# 3 Qubit Quantum Adders

Kacy Zhou

School of Computer Science, University of Sydney
kzho4399@uni.sydney.edu.au

August 8, 2021

**Abstract**

*This report will focus on investigating different implementations of 3 qubit quantum adder circuits by running them on IBM's quantum computing infrastructure, Qiskit, to find the most accurate implementation, as well as see the present limitations of quantum computing.*

## I. Introduction

Adders are a fundamental piece of logic in classical computers, with basic addition allowing more complicated operations such as table indexing and address calculation. Because it is an operation with many uses, this makes it a suitable operation to investigate in quantum computing. However, quantum computing is currently limited by various forms of noise, and therefore this report will investigate how this limits the accuracy of in and out of place 3 qubit quantum adders implemented with and without CCCNOT gates.

## II. Background

### i. Classical Adders

Classical adders involve binary digits represented by bits, and logic circuits that depend on the particular implementation involved. Gates used to compute this would be the AND gate, which gives an output of 1 if both inputs are 1, that can be used to represent a carry onto the next digit, and an XOR gate, which outputs 1 if its inputs are different (e.g 0 and 1 or 1 and 0), which would represent the sum at that digit. This is shown in Figure 1.

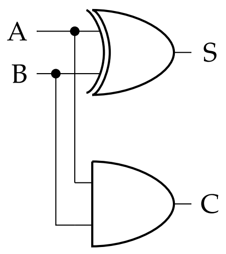


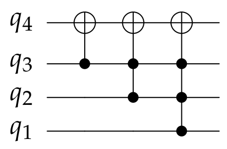
**Figure 1:** *Example half adder, taking two bits of input A and B, using the XOR gate to calculate the sum (S) and the AND gate to calculate the carry (C)*

### ii. Quantum Adders

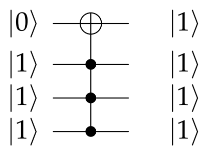
While classical and quantum adders perform the same operation, the nature and action of gates available to quantum circuits are different. Operations are done on qubits ($|0\rangle$, $|1\rangle$) instead of bits (0, 1). While qubits can have a superposition of the $|0\rangle$ and $|1\rangle$ states ($\alpha|0\rangle + \beta|1\rangle$, $\alpha, \beta \in \mathbb{C}$), for the purposes of this report we can disregard this for simplicity as we will only be working in the basis states $|0\rangle$ and $|1\rangle$. Therefore, we can still use such states to represent binary digits upon measuring them.

Instead of using AND or XOR gates, to create a quantum adder we now consider using the CNOT (Controlled NOT) gate and its derivatives, the Toffoli (CCNOT) and CCCNOT gates.

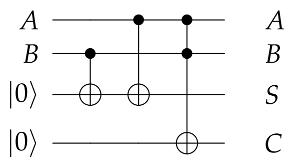
The effect of the NOT quantum gate is to turn the superposition $\alpha|0\rangle + \beta|1\rangle$ to $\beta|0\rangle + \alpha|1\rangle$ at the target qubit [Nielsen and Chuang, 2010] (p.18), however since we are not dealing with superpositions here, this effectively turns $|0\rangle$ to $|1\rangle$ and vice versa. The controlled not gates will only do this if all their control qubits have the value $|1\rangle$, as can be seen in Figure 2b. Therefore the CNOT gate has one control qubit, the Toffoli (CCNOT) gate has two control qubits, and the CCCNOT gate has three control qubits. In Figure 2a, each row represents a qubit, and each connected vertical line represents a single gate (CNOT/Toffoli/CCCNOT).



**(a)** *Left to right: CNOT, Toffoli, CCCNOT gates on qubits $q_1$ to $q_4$*



**(b)** *CCCNOT gate flips the topmost qubit from $|0\rangle$ to $|1\rangle$*



**(c)** *Quantum version of the classical half adder in Figure 1*

**Figure 2:** *Quantum circuits explaining the usage of controlled NOT gates*

## iii. Present Limitations

Present-day quantum computers are limited by noise, which results in a chance of returning an incorrect value upon measurement, due to interactions from outside influences [Nielsen and Chuang, 2010] (p. 353). This error is compounded when using more qubits, more gates, and more complicated gates. For the gates used in adder circuits, this error increases as the number of control bits increases, with CNOT gates being the least error-prone.

