HIGH SCHOOL EDITION
Journal of Student Research

Investigation of Quantum Adder Error Rate on IBM Quantum Systems

Cayden Tu[1] and Anthony Hoffman[#]

[1]Saint Ignatius College Preparatory High School, San Francisco, CA, USA
[#]Advisor

ABSTRACT

This paper reports the results of an investigation into the performance of quantum addition operations carried out by two quantum adders. These algorithms were run on IBM quantum computers and simulated with a noise model on Qiskit. Following this experimentation, we studied the rates of errors and the differences between the two adders. This study illuminated the benefits and drawbacks of the adders as well as the capabilities of a noise simulation consisting of coupling and basis gates noise to emulate real hardware. Our work provides additional data to inform the development of more efficient addition algorithms aimed at helping minimize error rates.

Introduction

As the scientific community explores quantum computing, researchers investigate the causes of error and methods increasing precision in real quantum computers. In real quantum hardware, errors are bound to occur. It is important to investigate the degree of errors so that researchers can further increase the precision and reliability of most quantum computers. In a modern classical computer, the error rate is sufficiently low, such that errors do not significantly impact the computer's functioning (Ladd et al., 2010; Obenland & Despain, 1998). However, for quantum computers to factor better than classical computers, quantum hardware's error rate needs to be much lower (Ladd et al., 2010; Obenland & Despain, 1998). For real quantum hardware, the probability of error is much higher than classical computers.

In this project, we look at a specific type of these common errors called "bit-flip errors" (BFE), which occur when one classical bit in a bit string result is the opposite of what it should be. Additionally, we investigate the accuracy of Qiskit's noise simulation of real devices to observe how well it emulates the observed quantum error. We compare the discrepancies of error rates in both real hardware and ideal/noisy simulations of hardware. Specifically, we look at the various results of different arithmetic operations on each of these devices. In my study, we utilize the quantum circuits of two types of adders, a VBE ripple carry adder and the CDKM adder. Ultimately, we compare each adder's error rates for different addition operations on real versus simulated quantum hardware.

Background

Despite modern quantum computers possessing relatively high error rates compared to classical hardware, quantum hardware has the potential to outperform its counterparts. Quantum hardware is much more powerful than classical hardware due to its ability to utilize superposition states, where the system is simultaneously in multiple classical states until a measurement is performed. For instance, a bit could have a 50% chance of being a 0 while also having a 50% chance of being a 1. Currently, on real quantum hardware, errors can accumulate from noise on quantum circuit components, as well as from non-ideal interactions between quantum states and their environment. Although we choose not to utilize superposition states in this study, we still collect data displaying certain inaccuracies of quantum computing without the use of quantum-like traits. Quantum adders are used to gather these results.

To comprehend how classical adders work, one must first understand how they implement binary addition. The binary system is expressed in the form of base-2 positional notation. A classical bit, or digit of a binary number, is an object that can assume the value 0 or 1. Any integer can be represented by a string of 0s and 1s, known as a bit string. The rightmost place in the string represents 2^0, the next-to-rightmost place in the string represents 2^1, and the nth place represents 2^n.

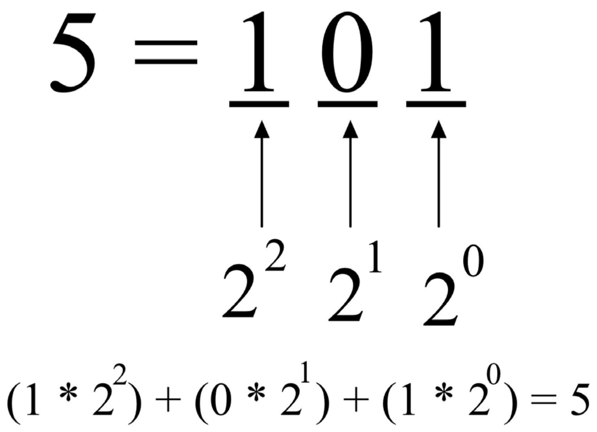


Figure 1. Example of 5 in binary.

For example, as illustrated in figure 1, the number 5 in binary would be 101. The rightmost digit corresponds to the value of 2^0.

In this case, the farthest right 1 in the previous addition example would be $(1 * 2^0)$. Each digit to the left increases by a power of 2. To convert the number to decimal form, each digit's $(0 \text{ or } 1 * 2^n)$ must be added together. Therefore, $(1 * 2^0) + (0 * 2^1) + (1 * 2^2)$ is equal to 5.

Although numbers are represented differently in binary notation and decimal notation, the long addition process is identical for both. In both systems, when adding digits in the same position, the resulting digit in the sum at that position is the sum of the digits added together. However, in binary, when the sum of those two digits is greater than or equal to 2, a carry bit is needed. The carry bit is carried over to the position of bits left of the current position, and in the case of 2, 1 is carried over. However, because 2 is 10 in binary, the 0 is the direct result of adding these two digits together. When we switch over to each digit, we incorporate the carry if there is one. For instance, if a 1 is carried over to 1+1, then the operation would be 1+1+1. Therefore, another carry for 1 would be needed.

Table 2. Truth table for addition logic operations.

A_0	**B_0**	**S_0 (XOR)**	**C (AND)**
0	0	0	0
1	0	1	0
0	1	1	0

1	1	0	1

The logic operations shown in Table I. successfully add numbers together as a half adder. In each half adder, two one-bit numbers are added together, and a carry is also produced. For instance, A_0 and B_0 are added together in binary and the result, s_0, is produced. The XOR gate switches s_0 only when either A_0 or B_0 is 1. For the carry that is produced in addition to s_0, the And gate is used. The carry can only switch from 0 to 1 during one scenario when both A_0 and B_0 are 1. When both conditions are true (in the case of being 1), the And gate performs its function similar to an "and" statement.

A full adder is made up of two half adders together. These two half adders must be added together in order to add three 1-bit numbers—namely, a carry-in bit alongside the two bits to be summed. Full adders can then be strung together to result in a ripple-carry adder, which can add arbitrarily large numbers together, as illustrated below in figure 2.

