A Quantum Computing Architecture based Decoherence Model

Lewis Westfall and Charles Tappert, PhD
Seidenberg School of CSIS, Pace University, New York

Email: {lw19277w, ctappert}@pace.edu

Abstract—Quantum computing is a field that only recently had operating quantum computers to run programs on. Up until their advent all programs were run on simulators. Simulators are very helpful to ensure that the programs do what they are designed to do. Unfortunately there is a gap between the nice clear cut results that the simulators generate and somewhat ambiguous results that the real quantum computers generate. The difference is the noise inherent in the current implementation of quantum computers. This paper describes a quantum computing architecture-based de-coherence model developed on the available IBM Q-Experience quantum computers. The approach used was by using entangled qubits in superdense coding, to investigate the effects that noise has on the results of real world quantum computers. In conducting this research, problematic effects were found in an optimization step in the software engineering design of the newest development of dynamic, adaptive, quantum circuits found in at least one of the quantum computing architectures of the IBM Q-experience, so these effects were addressed to obtain more accurate results.

I. INTRODUCTION

An important tool of quantum computing is entangled qubits. Their power has been demonstrated in algorithms like Superdense Coding [1] and Quantum Teleportation [2]. These have been studied on simulators and quantum computers such as the IBM Quantum Experience (IBM Q) [3]. This paper explores the different architectures used by the quantum computers in IBM Q and the performance differences between pairs sets of Qubits. The research was done using Superdense coding to explore entangled qubit performance. Recent changes to the software engineering of the family of IBM Q computers available openly to researchers may have introduced a problematic step as the newest development of adaptive quantum circuits means that when a circuit program is written, it is not executed as written, but instead goes through an intermediate step, middleware, meant to optimize and reduce the processing time that a quantum computer spends on that program, and that step that can cause a greater number of swap gates than is needed, and these swap gates fatigue the qubits, resulting in a higher number of errors in the output results.

A. Superdense Coding

Superdense coding is the ability of entangled qubits to carry more information that a classical bit allows [4]. This is possible because of quantum entanglement, which conveys a quantum state from one qubit to another [5]. The use of superdense coding by entanglement is a quantum cryptography method of great promise to defeat any eavesdropper [6]. Frequently the example is of Alice sending two classical bits of information in one qubit to Bob. In our case, Alice wants to tell Bob that the weather is clear or cloudy and cold or warm. The process starts when a third party, Eve, starts with two separate qubits, entangles the pair by applying a Hadamard (H) gate to the first qubit and then a Controlled Not (CNOT) gate to both qubits, where the first qubit is the control and the second qubit is the target. This entanglement is a Bell state [7].

Once the CNOT gate is applied, entanglement occurs between the two qubits. It is important in this exercises is to remember that quantum gates are reversible, unlike classical gates. The CNOT gate can go both ways - it can entangle, and it can disentangle. One qubit is given to Alice while the other is given to Bob. The qubits of Alice and Bob will remain entangled although one of the qubits is subjected to more gates than the other qubit. Table I shows what message options Alice has to send to Bob, the binary code for that message, and the quantum gates she will use to encode this message information to her qubit. Because of the entanglement, when Alice does her quantum gate operations on her qubit, she can not use a standalone gate because she has an entangled qubit. The gate she will use will be the tensor product of the desired gate with the Identity matrix to create a 4x4 matrix.

Although the qubits are entangled, while the qubits are in the possession of their original owner, only Alice can perform gate operations on her qubit, and only Bob can perform gate operations on his qubit. Once Alice’s qubit has been encoded by operations of the quantum gates, it is transmitted to Bob. Bob, with Alice’s qubit now in his possession, as well as his own qubit, can perform quantum gate operations on Alice’s qubit to discover the message in it. Bob first applies a Controlled-Not gate operation to both entangled qubits, where Alice’s qubit is the control and Bob’s qubit is the target. Applying the CNOT to an entangled pair of qubits causes them to become disentangled and break into two independent qubits.
Bob first measures the second qubit, the formerly target qubit that was Bob's qubit. Measurement ends the qubit's activation life, but it reveals data, which is either cold or warm. Bob then applies an H gate to Alice's original qubit, the first qubit, the former control qubit of the now disentangled pair. The H gate operation extracts the second bit of information in the message, i.e. clear or cloudy. When the qubit is measured the binary code that reveals the message that Alice encoded in her qubit is displayed.

The measurements end the process [8]. Circuit 1 shows the complete quantum circuit diagram for Alice using the X gate to encode warm and clear. We see the Hadamard and CNOT gates used by Eve to entangle, the Unitary gate, or set of gates, U, used by Alice to encode the message, and finally the CNOT and Hadamard gates used by Bob to extract the message.

**B. IBM Quantum Computing Architectures available free to researchers as of publication**

There are 13 different IBM Q computers that are available free to researchers as of this publication, and they are of four distinct architectures [9]: the 5 qubit butterfly seen in Figure 1, the 5 qubit linear seen in Figure 4, the 5 qubit T seen in Figure 3, and the 15 qubit system seen in Figure 2. There is one butterfly computer named ibmq_5_yorktown - ibmqx2, two linear computers named ibmq_athens and ibmq_santiago, three T computers named ibmq_ourense, ibmq_valencia, and ibmq_vigo, and one 15 qubit computer named ibmq_16_melbourne. This research, using superdense coding, was conducted on these quantum computer architectures.