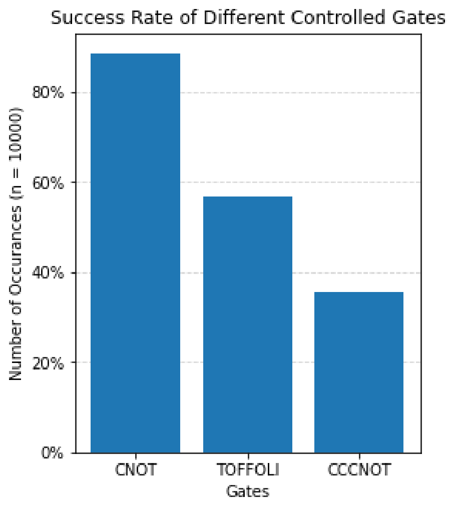


**Figure 3:** *Graph showing the percentage of times that the correct output was produced by IBM's FakeMelbourne with increasing numbers of control qubits (left to right)*

This is partly due to the fact that the representation of a Toffoli is given by at least 6 CNOTs [Shende and Markov, 2008], whilst a CCCNOT gate can be represented by 2 Toffolis and an ancillary (extra) qubit. Therefore each successive gate with an extra control qubit is at least as error-prone as the last.

## III. Methodology

This report uses Qiskit and IBM's Quantum Computers to investigate the error rates in 3 qubit quantum adders. 3 qubit adders were chosen so that the results would not be exceedingly error-prone or trivial. Because IBM does not have a free quantum computer with access to more than 5 qubits, the noise model from a previous quantum computer was used (FakeMelbourne) to simulate one. For simplicity, no incoming carry or outgoing carry was considered to reduce the number of qubits involved, and in place and out of place adders were both investigated to see how the different circuits would change the outcomes of success. All code for this project can be found at:

`https://github.com/kacy-zh/2021-Quantum-Winter-Adders`

## IV. Replacing Toffoli With CNOTs

One investigation of interest was finding out whether Toffolis could be replaced with less error-prone CNOTs. As mentioned and seen in Figure 3, using Toffolis in a quantum circuit would increase the error rates, and since they are used to calculate the carry, it is important to investigate whether the number of Toffolis in an adder can be reduced. Therefore, it was worthwhile investigating whether a Toffoli could be replaced with some combination of CNOT gates and ancillae.

This was done by generating all the combinations of CNOT gates which acted over 3 separate qubits (see Figure 4), and then finding all unique permutations of these gates using the algorithm described by Figure 5 to check whether any had the same matrix as a Toffoli.

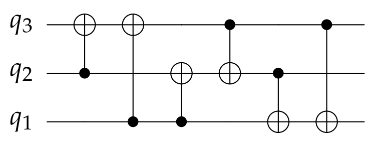


**Figure 4:** *All 6 different CNOT gates actionable on 3 qubits, which are representable by matrices.*

With no ancilla, 168 unique combinations were found but none could replace the action of the Toffoli. Adding further ancillae was not possible as the program crashed due to an increase in the number of possible CNOT combinations and an increase in the size of the

```
MAPPING ← Array mapping the indices 1-6 to the matrix representation of each CNOT gate
CHECKED ← Empty list
TOFFOLI ← Matrix that represents the Toffoli gate

UNCHECKED ← List of the numbers 1 to 6, representing all the initial single CNOT gates

while UNCHECKED is not empty do
  CURRENT ← Next sequence popped off UNCHECKED
  CURRENT_MATRIX ← The matrix represented by CURRENT
  if CURRENT_MATRIX is equal to TOFFOLI then
    print CURRENT
  else if CURRENT_MATRIX is not present in CHECKED then
    Add CURRENT to CHECKED
    Add all permutations of CURRENT with 1-6 to UNCHECKED
  end if
end while
```

**Figure 5:** *Pseudocode for the generator to find all unique permutations, does not include any optimisation.*

matrices.

Further research should be done to investigate optimising this program to work with more ancillae, and constructing a mathematical proof for this.