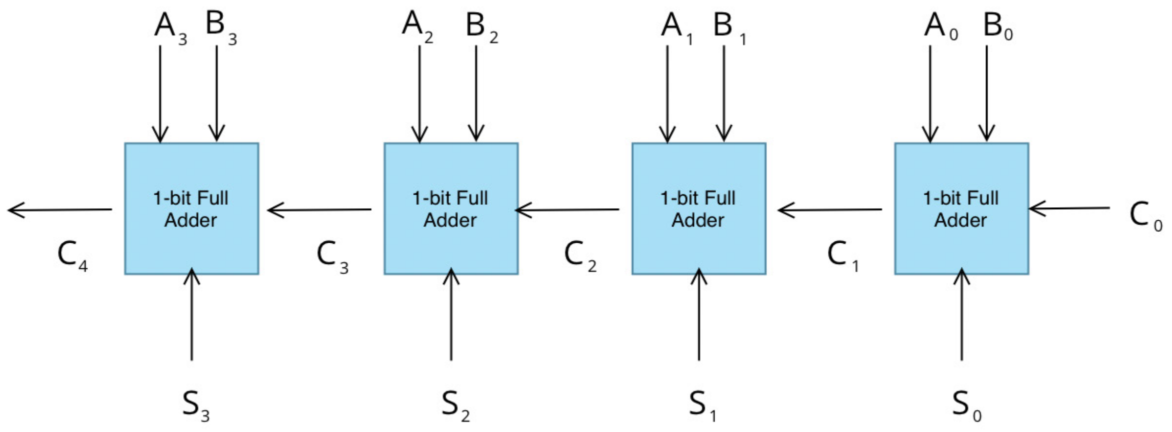


Figure 2. Illustration of full adders strung together for ripple-carry adder. Adapted from Wikipedia ("Adder [electronics]," 2022).

To implement a quantum adder, at least 2n+1 qubits are needed to add two equally sized n-bit numbers. A qubit is the equivalent of a bit in a classical computer, but is instead the quantum state of some system, rather than one of two binary states stored on a device. By convention, the spin-z basis of spin-up |0> and spin-down |1> is typically chosen to correspond with classical bits 0 and 1. The first n-qubit summand is represented as $A_{n-1}\ldots A_1A_0$. Similarly, the second n-qubit summand is $B_{n-1}\ldots B_1B_0$. Finally, there is a carry bit called C that can hold a value of either 0 or 1. A quantum adder circuit takes these inputs through a series of quantum gates to correctly add the summands and the carry together. To set an initial qubit from 0 to 1, an X-gate that inverts the quantum state must first be used. To implement XOR for summing, C-Not gates are used. Each C-Not gate applies an X-gate on the target, depending on the value of the control. If the control is a 1, then the X-gate is applied. If not, nothing changes. Similarly, AND can be implemented by Toffoli gates in order to calculate whether or not a carry is generated. Whereas the specific configuration of C-Not and Toffoli gates will vary from one implementation to another, the overall analysis of the gates above holds true in general. Afterwards, the outputs are measured at the end of the quantum circuit, which collects the resulting sum as classical bits.

Vedral, Barenco, and Ekert (VBE) construct their ripple-carry adder using carry and sum gates (Vedral et al., 1995). First, the carry gate (figure 3A) and its logic will now be described here. A carry-out is generated if at least two of the three bits being summed are 1. If the two summands A and B are both 1, then a carry-out bit is generated regardless of the value of the carry-in. Otherwise, if at least one of the summands is 1, such that their sum yields 1, a carry-out is still generated if the carry-in bit is 1. To implement this, VBE uses a Toffoli gate to implement an AND gate for the two summands to generate a carry-out. They then sum the summands with a C-Not. The resulting sum and the carry-in then go through another AND gate via a second Toffoli gate to generate the carry-out. Notably, VBE uses an inverse carry gate to erase the intermediate carry results, which are calculated in scratch-work qubits. The sum gate (figure 3B) consists of two C-Not gates. The first C-Not gate adds the two summands, which is then added to the carry-in bit by the second C-Not gate.

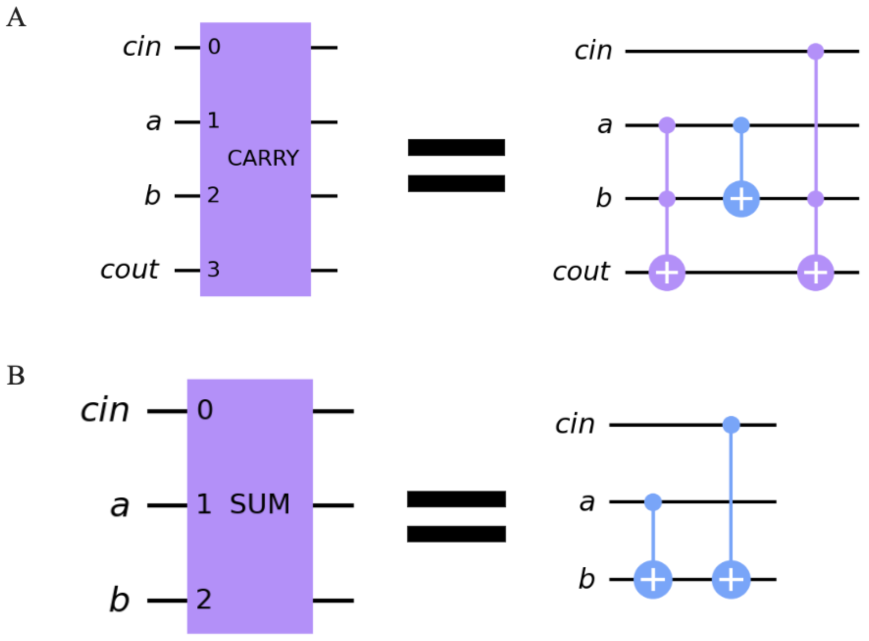


Figure 3. Carry/sum gate for VBE adder. Figure 3A shows a carry gate. Figure 3B is a sum gate.

