2019

T-COUNT OPTIMIZATION OF QUANTUM CARRY LOOK-AHEAD ADDER

Vladislav Ivanovich Khalus
University of Kentucky, khalus.vlad@gmail.com
Digital Object Identifier: https://doi.org/10.13023/etd.2019.227

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Vladislav Ivanovich Khalus, Student

Dr. Himanshu Thapliyal, Major Professor

Dr. Aaron Cramer, Director of Graduate Studies

T-COUNT OPTIMIZATION OF QUANTUM CARRY LOOK-AHEAD ADDER

THESIS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
in the College of Engineering
at the University of Kentucky

By
Vladislav Ivanovich Khalus
Lexington, Kentucky
Director: Dr. Himanshu Thapliyal, Assistant Professor of Electrical and Computer Engineering
Lexington, Kentucky
2019

ABSTRACT OF THESIS

T-COUNT OPTIMIZATION OF QUANTUM CARRY LOOK-AHEAD ADDER

With the emergence of quantum physics and computer science in the 20^{th} century, a new field was born which can solve very difficult problems in a much faster rate or problems that classical computing just can't solve. In the 21^{st} century, quantum computing needs to be used to solve tough problems in engineering, business, medical, and other fields that required results not today but yesterday. To make this dream come true, engineers in the semiconductor industry need to make the quantum circuits a reality.

To realize quantum circuits and make them scalable, they need to be fault tolerant, therefore Clifford+T gates need to be implemented into those circuits. But the main issue is that in the Clifford+T gate set, T gates are expensive to implement.

Carry Look-Ahead addition circuits have caught the interest of researchers because the number of gate layers encountered by a given qubit in the circuit (or the circuit's depth) is logarithmic in terms of the input size n. Therefore, this thesis focuses on optimizing previous designs of out-of-place and in-place Carry Look-Ahead Adders to decrease the T-count, sum of all T and T Hermitian transpose gates in a quantum circuit.

KEYWORDS: Quantum Computing, Fault Tolerant, T-count, Out-of-place Carry Look-Ahead Adder, In-place Carry Look-Ahead Adder

Vladislav Ivanovich Khalus

May 28, 2019

T-COUNT OPTIMIZATION OF
QUANTUM CARRY LOOK-AHEAD ADDER

By

Vladislav Ivanovich Khalus

Dr. Himanshu Thapliyal

(Director of Thesis)

Dr. Aaron Cramer

(Director of Graduate Studies)

May 28, 2019

(Date)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Quantum computing offers a lot of promises like performing tasks that are very computational and a unique capability when processing information [1]. Many algorithms were proposed to solve difficult problems in communication, computing, and sensing [1]. To make these quantum algorithms a reality, quantum hardware devices or more specifically quantum circuits need to be made to perform quantum logic, carrying out an order of elaborate calculations, and performing quantum information encoding [1].

In 1960, R Landuaer proposed that KTln2 J of energy is wasted on a single bit of data, k is defined as Boltzmann's Constant, T is defined as the temperature [2]. Thirteen years later, researcher named Bennet said that to avoid kTln2 J of energy loss in a circuit, the circuit has to be a reversible one. [2].

The way to make a reversible circuit is by using reversible gates. Reversible gates have a one-to-one mapping between inputs and outputs [2]. The good thing is that the circuit will not waste energy or information and the inputs could be recovered from outputs [2]. For quantum computing, its necessary for quantum gates to be reversible.

To improve more of the efficiency of quantum computing, the quantum gates employ superposition and entanglement, this greatly speeds up the computation [3].

Instead of having one bit of information being computed, multiple bits of can easily be computed with quantum computing and its' gates.

To make the quantum circuits even better, meaning reliable and scalable, they need to be fault tolerant [4]. For the quantum circuits to be fault tolerant, they need to omit noise errors [5]. Quantum gates that are fault tolerant have Clifford+T gates. Clifford+T gates include: NOT, Hadamard, T, T Hermitian transpose, Phase, Phase Hermitian transpose, and CNOT gates [5]. With the help of these gates and quantum correcting error codes, the quantum circuit can omit noise errors [5]. The main issue in implementing a fault tolerant quantum circuit is that T gate is expensive to implement, therefore the T gate needs to be implemented when necessary [6]. The reason why T gate is costly is costly to implement in practice compared to Clifford gates is that the gate isn't transversal for a lot of quantum error-correcting codes [6], therefore T gate need to be implemented when necessary.

When going into fault-tolerant computing, quantum gates that aren't in the Clifford group are difficult to make. In many quantum error correcting codes like the surface code in [7], just one T gate needs around 100 times circuit volume than compared to an H gate or a CNOT gate [8]. Therefore, the production cost is a big issue in fault tolerant computing. This is also a big problem in quantum computing because the T gate is a universal quantum gate, which is used everywhere in quantum computing and its algorithms. Therefore, to make quantum computing reliable and scalable, this issue needs to be resolved.

Because T gate is expensive to implement, reducing T-count became one of the necessary optimizing goals in literature [5]. Where T-count is defined as number of T and T Hermitian transpose gates in a quantum circuit [6].