## V. 3 Qubit In Place Adder

Two circuits were constructed, one using a CCCNOT gate and the other without. This was done to investigate the impact of the CCCNOT gate on the success rate of the circuit, to be defined as the percentage of times the correct outcome of adding the two numbers was returned.

### i. With CCCNOT

This circuit represents summing $a_0a_1a_2 + b_0b_1b_2 \rightarrow b_0b_1b_2$, where $x_2$ is the MSB. Sums require the last 3 CNOTs, which add the same qubit to each other, carries are done by the Toffolis, and the CCCNOT gate carries from the least significant bit to the most significant bit when necessary.

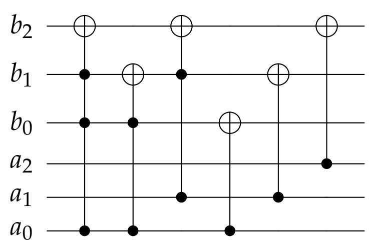


**Figure 6:** *3 Qubit in place adder summing $a_0a_1a_2 + b_0b_1b_2 \rightarrow b_0b_1b_2$, where $x_2$ is the MSB. This uses 1 CCCNOT, 2 Toffolis, and 3 CNOTs.*

### ii. Without CCCNOT

This set of gates adds an extra ancilla to remove the CCCNOT in Figure 6 by using 3 Toffolis in its place. This circuit also cleans up the ancilla, returning it to the $|0\rangle$ state.

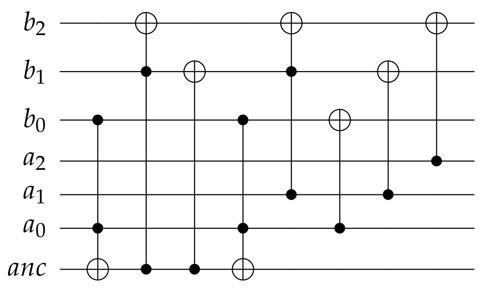


**Figure 7:** *3 Qubit in place adder with ancilla replacing CCCNOT, using 4 Toffolis and 4 CNOTs.*

### iii. Comparison of the Two Circuits

Figure 8 shows the success percentages of both circuits sorted by their Hamming weights. The average success rate for the circuit with the CCCNOT (Figure 6) was $12 \pm 5\%$, and without (Figure 7) was $14 \pm 5\%$, with the error being denoted by the standard deviation of all results.

100,000 was chosen as the number of iterations per input because an increase to 200,000 did not change the results significantly.

For 2 of the with CCCNOT inputs, a wrong outcome was the most frequent outcome. The percentage difference in success rates between the two was less than 25%, showing that the error was not major, especially since it only occurred in 2 inputs out of 64. Notably, none of the ancillae ended up in the $|1\rangle$ state as the most probable outcome, showing that the clean-up was mostly successful.

| **Input** | **Expected** | **Actual** |
|---|---|---|
| 100011 | 100111 | 000010 |
| 100101 | 100011 | 000001 |

**Table 1:** *Table detailing the wrong outcomes returned for the circuit in Figure 8.*

Of these two states, all the returned outcomes had fewer $|1\rangle$ states than the expected outcome, showing that the $|1\rangle$ state was more unstable. These wrong outcomes changed on each run of the code, indicating that this error was random.

Additionally, the Hamming weight tended to influence the outcome of the circuit, with inputs of low Hamming weights on the right side being more successful than those with higher Hamming weights on the right. Again, his is because of the instability of the $|1\rangle$ state in comparison to the $|0\rangle$ state, since it has a higher energy. Because of this, any calculation which begins in a state of high Hamming weight will be more susceptible to error (right side of Figure 8).

Overall, the success rates between each implementation appear to be quite similar, with the circuit without CCCNOTs appearing to be slightly less error-prone. However, with the standard deviations of the average success rate overlapping greatly, it cannot be said that this difference is significant.