The ripple-carry adder by Cuccaro, Draper, Kutin, and Moultin (CDKM) instead uses "Majority" (MAJ) and "Unmajority and Add" (UMA) gates (Cuccaro et al., 2004). The MAJ gate (figure 4A) uses two C-Not gates to compute XOR between the two summands *A* and *B* as well as XOR between summand *A* and the carry-in bit, *C*. If summand *A* is 0 and both XORs yield 1, it is implied that summand *B* and the carry-in are both 1. The Toffoli gate then performs an AND operation in order to generate a carry-out of 1 on the qubit for summand *A*. Alternatively, if summand *A* is 1 and both XORs yield 1, summand *B* and the carry-in are both 0. Then, the Toffoli gate performs a NAND (Not-AND) operation in order to generate a carry-out of 0 on the qubit for summand *A*. In all other cases where one or both of the XORs evaluate to 0, the value of summand *A* will be sufficient to determine the carry-out. The UMA gate (figure 4B) first uses a Toffoli gate and a C-Not between summand *A* and the carry-in to uncompute the carry-out, thereby erasing it. Then, it applies a C-Not between the carry-in and the qubit that contains summand *B*. Since the MAJ gate already applied C-Not between the two summands *A* and *B*, the total action of MAJ+UMA is equivalent to summing *A*, *B*, and the carry-in.

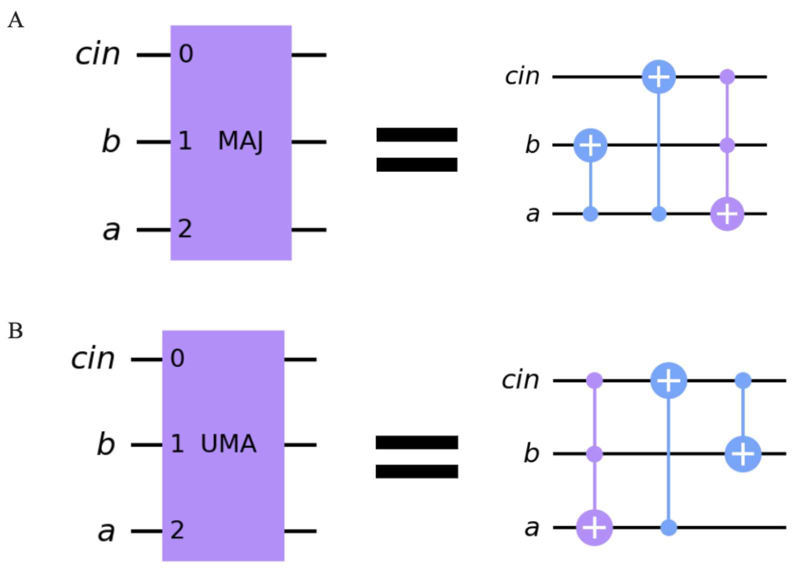


Figure 4. MAJ/UMA gate for CDKM adder. Figure 4A is a MAJ gate. Figure 4B is a UMA gate.

Researchers in the field of quantum computation have proposed several algorithms for addition on a quantum computer, such as Draper's CLA adder (Draper et al., 2004), CDKM's Ripple carry adder ("Adder [electronics]," 2022), and VBE's Ripple carry adder (Vedral et al., 1995). These arithmetic operations are the basics of computing, as shown by the importance of an adept adder to reduce the run-time of Shor's algorithm which relies on arithmetic operations (Draper et al., 2004). Although the accuracy of these adders is crucial, on real hardware a noticeable number of errors is recorded.

Due to the intricacy of the circuits, specifically the number of Toffoli and C-Not gates, an error often occurs when the number does not flip properly as it goes through these gates (Yu et al., 2013). As this paper explains, the Toffoli gate contains five two-qubit gates, which further demonstrates its complexity. Quantum decoherence creates this instability and susceptibility to error. In figure 5B, the Toffoli gate transpiled on an IBM quantum processing unit (QPU) illustrates its intricacy over the C-Not gate in figure 5A. When we implement both adders in IBM Quantum Lab, the effects of this intricacy will become apparent.

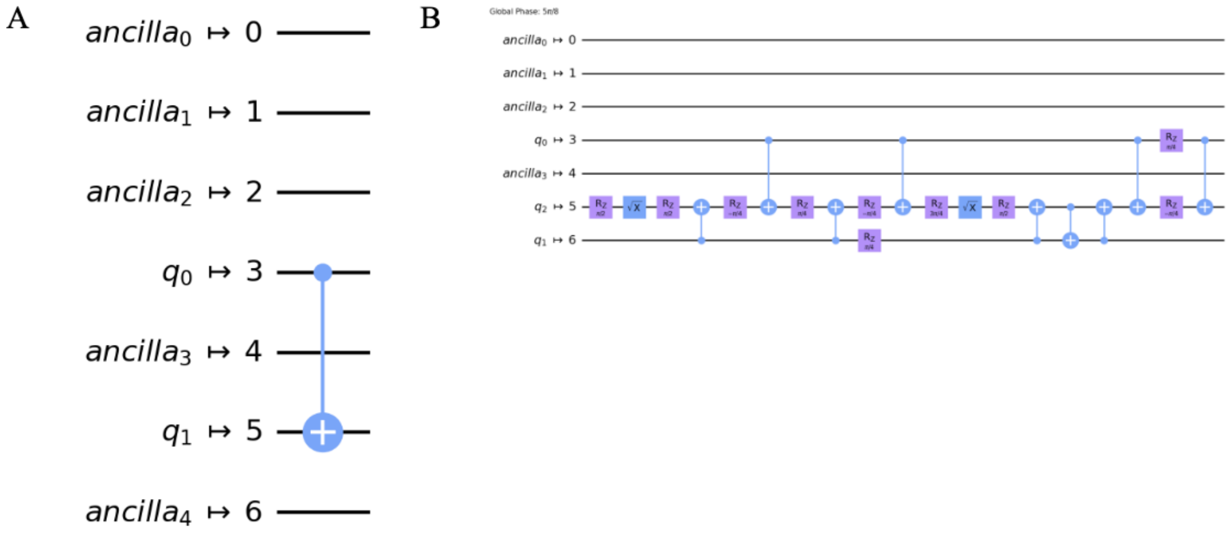


Figure 5. Transpiled C-Not/Toffoli gate. Figure 5A shows the C-Not gate. Figure 5B shows Toffoli gate.