Quantum circuits that implement arithmetic operations are a necessity for realizing quantum algorithms [4]. Examples of quantum algorithms are Peter Shor's factoring algorithm [4], Grover's search algorithm [9], Quantum Fourier Transform

[10], Simon's algorithm [11], and triangle finding algorithm [4].

Many arithmetic operations in quantum circuits like binary addition [12], binary and BCD addition [13], floating-point addition [14], adder-subtractor and subtractor [15], integer multiplication [16], modular multiplication [17], floating-point multiplication [18], and integer division [4] have been given attention from the researchers working on quantum computing. But the most important out of these arithmetic operations is addition because quantum adders form key components in subtraction, multiplication, and division.

Carry Look-Ahead addition circuits have caught the interest of researchers because the number of gate layers encountered by a given qubit in the circuit (or the circuit's depth) is logarithmic in terms of the input size n. As a result, designs of Quantum Carry Look-Ahead Adders (QCLA) have been proposed in literature in [19][20][21][22][23]. For example, designs in [19][20][23] are interesting but they suffer from high T gate cost. To overcome the limitations of the existing designs, this thesis presents quantum circuits for Carry Look-Ahead addition based on Clifford+T gates.

1.1 Contribution of Thesis

This thesis introduces integer arithmetic designs that will be used in quantum computing. Main focus will be on Quantum Carry Look-Ahead Adders (QCLAs):

1. Quantum out-of-place CLA (low T-count).
2. Quantum out-of-place CLA (high speed).
3. Quantum in-place CLA (low T-count).
4. Quantum in-place CLA (high speed).
5. Quantum in-place CLA (high speed + low T-count).

1.2 Outline of Thesis

This thesis will be broken down into five main Chapters. Chapter 2 gives a background of a quantum bit, quantum gates, the logical-AND computation and uncomputation gates and an overview of the classical CLA and a literature review of the existing QCLAs. Then Chapter 3 introduces new design for the out-of-place QCLA with a low T-count and a faster version of it. Next, Chapter 4 goes into the new design for the in-place QCLA, which has three versions: the low T-count, the high speed, and a mixture of both. Chapter 5 shows the simulation results for the out-of-place and in-place QCLAs. Finally, the thesis will end with a Conclusion in Chapter 6.

Chapter 2

Background

This chapter introduces the basics of quantum computing and the reversible quantum gates. It is highly recommended to get an understanding of the quantum world before moving on to the other Chapters of this thesis.

2.1 Quantum bit

To begin with quantum computing, one needs to know a quantum bit, or a qubit.

2.1.1 Introduction

The qubit is basically a vector in a two-dimensional complex vector space, $\mathbb{C}^2$ [24]. A generic vector $|\psi\rangle$ in $\mathbb{C}^2$ can be written as $|\psi\rangle = \alpha|c_1\rangle + \beta|c_2\rangle$ [25]. The vector can be written by using a symbol called a ket, $|\psi\rangle$. This notation for qubits is called Dirac bra-ket notation [24], and will be used throughout quantum information processing [3].

Going back to $|\psi\rangle = \alpha|c_1\rangle + \beta|c_2\rangle$, the $|\psi\rangle$ is called a state, α and β are amplitudes that are complex numbers [25]. The vectors $|c_1\rangle$ and $|c_2\rangle$ are defined as $|c_1\rangle = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, $|c_2\rangle = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$. In ket form, $|0\rangle = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $|1\rangle = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$. Therefore, $|c_1\rangle = |0\rangle$ and $|c_2\rangle = |1\rangle$.

With $|\psi\rangle = \alpha|c_1\rangle + \beta|c_2\rangle$, this means that the qubit is in superposition, where its being in both states $|0\rangle$ and $|1\rangle$ [3], not just $|0\rangle$ or $|1\rangle$. The state $|\psi\rangle$ can also be called a linear combination of states [24], which is the actual definition of superposition. With the help of superosition, large amount of classical bits can be processed by running the quantum computer once.

One thing to note, a qubit doesn't start out in superposition, it is usually in spin-up state: $|0\rangle$ or spin-down state: $|1\rangle$, which correspond to classical bits 0 and 1. But when an operation is performed on a qubit like a qubit going through the quantum gate, things get interesting, the qubit ends up in superposition or in a multi-qubit system, the qubits get entangled.

2.1.2 Operations on Qubits

Before going further in understanding quantum gates, one needs to know how they are used with qubits. There are two forms that can be used to minipulate qubits and gates, these two forms are matrix and Dirac bra-ket notation. For example, the matrix is $\left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$ and the equivalent bra-ket notation will be $|0\rangle\langle 0| + |1\rangle\langle 1|$ [3]. The bra form $\langle\psi|$ or the combination of the ket and the bra forms: $|\psi_1\rangle\langle\psi_2|$ will not be used in this thesis because that signifies a gate, and a gate is better understood in matrix form. As for the ket form $|\psi\rangle$, it signifies a qubit and qubit can be used interchangeably in the ket form or in the matrix form.