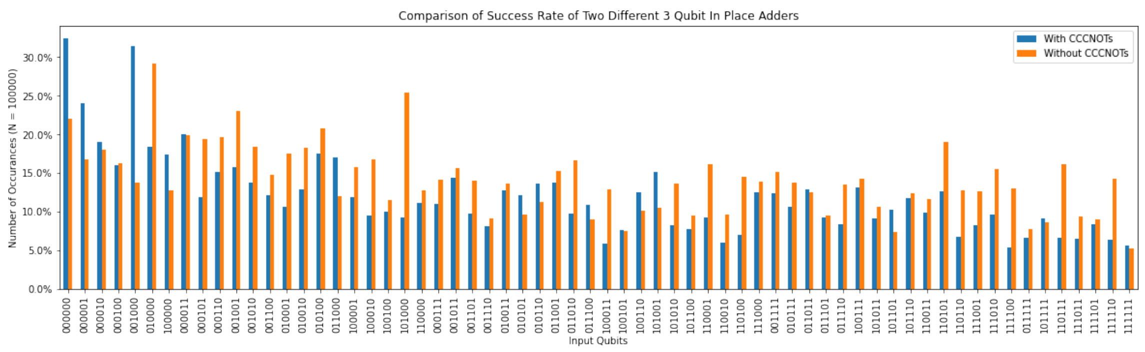


**Figure 8:** *Graph showing success rates of the circuits in Figure 6 (blue) and Figure 7 (orange) having the correct output value for the input given (bottom to top, $a_0a_1a_2b_0b_1b_2$)*

## VI. 3 Qubit Out of Place Adder

### i. With CCCNOTs

This circuit represents summing $a_0a_1a_2 + b_0b_1b_2 \rightarrow s_0s_1s_2$, where $x_2$ is the MSB. In a similar fashion to the in place adder, CNOTs are used to sum individual qubits, and any carries are then done by the Toffolis or CCCNOTs.

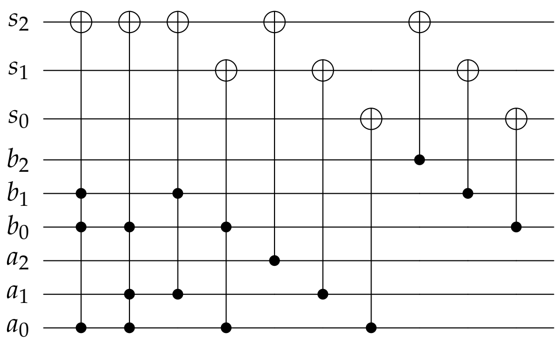


**Figure 9:** *3 Qubit out of place adder summing $a_0a_1a_2 + b_0b_1b_2 \rightarrow s_0s_1s_2$ with 2 CCCNOTs, 2 Toffolis, and 6 CNOTs.*

### ii. Without CCCNOTs

The $s_1$ qubit can be used to reduce the number of CCCNOT gates present in Figure 9, by utilising it to compute the first carry.

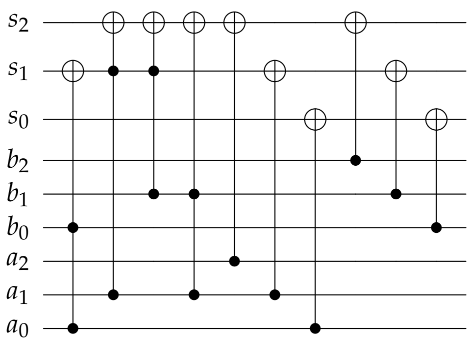


**Figure 10:** *3 Qubit out of place adder summing $a_0a_1a_2 + b_0b_1b_2 \rightarrow s_0s_1s_2$ replacing CCCNOTs with Toffolis. This circuit has 4 Toffolis and 6 CNOTs.*

### iii. Comparison of the Two Circuits

The average success rate for Figure 9 was $3 \pm 2\%$, and for Figure 10 it was $7 \pm 3\%$, with all results shown in Figure 11. These success rates have decreased in comparison to the in place adders, which had rates of $12 \pm 5\%$, and $14 \pm 5\%$ respectively. This is likely due to the increased number of qubits needed in the circuit to compute the out of place sum. There also is a larger difference in success percentages between the two out of place adders. Since both circuits have the same number of qubits and the same number of gates, the greater error rate can be mainly attributed to the usage of CCCNOTs.