Methods

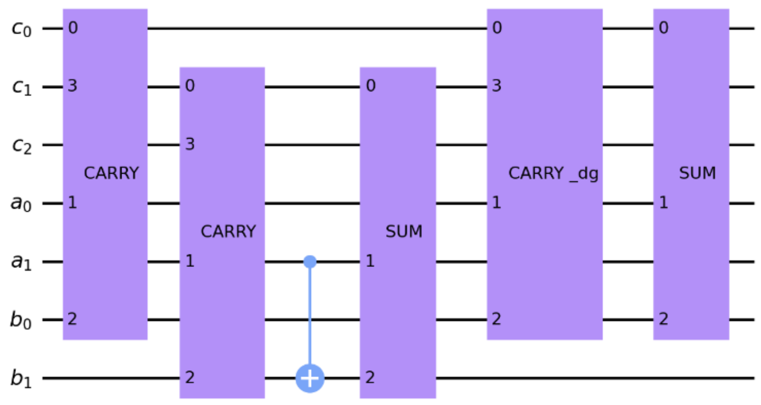
In order to implement the quantum adders on real hardware, we make use of IBM Quantum Lab to run Python code on real quantum processing units. Further, we use one consistent device for all 32 operations of adders. IBM offers multiple QPUs for free; however, we only have access to QPUs with a maximum of 7 qubits: IBM Nairobi and IBM Oslo. To minimize the queue time of each operation run, we choose whichever device has a shorter queue time during the day we run the operations. Though smaller adders could have been devised on a smaller 5 qubit system, we want to maximize the number of qubits that are used. Therefore, with the maximum 7 qubits, two 2-qubit numbers could be added together, including a carry. If fewer qubits are used, a half adder can accomplish the same job and a full adder will not be needed.

The operations range from 0+0+0 to 3+3+1. Denoting the carry bit as C and the two-bit summands as A_1A_0 and B_1B_0, there are 32 possible operations of $A_1A_0 + B_1B_0 + C$. As previously mentioned, all of these could potentially perform differently on real quantum hardware or simulations.

The two types of quantum ripple-carry adders that we use are the VBE and CDKM adders where two-digit addition was implemented in Qiskit. Then, data for the outcomes is measured specifically, 8192 times or “shots” for each run—on IBM's real quantum hardware, as well as simulated with a noise model.

The largest inputs of numbers will be two 2-qubit numbers, excluding the carry because we only have 7 qubits available. For each summand, 11, in binary, or 3 is the largest possible number. The VBE adder is composed solely of sum and carry gates that are shown in the background (figure 6). Two carry gates and two sum gates are needed for a 7-qubit adder, with the inverse carry gate being used for erasure of the scratch-work qubit.

A



B

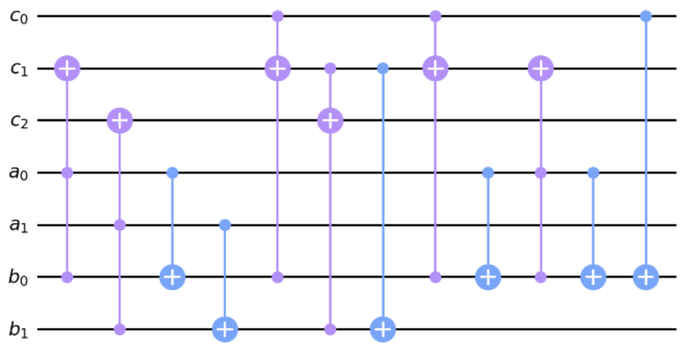
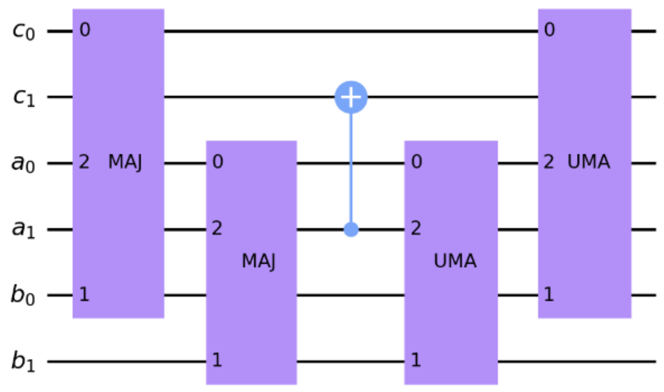


Figure 6. VBE adders. Figure 6A shows a VBE adder. The summands a_0a_1 and b_0b_1 are listed at the bottom while the carry and sums are at the top. For the output, b_0 serves as s_0. b_1 serves as s_1. c_2 serves as the final carry out. Figure 6B is a VBE adder showing the Toffoli and C-Not gates.

Meanwhile, the CDKM is composed of MAJ and UMA gates (figure 7). The adder's inputs are ordered in a similar format as the VBE adder's. However, the CDKM adder only requires two carry qubits, c_0 and c_1, while still running the same 32 operations. Additionally, the CDKM adder only requires 6 qubits.

A



B

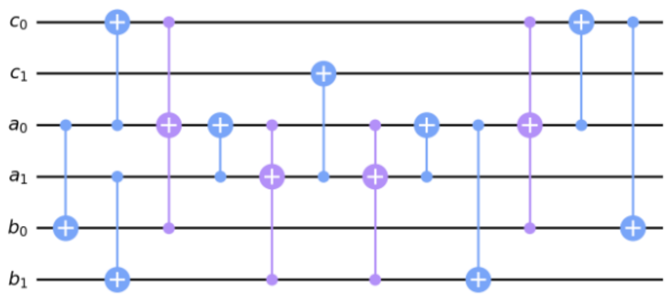


Figure 7. CDKM adder. Figure 7A shows a CDKM adder. The summands a_0a_1 and b_0b_1 are listed at the bottom while the carry and sum are at the top. For the output, b_0 serves as s_0. b_1 serves as s_1. c_1 serves as the final carry out. Figure 7B shows the CDKM adder showing the Toffoli and C-Not gates.