As previously explained, the states for a 1-qubit: $|0\rangle = \left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right]$ and $|1\rangle = \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]$. Which shows that just one qubit has possible states of $\{|0\rangle, |1\rangle\}$, which are called computational basis states [24]. Going into the 2-qubit quantum system, there are four computational basis states of $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, and the state can be written as $|\psi\rangle = c_{0,0}|00\rangle + c_{0,1}|01\rangle + c_{1,0}|10\rangle + c_{1,1}|11\rangle$ [24]. In the matrix form, the computational basis states look like:

$|00\rangle$ in matrix: $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ $|01\rangle$ in matrix: $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ $|10\rangle$ in matrix: $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ $|11\rangle$ in matrix: $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$.

Lastly, when the two qubits go through a gate, the operation looks like this:

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \bullet \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}.$$

Going into the 3-qubit quantum system, there are eight computational basis states: $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$, The state can be written as $|\psi\rangle = c_{0,0,0}|000\rangle + c_{0,0,1}|001\rangle + c_{0,1,0}|010\rangle + c_{0,1,1}|011\rangle + ... c_{1,1,1}|111\rangle$ [24]. In the matrix form, the computational basis states look like this:

$|000\rangle$: $\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$ $|001\rangle$: $\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}$ $|010\rangle$: $\begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$ $|011\rangle$: $\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix}$ $|100\rangle$: $\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix}$ $|101\rangle$: $\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}$

$|110\rangle$: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $|111\rangle$: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$.

When three qubits go through a gate, the operation looks like this:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Using qubits as matrices gets tedious, so Dirac bra-ket notation or more precisely the ket form $|\psi\rangle$ is used to simplify things. There is a Dirac bra-ket notation for the gates but it will be omitted for simplicity. Therefore, when there is a operation between the quantum gate and the qubit(s), then the operation can be written as X$|1\rangle$ or CNOT$|11\rangle$ or the operation can say that a CNOT gate was applied on qubits one and three.

2.1.3 Entanglement

Entangled states are crucial pieces in quantum computation [24]. They are unique quantum phenomena, because of non-existent classical counterpart [3]. When going into a multi-qubit system, most states are entangled. The use of entanglement is important in quantum teleportation, Bell's inequality, and superdense coding [24].

An entangled state has a property that there does not exist two 1-qubit states, $|c_1\rangle$ and $|c_2\rangle$ that $|\psi\rangle = |c_1\rangle|c_2\rangle$. For an entangled state $|\psi\rangle \neq |c_1\rangle|c_2\rangle$ because the product of two states cannot equal an entangled state, $|\psi\rangle$ [24]. With the number of qubits increasing, an exponential growth of the complexity of entangled states is expected.

One way to make an ordinary state entangled, there has to be an interaction between the qubits by using gates. If the quantum system is 2-qubit system, then a 2-qubit gate is required to entangled the qubits, with the help of the CNOT gate, this can be achieved [25].

An example of non-entangled state is $|\psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$, which was extracted from the product $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. But an example of an entangled state is $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, which is also one of Bell states that is used for quantum teleporation [24].

To understand entanglement clearly, there is a state $|\psi_1\rangle = |01\rangle$, the first qubit which is 0 signifies night, and the second qubit which is 1 signifies the stars can be seen, these two qubits are entangled or they cannot be separated from each other because one will not function without the other. If its day time, the stars cannot be seen and the new state will be $|\psi_2\rangle = |10\rangle$. Therefore, the stars seen or not seen qubit is entangled to the night or day qubit.

2.1.4 Measurement Gate

Just like in a classical computer when the user wants to retrieve the content in the memory, a quantum computer can do the same. The measurement gate as shown in Figure 2.1 is the only way to get the information about the qubit [3]. Using the measurement gate, one classical bit of information can be retrieved from the qubit [3]. This means that one needs to be careful where to put the measurement gate because it restricts the amount of information that can be retrieved from the qubit [3]. Another restriction of the measurement circuit is that a state cannot be cloned, and one of the cloned states measured [3], once the state is measured, the measurement yields a classical bit of information [3]. As shown in Figure 2.1, the single line is the qubit and the double line is the classical bit.

With the qubit being in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, when measuring, the result for the classical bit of 0 will have a probability of $|\alpha|^2$ and the result for the classical bit of 1 will have a probability of $|\beta|^2$, which means $|\alpha|^2 + |\beta|^2$ equals a probability of 1 [24].

Lastly, Combining Entanglement and Measuring, when two qubits are entangled, and one of the qubits are measured, then the measuring will let the user know of the other qubit that wasn't measured, even though the non-measured qubit was untouched.

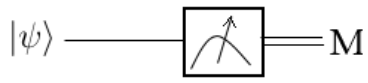


Figure 2.1: The Measurement gate

2.2 Quantum Gates

Before making any quantum circuit, like the Carry Look-Ahead Adder, an understanding of quantum gates needs to be made. One thing to note that in quantum computing, all the gates are reversible. In this thesis, only the quantum gates that built the quantum circuits in Chapters 4 and 5 will be discussed. These and other quantum gates can be found in [24].

2.2.1 1-qubit Gates

Starting with the 1-qubit Clifford+T gates shown more descriptly and clearly in [26] than in [24], they consist of one input and one output. Figures 2.2(a), 2.2(b), 2.2(c), 2.2(d), 2.2(e), 2.2(f) show these gates that will be used to make a fault tolerant quantum circuit. Also, looking at Table 2.1, it shows all the information for these gates and what happens when a qubit goes through them.