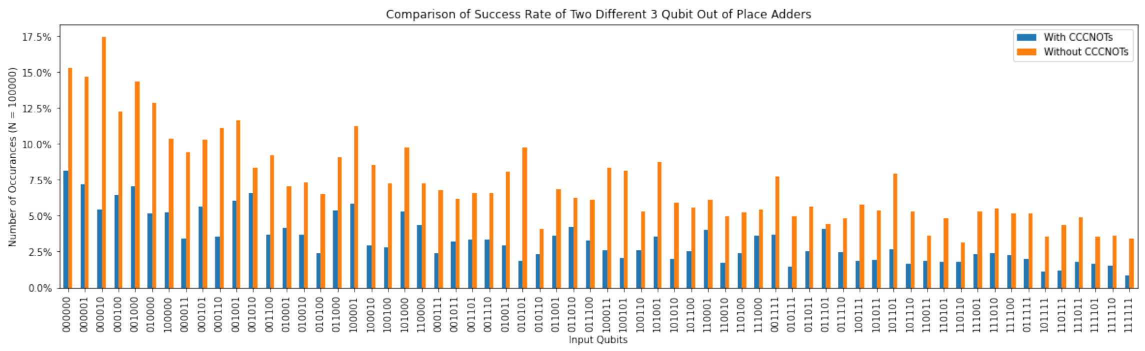


**Figure 11:** *Graph showing success rates of circuits Figure 9 (blue) and Figure 10 (orange) having the correct output value for the input given (bottom to top, $a_0a_1a_2b_0b_1b_2$)*

In the CCCNOT circuit, around half the time, the most frequent result returned was not the correct answer, indicating the presence of random errors being more frequent to the point of being overpowering. These also often had fewer $|0\rangle$ states than the answer, which is hypothesised to be due error propagation from the CCCNOTs, especially since they are placed at the start of my circuit.

Again, with increasing Hamming weights for the input states, the success rate tends to decrease. Additionally, this tends to be affected by the output Hamming weight as well. This is especially noticeable in the CCCNOT circuit, with additions such as 7 (111) + 7 (111) starting (111111000) and ending (111111011) on a high Hamming weight. Further investigation should be done to explicitly characterise this trend.

## VII. In Place vs Out of Place Adders

| Adder Type | Success Rate |
|---|---|
| In Place w/CCCNOT | $12\% \pm 5\%$ |
| In Place w/o CCCNOT | $14\% \pm 5\%$ |
| Out of Place w/ CCCNOT | $3\% \pm 2\%$ |
| Out of Place w/o CCCNOT | $7\% \pm 3\%$ |

**Table 2:** *Table summarising the success percentages of different in and out of place adders.*

From these results, the most accurate adder to use would be the in place adder without CCCNOTs (Figure 7), having the best average success rate of $14 \pm 5\%$. However, with in place adders needing to modify one of the existing numbers, if this is not doable, then the out of place adder without CCCNOTs (Figure 10) should be used instead ($7 \pm 3\%$).

Further investigations include finding whether error is introduced in the measurement or in the circuit, optimising the order of gates and to ensure greater accuracy, and comparing the accuracy of these circuits to those of other implementations of a 3 qubit adder, such as in [Cuccaro, Draper, Kutin, and Moulton, 2004]. Methods to improve fault tolerance and correct for errors should also be investigated, although many of these require the usage of additional qubits, such as the Shor code [Nielsen and Chuang, 2010] (p. 432), and so are unfeasible in the present day.

## VIII. Conclusion

3 main items were found to affect the success rates of these 3 qubit adders: whether there were more or less qubits involved in the implementation, replacing the controlled gates with those of fewer control qubits, and the Hamming weights of the input and output numbers. For future use, operations should be integrated into toolchains to select the optimal implementation dependent on the user's needs.

## References

[Cuccaro, Draper, Kutin, and Moulton, 2004] Cuccaro, S. A., Draper, T. G., Kutin, S. A, and Moulton, D. P. (2004). A new quantum ripple-carry addition circuit. *quant-ph/0410184*

[Nielsen and Chuang, 2010] Nielsen, M. A. and Chuang, I. L. (2010). Quantum Computation and Quantum Information, 10th Anniversary Edition.

[Shende and Markov, 2008] Shende, V. V. and Markov, I. L. (2008). Quantum Computation and Quantum Information, 10th Anniversary Edition. *Quant.Inf.Comp*, 9(5-6):461-486