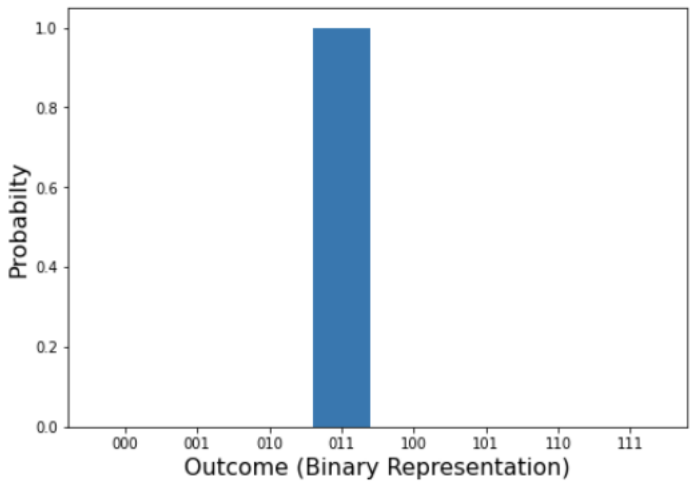
We automate the data collection process to run all 32 operations of a given adder circuit on real hardware, noise simulation, and ideal simulation. In order to run all 32 operations in one job, which would ultimately shorten the queue time, we add all the various circuits of each operation to a circuit list. It only takes one job for the specific data of each circuit to be retrieved because each circuit is listed in the circuit list. The data of the outcome counts is then retrieved. Afterwards, it is processed to generate histograms of the bit-flip errors and a CSV of the average bit-flip errors in each operation.

The device noise model used consists of the qubit coupling from the QPU as well as noise from the basis gates (Qiskit Development Team, 2022). The noise simulation pulls the qubit coupling immediately after the real hardware runs. This makes sure that the noise simulation will use a coupling from the same device to emulate real hardware as closely as possible.

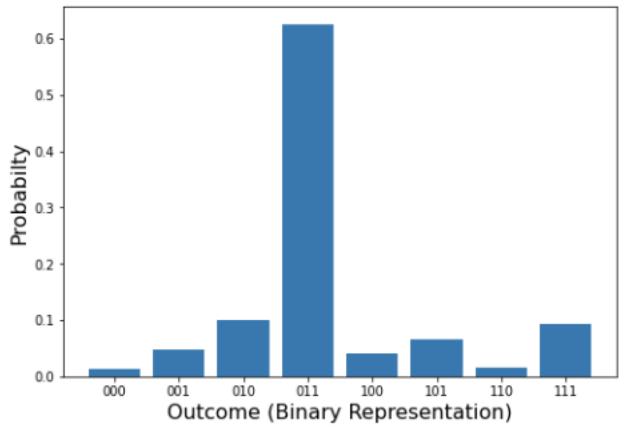
Results and Discussion

For our analysis, the outcome histograms for both adders on both noise simulation and real hardware are examined first. For most operations, including the 3+0+0 operation in figure 8, the noise simulation roughly emulates the real hardware. Looking at the histograms of the noise simulation and the real hardware, we notice that the graphs follow the same pattern. The difference between the graphs is that the noise simulation has a higher probability of getting the correct answer while erroneous results have lower probabilities compared to real hardware. For instance, in the histograms for the operation 3+0+0 shown in figure 8, the noise simulation has around a 60% chance of getting the correct answer while real hardware has around a 40% chance. Although the noise simulation still performs better than the real hardware, there is the highest probability of getting the correct answer in both the noise simulation and the real hardware (figure 8).

A



B



C

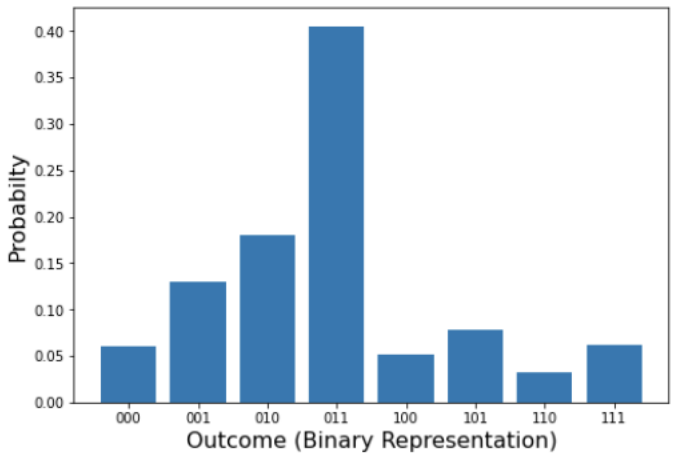


Figure 8. VBE adder 3+0+0. Figure 8A shows the ideal simulation. Figure 8B shows noise simulation. Figure 8C shows real hardware.

In this example of 3+3+1 on the VBE adder, the noise simulation and real hardware both have the highest chance of getting the correct result (figure 9). However, the real hardware still has higher probabilities of getting other results compared to the noise simulation. For the noise simulation, there is a 60% chance of getting the correct result. In contrast, the real hardware has less than a 30% chance. Further, the bars representing the other possible results are much higher for the real hardware.

Looking at the results, we see that the number with the second highest probability is typically one bit-flip away from the correct answer. Specifically, this bit-flip is generally in the least significant digit of the number. Therefore, the quantum process involving the least significant bit is likely the most prone to error. For the VBE adder, the least significant bit encounters the most gates, specifically four C-Not and four Toffoli gates; thus, we would indeed expect the least significant bit to be subject to the most error. On the other hand, the most significant bit of the CDKM adder passes through only one C-Not gate, while both the least significant and second-least significant bit pass through two C-Not gates and two Toffoli gates each. Regardless, the least significant bit is oftentimes still the second-most common result. However, the second-least significant bit is the second most common result far more often than it is for the VBE adder. Therefore, the number of gates encountered seemingly elucidates how error prone a given bit would be for both adders.