2.2.2 2-qubit Gates

CNOT gate/Feynman Gate

The Controlled NOT gate or CNOT gate for short is a 2x2 reversible gate and a Clifford gate with two inputs: $|A\rangle$, $|B\rangle$ and two outputs: $|P\rangle$ and $|Q\rangle$. The first input equals the first output, therefore $|P\rangle = |A\rangle$. As for the second output, the first input is EXOR'ed with the second input, therefore $|Q\rangle = |A \oplus B\rangle$. The quantum configuration of the CNOT gate can be EXOR-Down or EXOR-Up. This all can be shown in Figures 2.3(a), 2.3(b), 2.3(c).

Controlled-Z Gate

The Controlled-Z gate or CZ gate is a 2x2 reversible gate with two inputs: $|A\rangle$, $|B\rangle$ and two outputs: $|P\rangle$ and $|Q\rangle$. The first input equals the first output, therefore $|P\rangle$

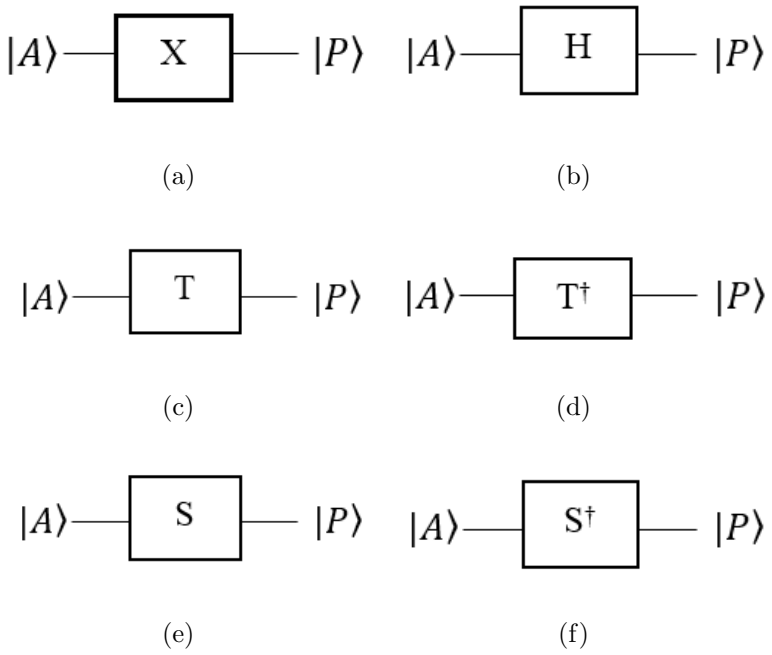


Figure 2.2: The Clifford+T gates for a 1-qubit
(a) The NOT gate
(b) The Hadamard gate
(c) The T gate
(d) The $T^\dagger$ gate
(e) The S gate
(f) The $S^\dagger$ gate

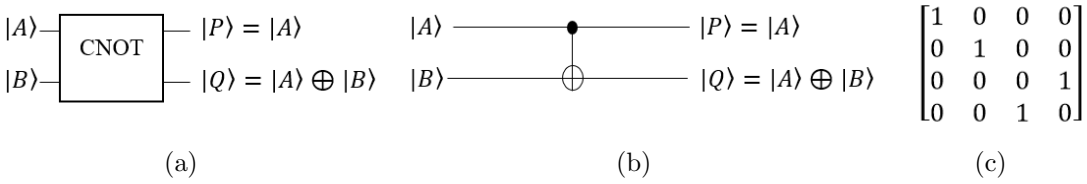


Figure 2.3: The CNOT gate
(a) Graphical Representation
(b) Quantum Representation
(c) Matrix Representation

$= |A\rangle$. As for the second output, $|Q\rangle = (-1)^{A \cdot B}|B\rangle$, which means when both $|A\rangle$ and $|B\rangle$ have a qubit of $|1\rangle$, then $|Q\rangle$ = -$|1\rangle$ else $|Q\rangle = |B\rangle$. The CZ gate can be shown in Figures 2.4(a), 2.4(b), 2.4(c).

Table 2.1: Clifford+T 1-qubit gates

Type of gate	Symbol	Matrix	Input: $\|A\rangle$	Output: $\|P\rangle$
NOT	X or $-\oplus-$	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\|0\rangle$	$\|1\rangle$
			$\|1\rangle$	$\|0\rangle$
Hadamard	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$\|0\rangle$	$\frac{1}{\sqrt{2}}\big(\|0\rangle + \|1\rangle\big)$
			$\|1\rangle$	$\frac{1}{\sqrt{2}}\big(\|0\rangle$ - $\|1\rangle\big)$
T gate	T	$\begin{bmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{bmatrix}$	$\|0\rangle$	$\|0\rangle$
			$\|1\rangle$	$e^{\frac{i\pi}{4}}\|1\rangle$
T gate Hermitian Transpose	$T^\dagger$	$\begin{bmatrix}1 & 0\\ 0 & e^{\frac{-i\pi}{4}}\end{bmatrix}$	$\|0\rangle$	$\|0\rangle$
			$\|1\rangle$	$e^{\frac{-i\pi}{4}}\|1\rangle$
Phase	S	$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$	$\|0\rangle$	$\|0\rangle$
			$\|1\rangle$	$i\|1\rangle$
Phase Hermitian Transpose	$S^\dagger$	$\begin{bmatrix}1 & 0\\ 0 & -i\end{bmatrix}$	$\|0\rangle$	$\|0\rangle$
			$\|1\rangle$	$-i\|1\rangle$