In the illustrative Figures 1, 2, 3, and 4 where these architectures are visualized: the circles represent the qubits and the lines between the circles represent the connections between qubits. A mathematician, or a deep learning scientist, would say, the nodes represent the qubits and the edges represent the quantum channels. The quantum channels are the connection paths available between qubits in which the gate operations take place.

On IBM Q's website the circles and lines have color gradients that represent the error rate. The circle color for single qubit operations and the color of the lines between the circle represent the error rate for operations that use the specific pairs of qubits. The darker the color, the clearer the communications, and the lower the error rate.

As can be seen from the diagrams, not all qubits are directly connected to each other but must interact through one or more other qubits. This research will only look at the performance of two qubits that are directly connected and will not test the qubits that are not. The performance of two qubits that are not directly connected requires a more nuanced measurement and the initial findings of that effort are described in the Future Work section.

**C. New IBM Quantum Computing Architectures available to IBM Partners and Customers**

There are 12 additional IBM Quantum Computers available to IBM Partners, however, they are not openly available for free. These quantum computers have greater quantum volume, more qubits, and lower error rates.

The IBMQ Manhattan is a 65 qubit computer that uses the Hummingbird r2 processor as seen in Figure 5.

**D. IBM Quantum Computing Simulators**

IBM makes high performance quantum computing simulators available in the open-source QISKit software stack using a package called IBM offers a high-performance simulation framework called Qiskit Aer. There are 5 simulators available for researchers that are accessed through the IBM Cloud: the Clifford gate simulator can simulate a 5,000 qubit quantum computer, the Matrix Product State can simulate a 100 qubit computer, the Extended Clifford can simulate a 63 qubit computer, the Schrodinger wavefunction simulator is capable of up to 32 qubits and finally, a General, context-aware simulator can model a 32 qubit machine. All of these cloud based simulators can emulate four different noise models: decoherence, depolarizing, Pauli and general noise. QISKit can also be installed on a local computer with the capability to run three different circuit simulators: the statevector, the stabilizer and the extended-stabilizer, each of which can emulate the same four different noise models of decoherence, depolarizing,
Pauli and general noise. IBM simulators are used to examine to circuits, rapidly prototype circuits for design and characterize the noise response and sensitivity of a circuit. However, to study response to real noise and test how circuits respond and run on real quantum devices, the programs must be run on real IBM hardware.

II. LITERATURE REVIEW

The challenges of error correction between two connected qubits is likely NP-Hard, as this topic is being approached relentlessly, making incremental improvements in heated competition. It is similar to how scientists use linear algebra on solving for a Bounded Error Quantum Computing Polynomial (BQIP) problem. Its complexity is really the same problem viewed over and over from a different linear algebra perspective until one reveals the solution.

The competition is so intense that Google’s recent patent on an adaptive method for adjusting the boundaries of acceptable error that should be easy to read is unintelligible, quickly written to claim that intellectual property first, like a flag on the moon [10]. Google and its parent company, Alphabet, claimed “quantum supremacy” [11] last year, with the head of Google Research quickly becoming promoted to CEO of Alphabet because of that claim [12]. Then, other researchers found they could not replicate the results, as Google’s Bristlecone and Sycamore quantum computers are carefully guarded proprietary information that is not open for outside researchers to independently confirm (as compared to IBM, that does allow free access to its quantum computers via cloud computing), so the powerful initial claim [13] that a multiple qubit machine using the annealing design did not have the suspected decoherence problems [14]. If their claim was true, Google’s quantum computing supported Artificial Intelligence would have become much more accurate quickly.

Google redefined what was meant by “quantum supremacy”, to mean that they had performed at least one math operation that could not be done by any known classical computer, a much lower bar than the original idea [15].

III. METHODOLOGY

The first task is to run the programs on the simulator to verify that they give the expected results and set the expectation for the results to follow.

The actual tests will be run on the ibmq_5_yorktown - ibmqx2 Which has a butterfly architecture. A base line was established for the performance of entangled adjacent qubit by running the superdense coding tests on qubits 0 and 2. The original plan was to run the tests on qubits 0 and 1, but since the no-adjacent tests would be run on qubits 0 and 3, which are connected through qubit 2, see figure 1, the decision was made to use qubits 0 and 2 to eliminate any difference in behavior that depends on the qubit pair used.

Once the base line is established the next tests will be running superdense coding on different sets of adjacent qubits.

Superdense coding allows one qubit of an entangled pair to carry two classical bits of data [3]. The circuit, Figure 6, shows the composer circuit, how IBM Q graphically presents the circuit, for superdense coding between qubits 0 and 2 on the ibm-qx2 butterfly computer. On this computer qubits 0 and 2 are adjacent.