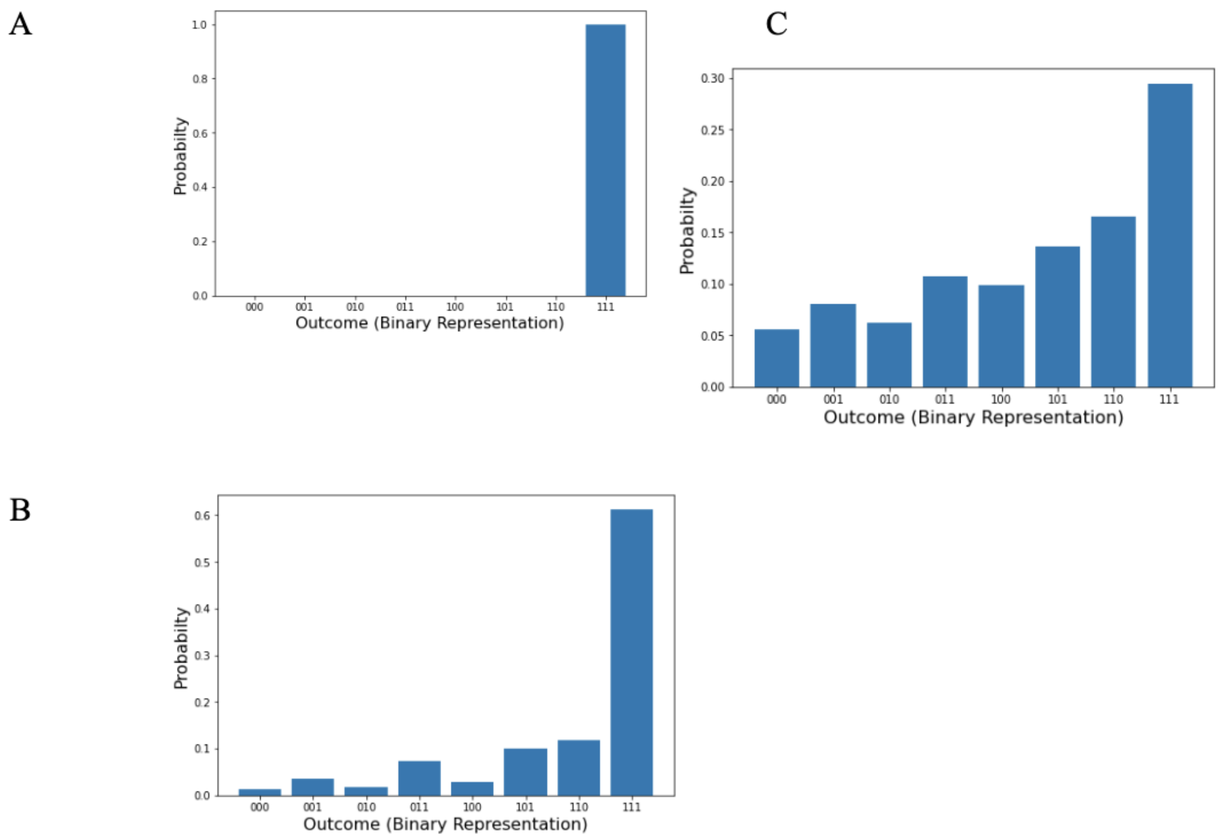


Figure 9. VBE adder 3+3+1. Figure 9A shows an ideal simulation. Figure 9B shows noise simulation. Figure 9C shows real hardware.

After looking at the performances on QPUs compared to simulation, we look at the difference in the performances of the adders. For all operations, the VBE adder implemented on real hardware correctly measures the right outcome with the highest probability compared to all other erroneous results. On the other hand, there are certain operations for which the real hardware CDKM adder would instead be most likely to measure an incorrect result. Operations 0+2+0, 2+0+0, and 3+3+1 would typically result in an erroneous result having the highest probability for the CDKM (figure 10). Occasionally, other operations also yield incorrect results. In one run on the CDKM adder, most operations that have 3 as one of its summands have the highest probability of getting an erroneous result. These operations are 3+3+1, 3+3+0, 3+2+1, 3+1+0, and 3+0+1.

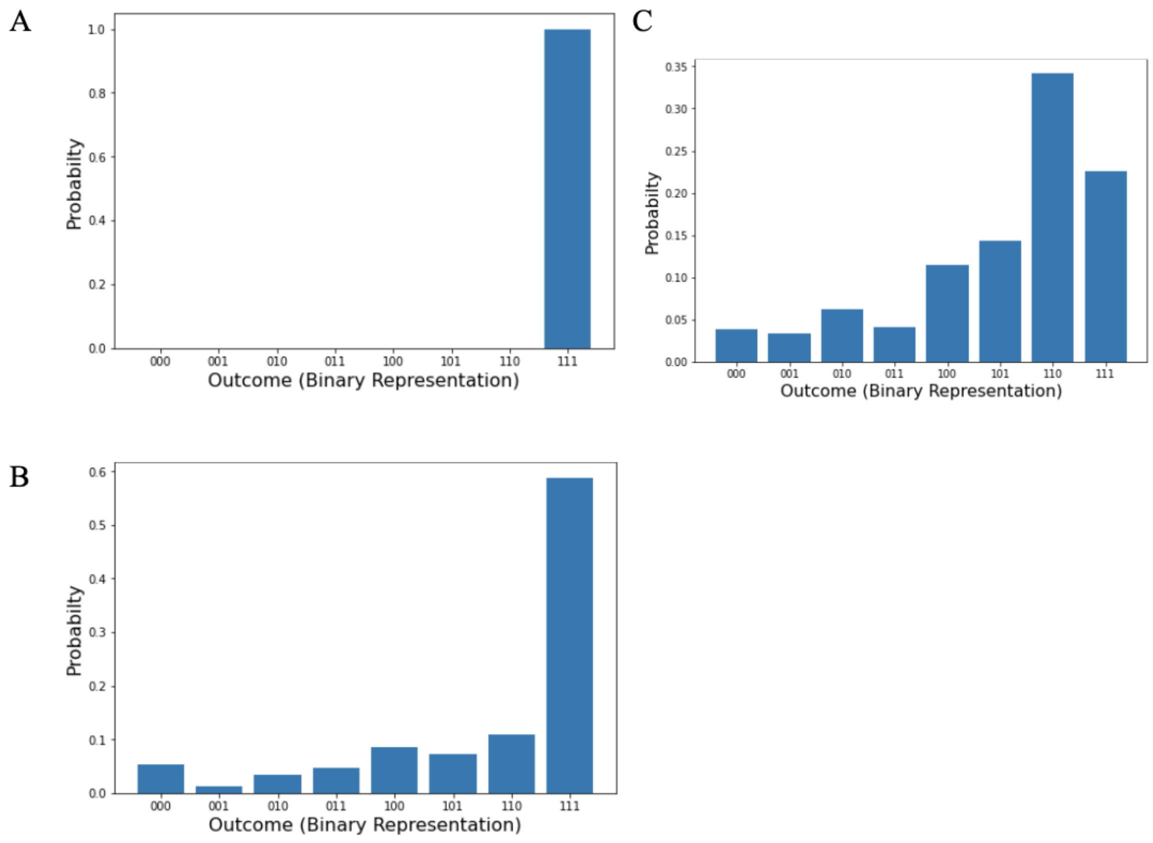


Figure 10. CDKM adder 3+3+1. Figure 10A shows an ideal simulation. Figure 10B shows noise simulation. Figure 10C shows real hardware.

