ and ZX circuits gave the same binary results, they have a different phase.

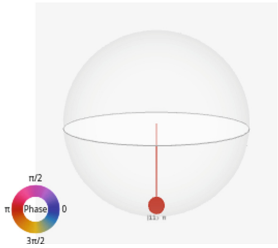

Fig. 20. The Q-sphere for the XZ circuit[8]

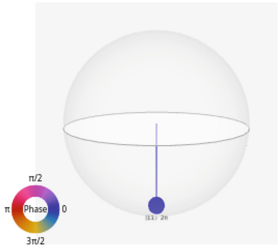

Fig. 21. The Q-sphere for the ZX circuit[8]

As seen above, the XZ circuit has a phase of π, while the ZX circuit has a phase of 0. This means that the matrices contain the same values, but in a different order, resulting in different vector direction.

$$XZ => \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
$$ZX => \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Note the difference in the location of the -1 in the matrix. The output in both cases is 11 but the phase difference was not shown in our results. This is an example of how using classical computers to obtain results from quantum computers loses some information, which will be improved in the future when we are able to reveal more of the quantum computing calculations in our results than we can today.

Also, it should be noted that Amazon Braket/IonQ did not provide a visualization of the Q-sphere, which makes the gates seem interchangeable based on our results.

Since qubits are very difficult to manipulate precisely, they are prone to error. Quantum computers can have varying degrees of error, caused by various factors.

The IonQ computer had more accurate results than the quantum computers we tested on the IBM Q-Experience, as you can see in Table 1 and Table 2. This is a direct comparison to the results we obtained from our tests of the same circuits on the IBM-Q Santiago computer as shown in Table 3 and Table 4.

Table 1. IonQ ZX circuit results showing accuracy rate

IonQ ZX Circuit

Result	11	01	10	00
Shots	970	19	9	2

For the ZX gate, the computer got the correct answer 970 out of the 1000 shots we ran it for. In contrast, in our testing, the most accurate quantum computer on the IBM Q-Experience was "ibmq_santiago". The results are shown in Table 3 and Table 4.

Table 2. IonQ XZ circuit results showing accuracy rate

IonQ XZ Circuit

Result	11	01	10	00
Shots	962	26	11	1

For the XZ gate, the computer got the correct answer in 962 out of the 1000 shots, or times we ran the circuit. So, on average, the IonQ computer was 96.6% accurate in our testing.

Table 3. ibmq_santiago ZX circuit results showing accuracy rate

ibmq_santiago ZX Results

Result	11	10	01	00
Shots	934	30	35	1

Table 4. ibmq_santiago XZ circuit results showing accuracy rate

ibmq_santiago XZ Results

Result	11	10	01	00
Shots	923	28	38	11

In this case, it does not matter (since our desired result, 11, is the same backwards), but note that IBM and IonQ use binary in the reverse order of each other. The top row in the table is different for IonQ and IBM to reflect this.

As seen in Table 4, the XZ circuit got the correct result 934 out of the 1000 shots. As seen in Table 3, the ZX circuit got the correct result 923 out of the 1000 shots. So, on average, in our testing, the "ibmq_santiago" computer was approximately 92.4% accurate.

Both the IonQ and "ibmq_santiago" machines took multiple minutes to get results back from. The IonQ machine took slightly longer to complete.

In conclusion, the IonQ machine offers superior accuracy to the IBM machines, but is more complex to use. The IBM Q-Experience is much more straightforward and provides more information, but is less accurate.

The IBM Q-Experience seems to be the better choice for those that are new to the field of quantum computing, with its well-written tutorials, educational resources such as QISKit, its free time to use actual quantum computers, and its drag-and-drop interface.

IonQ seems to be the better choice for those with more experience in the field of quantum computing. For projects which need high accuracy results, IonQ would be a better choice than IBM. The Amazon Braket Developer Guide provides sufficient information to create quantum computing circuits for those with a higher understanding of quantum computing. Having some background with AWS would also be beneficial, as it is somewhat confusing to navigate for people that are new to it.

6 Future Work

In the future, we will study the design of the IonQ computer, because of its low error rate, and identify its strengths so that they could be used by other quantum computing companies to improve their designs and decrease their error rate. IBM, for example, could improve their quantum computers based on these strengths. If IonQ doesn't give us sufficient information, we will study other high-accuracy quantum computers that are available to use over the Internet, such as Rigetti's computers, and identify their advantages, which could be implemented on IBM's quantum computers.

Recently IBM is hosted a challenge where they are requesting participants' help in "reducing gate errors" and "increasing circuit fidelity for graph state preparation," for a $100,000 reward [5]. This and similar incentives will likely lead to innovative new solutions to these problems by continuing to stimulate research in comparing quantum computers.

The improvement of the design of quantum computers comes from adding more qubits, in order to solve more complex problems. Currently, quantum computers have an increasing rate of error as more qubits are added. This should be solved because quantum computers have a wide range of applications. These include encryption algorithms, social network analysis, and medicine design. Advancements in fields such as these could improve many aspects of our lives, such as security on the Internet, physical safety, and health, respectively.

References

1. Bloch, F.: Nuclear induction. Phys. Rev. **70**(7–8), 460 (1946)
2. Dirac, P.A.M.: The Principles of Quantum Mechanics. Number 27. Oxford University Press (1981)

3. Dowling, M.R., Nielsen, M.A.: The geometry of quantum computation. Quantum Inf. Comput. **8**(10), 861–899 (2008)
4. IBM Research: Getting started with qiskit (2021). https://quantum-computing.ibm.com/lab/files/qiskit-tutorials/tutorials/circuits/1_getting_started_with_qiskit.ipynb
5. Lanes, O., Kim, J.-S., Sheldon, S.: The open science prize: solve for swap gates and graph states (2020). https://www.ibm.com/blogs/research/2020/11/open-science-prize/
6. Nielsen, M.: Superdense coding: how to send two bits using one qubit (2010). https://youtu.be/w5rCn593Dig
7. Nielsen, M.: Quantum computing for the determined (2010). https://michaelnielsen.org/blog/quantum-computing-for-the-determined/
8. QuTiP: Bloch sphere (2021). http://qutip.org/docs/4.1/guide/guide-bloch.html
9. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
10. Westfall, L., Leider, A.: SuperDense coding step by step. In: Arai, K., Bhatia, R. (eds.) FICC 2019. LNNS, vol. 70, pp. 357–372. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-12385-7_28