The Hadamard gate followed by the CNOT gate entangles qubits 0 and 2. To encode the two classical bit pattern on the entangled qubits an operation is performed on qubit 0. Table 1 shows the operations or gates used for each of the desired bit patterns. The final CNOT and Hadamard gates disentangle the qubits and decode the message.
Fig. 6. Circuit for Superdense Coding

<table>
<thead>
<tr>
<th>Temp</th>
<th>Sky</th>
<th>Code</th>
<th>Quantum Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>Clear</td>
<td>00</td>
<td>I</td>
</tr>
<tr>
<td>Cold</td>
<td>Cloudy</td>
<td>01</td>
<td>Z</td>
</tr>
<tr>
<td>Warm</td>
<td>Clear</td>
<td>10</td>
<td>X</td>
</tr>
<tr>
<td>Warm</td>
<td>Cloudy</td>
<td>11</td>
<td>Y</td>
</tr>
</tbody>
</table>

TABLE I
MESSAGE CONDITIONS AND ACTIONS

IV. RESULTS

“In fact, the mere act of opening the box will determine the state of the cat, although in this case there were three determinate states the cat could be in: these being Alive, Dead, and Bloody Furious.” [16]

There appears to be a optimizer, a middleware that is not visible, between the code generated in the Jupyter Python Notebook and the code that is actually run on the quantum computer. This was first observed when a test was run trying to entangle qubits 0 and 4 on the T architecture of the ibmq_ourence quantum computer. Qubits 0 and 4 are not adjacent and if they were to interact, they would have to do so through intervening qubits. The composer circuit in the python notebook had only five operations and would have, in theory, entangled qubits 0 and 4, Figure 8, but the composer circuit after the optimizer had many more operations, including two sets of qubit swaps and entangling qubits 3 and 1, Figure 9. Since the qubits that the optimizer used are adjacent it will require further research to determine if non-adjacent qubits can be entangled. NB the controlling qubit is the first one specified and the target qubit is the second one specified.

Table II shows the results of running the superdense circuit using the I gate to encode '00'. The simulation had 1024, 100%, of '00'. The first actual run had 929 '00', 90.7% and the second run had 915 '00', 89.4%. The optimizer decides that some operation can occur at unexpected times. To prevent the optimizer from rearranging the operations a 'barrier' can be inserted in the quantum code between steps. After inserting the barriers in the code the test had 911 '00', 89.0%.

Note that since the tests use qubits 0 and 2, the only bits of the output pattern that are of significance xx0x0.
Table III shows the results when using the Z gate to encode '01'. The simulation had 1024 '01', 100%. The actual test had 880 '01', 85.9%. The barrier test had 874 '01', 85.5%.

<table>
<thead>
<tr>
<th></th>
<th>00000</th>
<th>00010</th>
<th>00100</th>
<th>00101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>1024</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Run 1</td>
<td>929</td>
<td>71</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Run 2</td>
<td>915</td>
<td>67</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Barrier</td>
<td>911</td>
<td>71</td>
<td>28</td>
<td>14</td>
</tr>
<tr>
<td>Accuracy</td>
<td>90.7%</td>
<td>89.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barrier</td>
<td>89.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III**

**QBITS 0 & 2 - ACTION Z**

Table IV shows the results of using the X gate to encode '10'. The simulation test showed 1024 '10', 100%. The actual run without the barriers had 876 '10', 85.5%. The barrier test showed 874 '10', 85.4%.

<table>
<thead>
<tr>
<th></th>
<th>00000</th>
<th>00010</th>
<th>00100</th>
<th>00101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>0</td>
<td>1024</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Run 1</td>
<td>113</td>
<td>880</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>Barrier</td>
<td>103</td>
<td>874</td>
<td>8</td>
<td>39</td>
</tr>
<tr>
<td>Accuracy</td>
<td>85.9%</td>
<td>85.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barrier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV**

**QBITS 0 & 2 - ACTION X**

Table V shows the results of the test using the Y gate to encode '11'. The simulation test had 1024 '11', 100%. The test without the barriers returned 853 '11', 83.3% and the barrier test showed 850 '11', 83.0%.

<table>
<thead>
<tr>
<th></th>
<th>00000</th>
<th>00010</th>
<th>00100</th>
<th>00101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>0</td>
<td>0</td>
<td>1024</td>
<td>0</td>
</tr>
<tr>
<td>Run 1</td>
<td>9</td>
<td>53</td>
<td>109</td>
<td>853</td>
</tr>
<tr>
<td>Barrier</td>
<td>9</td>
<td>49</td>
<td>116</td>
<td>850</td>
</tr>
<tr>
<td>Accuracy</td>
<td>83.3%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barrier</td>
<td>83.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V**

**QBITS 0 & 2 - ACTION Y**

V. Conclusion and Future Work

Running the test on a real quantum computer shows about a 10% degradation of accuracy compared to the simulations. Further the use of barrier also slightly degrades accuracy although the single tests should not be considered conclusive. Future work will include testing Superdense Coding on different pairs of qubits on all the IBM Q computers and testing the accuracy of the complex circuits created by the optimizer compared to more concise circuits.

REFERENCES