Further, the histograms of the bit-flip error distribution are examined on both real hardware and noise simulation. Results show that, for erroneous outcomes, the percentage of bit-flip errors decreases as the number of bit-flips in each error increases. For instance, the percentage of 3 bit-flip errors is much less than the percentage of 0 or 1 bit-flip errors. Although the percentages can vary, the trend is the same with a high number of 1 bit-flip errors, fewer 2 bit-flip errors, and the fewest 3 bit-flip errors. This trend is logical because, despite the high rates of error, the quantum computing process would have to be extremely erroneous to get all its digits incorrect. In comparison, one bit-flip error would be relatively likely as quantum computing is prone to error, and it is likely that one gate switch could have not worked properly. Regardless, the chances of having no bit-flip errors could be lower than 50%, as shown in figure 11, and can be even lower than the chances of having one bit-flip error, as shown in figure 12A. Simply put, it is more likely to have a bit-flip error, resulting in an incorrect answer, than it is to have none.

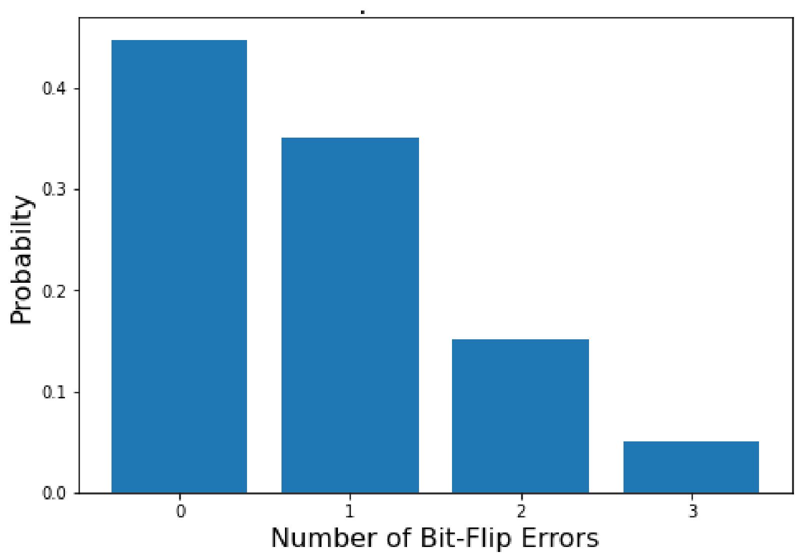
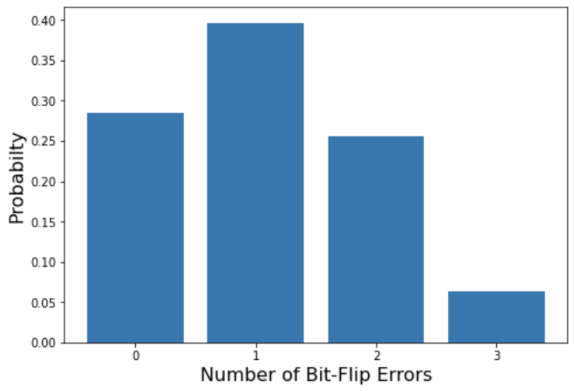


Figure 11. VBE adder 3+0+0 Real hardware bit-flip error distribution.

Throughout multiple runs, the real hardware has higher average bit flip errors than the noise simulation. Looking at the histograms, the real hardware has a much lower probability of having 0 bit-flip errors compared to the noise simulation, which is consistent with the observations made based on the outcome histograms. As shown in figure 12, the noise simulation has above a 50% chance, while the real hardware has below a 30% chance. Aside from that, having one bit-flip error still has a higher probability than two or three bit-flip errors.

A



B

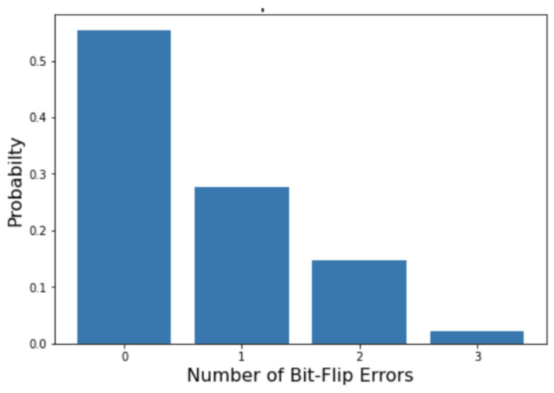


Figure 12. VBE adder 3+2+1. Figure 12A shows real hardware bit-flip error distribution. Figure 12B shows noise simulation bit-flip error distribution.

After running the program, we look at a table of the average bit-flip error for each operation ordered from smallest to greatest in average BFE. Looking at the outcomes histogram for the CDKM adder, we see a trend that emerges amongst a few operations. During multiple runs of the CDKM adder, the operations 0+2+0 and 2+0+0 always have a higher percentage of getting a result other than the theoretically correct one. For 0+2+0, a large percentage, specifically 47%, of the results are 3. Similarly, for 2+0+0, 44% of the results are 3. We can calculate the average bit-flip error of all operations because we have the average bit-flip error for each operation. After running the adders multiple times, the overall average bit-flip error rate is determined. For the noise simulation of the VBE adder, the average BFE is 0.561, while on the real hardware implementation, the average BFE is 0.962. For the noise simulation on the CDKM adder, the average BFE is 0.634, while on the real hardware implementation, the average BFE is 0.893. From the data measured on the QPUs, we see that the CDKM adder typically has a lower average BFE than the VBE adder, despite having more situations in which it would get the wrong number. This implies that the CDKM adder would have less chance of getting other erroneous results, despite being most likely to get a particular incorrect result. Although the VBE adder has higher rates of error, it still produces the theoretically correct answer for each operation more times than the CDKM adder. The VBE distributes its errors amongst numerous incorrect answers, while the CDKM condenses the errors into an erroneous result. Thus, the VBE adder would have a higher chance of giving the correct result, but in order to produce such results, more shots have to be used in comparison to the CDKM adder. For both adders, the noise simulation has a considerably lower average bit-flip error than the real hardware. Therefore, the noise simulation does not account for the amount of error real hardware has. On the other hand, for the CDKM adder, the noise simulation more accurately accounts for the noise. The difference in average BFE between noise simulation and QPUs for the CDKM adder is .259 compared to the VBE adder's .401. Although the average BFE on real hardware for the CDKM adder is lower, the noise simulation's average BFE is higher in comparison to the VBE adder.