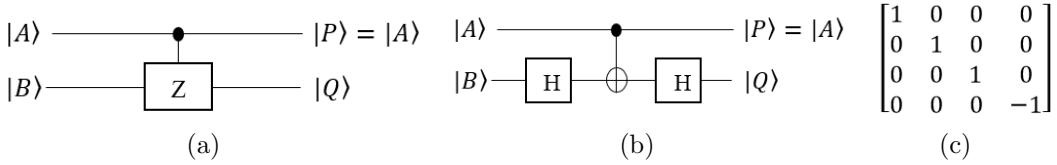


Figure 2.4: The Controlled-Z gate
(a) Quantum Representation
(b) Clifford+T Representation
(c) Matrix Representation

2.2.3 3-qubit Gates

Toffoli Gate:

The Toffoli gate or the Controlled-Controlled NOT gate as shown in Figures 2.5(a), 2.5(b), 2.5(c), 2.5(d) is a 3x3 reversible gate with three inputs: $|A\rangle$, $|B\rangle$, $|C\rangle$ and three outputs: $|P\rangle$, $|Q\rangle$, $|R\rangle$. The first input equals the first output, therefore $|P\rangle = |A\rangle$. The same is true for the second output, meaning $|Q\rangle = |B\rangle$. The last output takes the first, second inputs and ANDs them together, and finally the AND operation gets EXOR'ed by the third output, therefore $|R\rangle = |A\&B\rangle \oplus |C\rangle$. The Toffoli gate can be in three configurations EXOR-Down, EXOR-Up or EXOR-Middle. The Clifford+T implentation shown in Figure 2.5(c) was taken from [24].

2.3 Logical-AND Computation and Uncomputation Gates

One of the last steps before making the main quantum circuits in Chapters 4 and 5 is understanding the three gates: logical-AND computation gate, logical-AND uncomputation Measure-and-Fixup gate, and logical-AND uncomputation Computation-Reversal gate, which were all discussed by Gidney in [27].

2.3.1 Logical-AND Computation Gate

Many addition circuits require the AND of two qubits, this gate is a great implementation inside of them. As it can be seen in Figures 2.6(a) and 2.6(b), the two inputs are the two qubits that are of interest, and the third input is the ancilla that is set at $|T\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}\ |1\rangle\big)$, which was extracted from one Hadamard and one T gate. Talking about the T-count, this gate has two (one from ancilla and one from the gate) T gates and two $T^\dagger$ gates; therefore the T-count is four.

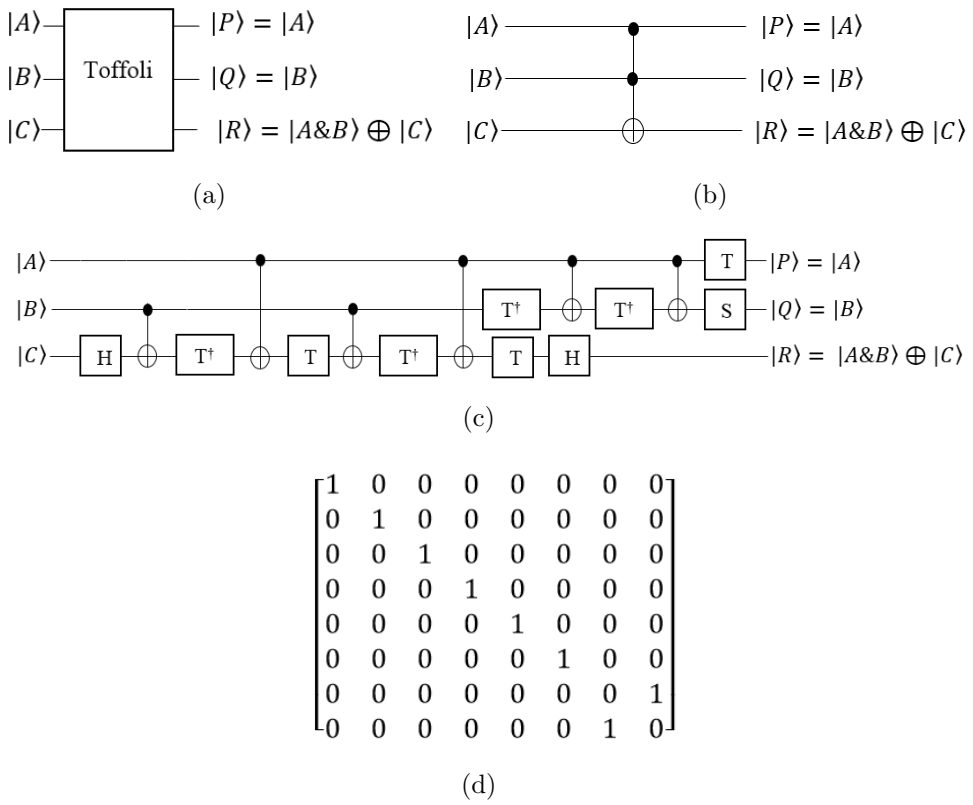


Figure 2.5: The Toffoli gate
(a) Graphical Representation
(b) Quantum Representation
(c) Clifford+T Representation
(d) Matrix Representation