Looking at the *c* column shown in tables 2 and 3, we see that operations with a carry bit of 1 have a higher average BFE. This could be due to the fact that real hardware struggles when a carry-in input is involved, and these operations are more prone to error when more numbers have to go through gates. Additionally, in some runs, the noise simulation does not have more than 1 average bit-flip error for any of its operations (table 2). In comparison to the real hardware runs, on the VBE adder, the average of the BFEs on noise simulation is 0.561, almost half of the average (0.962) from the real hardware.

Table 2. Table of average bit-flip errors for the VBE adder on real hardware.

a	b	c	Average BFE
0	1	0	0.649
1	0	0	0.659
0	0	1	0.685
0	0	0	0.712
0	3	0	0.848

1	2	0	0.886
1	0	1	0.889
3	0	0	0.895
3	2	0	0.896
1	1	0	0.900
1	1	1	0.911
2	2	0	0.920
0	1	1	0.921
2	3	0	0.932
0	2	1	0.937
2	1	0	0.958
0	2	0	0.986
2	0	1	0.995
2	0	0	1.042
2	2	1	1.047
1	3	1	1.051
3	1	1	1.088
0	3	1	1.097
3	0	1	1.098
1	2	1	1.104
3	3	0	1.110
2	1	1	1.116
3	3	1	1.118
2	3	1	1.124
3	2	1	1.148
1	3	0	1.196
3	1	0	1.241

Table 3. Table of average bit-flip errors for the VBE adder on noise simulation.

a	b	c	Average BFE
0	0	0	0.331
1	0	0	0.413
2	2	0	0.422
3	3	0	0.433
0	1	1	0.446
3	2	0	0.447
0	1	0	0.459
3	1	1	0.475
0	0	1	0.482
2	0	0	0.485
2	2	1	0.485
2	3	0	0.488
3	3	1	0.495
1	0	1	0.499
2	1	0	0.501
0	2	1	0.507
0	2	0	0.512
1	2	0	0.517
3	2	1	0.531
1	1	0	0.533
2	0	1	0.549
2	1	1	0.555
2	3	1	0.559
1	1	1	0.579
0	3	1	0.592
0	3	0	0.593
3	1	0	0.598
1	3	0	0.609
3	0	0	0.622
1	3	1	0.666
1	2	1	0.675
3	0	1	0.703

Conclusion

Overall, our experiment is successful in its attempt to explore the variations between different adders and the capabilities of the noise model to emulate the real hardware. Running our algorithms gives us a plethora of data, principally histograms of bit-flip errors from the results of each of the 32 addition operations. Using this data, we can compare the rates of error for both real hardware and noise simulation in both of the adders. One of the intriguing trends that emerges is that of the relative performances of the VBE and the CDKM adders. After multiple runs, the VBE adder always outperforms the CDKM adder on IBM's QPUs, as it almost certainly has the highest chance of getting the right answer; meanwhile, the CDKM adder struggles with a few operations for which it would have the highest probability of getting an erroneous result. However, both adders have high rates of errors that would incorrectly add numbers together on the real hardware. The VBE adder's success over the CDKM adder is surprising, as the VBE adder had more Toffoli gates which theoretically should have added more complexity and therefore more noise in comparison to the CDKM adder. Further, comparing the real hardware to the noise simulation, the adders with simulated noise always outperform the adders operating on real hardware; although this outcome might seem positive, the noise simulation often does not closely emulate the real hardware's results. This is due to the fact that the noise model used does not include sufficient sources of noise to account for the real hardware's error tendencies.

While VBE and CDKM are both reversible implementations of otherwise classical adders, some future studies could similarly test quantum adder algorithms, such as the Draper QFT adder. Comparing how QFT adders perform on real hardware compared to the adders evaluated in this paper could lead to interesting findings relevant to applications in quantum factorization algorithms.

Acknowledgements

The author would like to thank and acknowledge Dr. Anthony Hoffman, an Associate Professor in the Department of Electrical Engineering at the University of Notre Dame, for his exceptional mentorship, guidance, and support throughout this research project. Further, the author acknowledges his gratitude to his parents for their immense support during late nights spent writing.

References

Adder (electronics). (2022, June 27). In *Wikipedia*. https://en.wikipedia.org/wiki/Adder_(electronics)

Cuccaro, S. A., Draper, T. G., Kutin, S. A., & Moulton, D. P. (2004). A new quantum ripple-carry addition circuit. *Quantum Physics*. https://doi.org/10.48550/arXiv.quant-ph/0410184

Draper, T. G., Kutin, S. A., Rains, E. M., & Svore, K. M. (2004). A logarithmic-depth quantum carry-lookahead adder. *Quantum Physics*. https://doi.org/10.48550/arXiv.quant-ph/0406142

Ladd, T. D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C., & O'Brien, J. L. (2010). Quantum computers. *Nature*, *464*, 45-53. https://doi.org/10.1038/nature08812

Obenland, K. M. & Despain, A. M. (1998). A parallel quantum computer simulator. *Quantum Physics*. https://doi.org/10.48550/arXiv.quant-ph/9804039

Qiskit Development Team. (2022). *Noise models*. Qiskit. https://qiskit.org/documentation/apidoc/aer_noise.html

Vedral, V., Barenco, A., & Ekert, A. (1995). Quantum networks for elementary arithmetic operations. *Quantum Physics*. https://doi.org/10.1103/PhysRevA.54.147

Yu, N., Duan, R., & Ying, M. (2013). Five two-qubit gates are necessary for implementing the Toffoli gate. *Physical Review A*, *88*(1), 010304. https://doi.org/10.1103/PhysRevA.88.010304