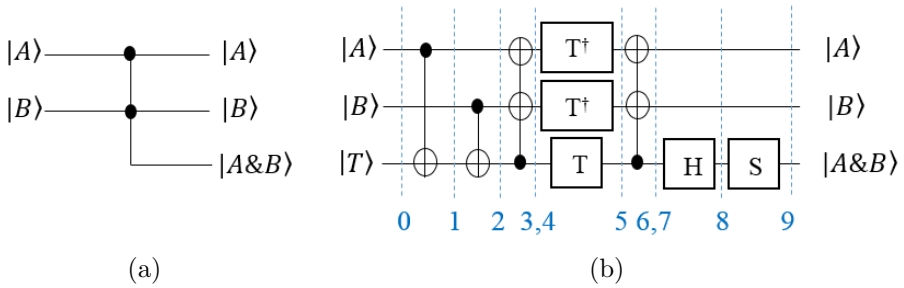


Figure 2.6: The logical-AND computation gate
(a) Graphical Representation
(b) Quantum Representation

The logical-AND computation gate was divided into four sections to prove that the logical-AND of two qubits work. Looking at Table 2.2, it shows all possible combinations of $|A\rangle$ and $|B\rangle$.

Table 2.2: Truth Table of logical-AND

$\|A\rangle$	$\|B\rangle$	$\|A\&B\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$

Starting with the first case: $|A\rangle = |0\rangle$**,** $|B\rangle = |0\rangle$

0. Initial values:

   $|0\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}\,|1\rangle\big) \therefore |0\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + (\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i)|1\rangle\big) \therefore |0\rangle|0\rangle\frac{1}{\sqrt{2}}|0\rangle + |0\rangle|0\rangle\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i)|1\rangle \therefore \frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)|0\rangle|0\rangle|1\rangle$

1. Applying the CNOT gate on the first qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)|0\rangle|0\rangle|1\rangle$

2. Applying the CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)|0\rangle|0\rangle|1\rangle$

3. Applying another CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)|0\rangle|1\rangle|1\rangle$

4. Applying the CNOT gate on the first and third qubit:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)|1\rangle|1\rangle|1\rangle$

5. Applying the $T^{\dagger}$ gates on the first and second qubits and T gate on the third qubit:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + (\frac{1}{2} + \frac{1}{2}i)(\frac{1}{2} - \frac{1}{2}i)(\frac{1}{2} - \frac{1}{2}i)(\frac{1}{2} + \frac{1}{2}i)|1\rangle|1\rangle|1\rangle \therefore \frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle|1\rangle$

6. Applying the CNOT gate on qubits two and three:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle$

7. Applying the CNOT gate on first and third qubits:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|0\rangle|0\rangle|1\rangle$

8. Applying Hadamard gate on qubit three:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) + \frac{1}{\sqrt{2}}|0\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \therefore \frac{1}{2}|0\rangle|0\rangle|0\rangle + \frac{1}{2}|0\rangle|0\rangle|1\rangle + \frac{1}{2}|0\rangle|0\rangle|0\rangle$
   $- \frac{1}{2}|0\rangle|0\rangle|1\rangle \therefore 1|0\rangle|0\rangle|0\rangle$

9. Applying the S gate on the third qubit:

   $1|0\rangle|0\rangle|0\rangle$

Finally, measuring the value $(1)^2 = 1 \therefore 100\%$ probability on $|0\rangle|0\rangle|0\rangle$

Going into the second case: $|A\rangle = |1\rangle$**,** $|B\rangle = |0\rangle$

0. Initial values:

   $|1\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big) \therefore |1\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + (\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i)|1\rangle\big) \therefore |1\rangle|0\rangle\frac{1}{\sqrt{2}}|0\rangle + |1\rangle|0\rangle\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}}$
   $+ \frac{1}{\sqrt{2}}i\big)|1\rangle \therefore \frac{1}{\sqrt{2}}|1\rangle|0\rangle|0\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|1\rangle$

1. Applying the CNOT gate on the first qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|0\rangle$

2. Applying the CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|0\rangle$

3. Applying another CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|0\rangle$

4. Applying the CNOT gate on the first and third qubit:

   $\frac{1}{\sqrt{2}}|0\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|0\rangle$

5. Applying the $T^\dagger$ gates on the first and second qubits and T gate on the third qubit:

   $\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\big)\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)\big(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|0\rangle \therefore \frac{1}{\sqrt{2}}|0\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle|0\rangle$

6. Applying the CNOT gate on qubits two and three:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle|0\rangle$

7. Applying the CNOT gate on first and third qubits:

   $\frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle|0\rangle$

8. Applying Hadamard gate on qubit three:

   $\frac{1}{\sqrt{2}}|1\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) + \frac{1}{\sqrt{2}}|1\rangle|0\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \therefore \frac{1}{2}|1\rangle|0\rangle|0\rangle - \frac{1}{2}|1\rangle|0\rangle|1\rangle + \frac{1}{2}|1\rangle|0\rangle|0\rangle + \frac{1}{2}|1\rangle|0\rangle|1\rangle \therefore 1|1\rangle|0\rangle|0\rangle$

9. Applying the S gate on the third qubit:

   $1|1\rangle|0\rangle|0\rangle$

Finally, measuring the value $(1)^2 = 1 \therefore 100\%$ probability on $|1\rangle|0\rangle|0\rangle$

Going into the third case: $|A\rangle = |0\rangle$, $|B\rangle = |1\rangle$

0. Initial values:

   $|0\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big) \therefore |0\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + (\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i)|1\rangle\big) \therefore |0\rangle|1\rangle\frac{1}{\sqrt{2}}|0\rangle + |0\rangle|1\rangle\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle \therefore \frac{1}{\sqrt{2}}|0\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|1\rangle$

1. Applying the CNOT gate on the first qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|1\rangle$

2. Applying the CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|0\rangle$

3. Applying another CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|0\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|0\rangle$

4. Applying the CNOT gate on the first and third qubit:

   $\frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|0\rangle$

5. Applying the $T^\dagger$ gates on the first and second qubits and T gate on the third qubit:

   $\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\big)\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)\big(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\big)|0\rangle|1\rangle|0\rangle \therefore \frac{1}{\sqrt{2}}|1\rangle|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|0\rangle|1\rangle|0\rangle$

6. Applying the CNOT gate on qubits two and three:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}|0\rangle|1\rangle|0\rangle$

7. Applying the CNOT gate on first and third qubits:

   $\frac{1}{\sqrt{2}}|0\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}|0\rangle|1\rangle|0\rangle$

8. Applying Hadamard gate on qubit three:

   $\frac{1}{\sqrt{2}}|0\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) + \frac{1}{\sqrt{2}}|0\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \therefore \frac{1}{2}|0\rangle|1\rangle|0\rangle - \frac{1}{2}|0\rangle|1\rangle|1\rangle + \frac{1}{2}|0\rangle|1\rangle|0\rangle + \frac{1}{2}|0\rangle|1\rangle|1\rangle \therefore 1|0\rangle|1\rangle|0\rangle$

9. Applying the S gate on the third qubit:

   $1|0\rangle|1\rangle|0\rangle$

Finally, measuring the value $(1)^2 = 1 \therefore 100\%$ probability on $|0\rangle|1\rangle|0\rangle$

Going into the last case: $|A\rangle = |1\rangle$, $|B\rangle = |1\rangle$

0. Initial values:

   $|1\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big) \therefore |1\rangle|1\rangle\frac{1}{\sqrt{2}}\big(|0\rangle + (\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i)|1\rangle\big) \therefore |1\rangle|1\rangle\frac{1}{\sqrt{2}}|0\rangle + |1\rangle|1\rangle\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle \therefore \frac{1}{\sqrt{2}}|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\big)|1\rangle|1\rangle|1\rangle$

1. Applying the CNOT gate on the first qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|1\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)|1\rangle|1\rangle|0\rangle$

2. Applying the CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)|1\rangle|1\rangle|1\rangle$

3. Applying another CNOT gate on the second qubit and the third:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)|1\rangle|0\rangle|1\rangle$

4. Applying the CNOT gate on the first and third qubit:

   $\frac{1}{\sqrt{2}}|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)|0\rangle|0\rangle|1\rangle$

5. Applying the $T^\dagger$ gates on the first and second qubits and T gate on the third qubit:

   $\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\right)\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\right)|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)|0\rangle|0\rangle|1\rangle \therefore \frac{1}{\sqrt{2}}(-i)|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}i|0\rangle|0\rangle|1\rangle$

6. Applying the CNOT gate on qubits two and three:

   $\frac{1}{\sqrt{2}}(-i)|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}i|0\rangle|1\rangle|1\rangle$

7. Applying the CNOT gate on first and third qubits:

   $\frac{1}{\sqrt{2}}(-i)|1\rangle|1\rangle|0\rangle + \frac{1}{\sqrt{2}}i|1\rangle|1\rangle|1\rangle$

8. Applying Hadamard gate on qubit three:

   $\frac{1}{\sqrt{2}}(-i)|1\rangle|1\rangle\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}i|1\rangle|1\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \therefore \frac{-i}{2}|1\rangle|1\rangle|0\rangle - \frac{-i}{2}|1\rangle|1\rangle|1\rangle + \frac{i}{2}|1\rangle|1\rangle|0\rangle - \frac{i}{2}|1\rangle|1\rangle|1\rangle \therefore -i|1\rangle|1\rangle|1\rangle$

9. Applying the S gate on the third qubit:

   $-i|1\rangle|1\rangle(-i)|1\rangle \therefore 1|1\rangle|1\rangle|1\rangle$

Finally, measuring the value $(1)^2 = 1 \therefore 100\%$ probability on $|1\rangle|1\rangle|1\rangle$

2.3.2 Logical-AND Uncomputation Gate

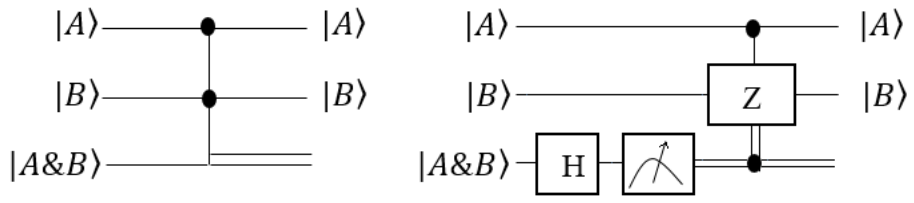
Measure-and-Fixup Method

Talking about the T-count, this gate has no T and $T^\dagger$ gates; therefore the T-count is 0. A reversible implementation of the logical-AND function takes two inputs, classical binary 1 and 0, which are represented as qubits, and at the end of computation, the two original inputs will be returned as classical values. Since classical 0s and 1s don't have phase angles, we need to remove them. The purpose of this uncomputation gate is phase correction. As it can be seen from the third qubit or the ancilla in one of the Figures like Figure 2.8(a), the measurement of the ancilla is collapsed from superposition to a classical bit. If the ancilla value wants to be restored to $|T\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}\ |1\rangle\big)$ for another computation, the classical bit needs to be cleared and a Hadamard with the T gate needs to be applied.

Instead of writing all the cases by hand, a quantum circuit simulator called Quirk in [28] was used to verify the correct operation of the Measure-and-Fixup method. Four possible test cases were used (See Figures 2.8(a), 2.8(b), 2.8(c), and 2.8(d)) to show the simulation results, but also most importantly, the two top qubits didn't change. There are four steps for the gates in 2.8(a), 2.8(b), 2.8(c), and 2.8(d); the first step is the ancilla or the third qubit is set, then the inputs (first and second qubits) are adjusted to the four possible test cases, next the qubits go through the logical-AND computation gate and then the Measure-and-Fixup gate, and finally the qubits are measured.

One thing to note that if the uncomputation gate is applied in the middle of a quantum circuit, more specifically meaning that if a measure is placed in the middle of any quantum circuit, quantum properties will be lost, and the quantum circuit will be slower. The gate can be shown in 2.7(a) and 2.7(b). From the name, Measure means that its measuring the third qubit which is in a superposition, and Fixup means that

the controlled-Z gate is fixing the phase errors on the first two qubits.



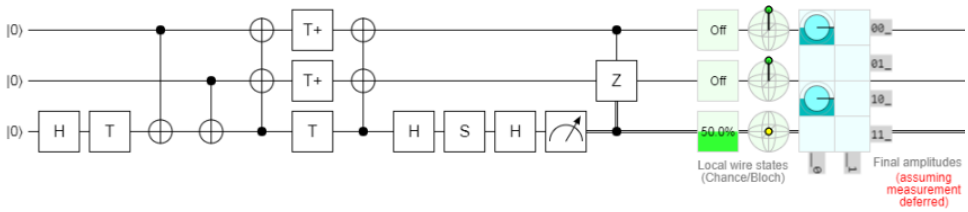
(a) (b)

Figure 2.7: The logical-AND uncomputation gate: Measure-and-Fixup
(a) Graphical Representation
(b) Quantum Representation

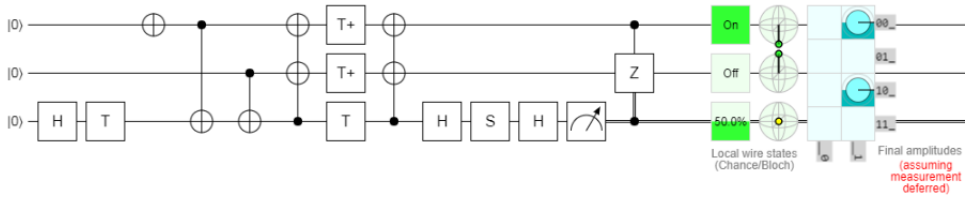
Computation-Reversal Method

An alternative option to the Measure-and-Fixup method is reversing the logical-AND computation gate as shown in Figure 2.9(a) and 2.9(b). Talking about the T-count, this gate has two T gates and one $T^\dagger$ gate; therefore the T-count is three. Unlike the Measure-and-Fixup method which returns a classical bit for the third output, the Computation-Reversal method returns the ancilla value of $|T\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}\ |1\rangle\big)$.

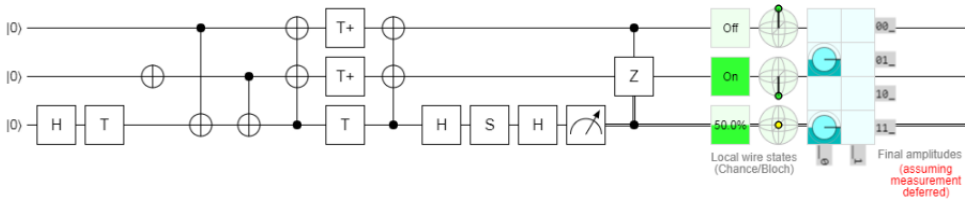
Instead of writing all the cases by hand, a quantum circuit simulator called Quirk in [28] was used to verify the correct operation of the Computation-Reversal method. As it can be seen in 2.10(a), the original ancilla value of $|T\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}\ |1\rangle\big)$. Four possible test cases were used (See Figures 2.10(b), 2.10(c), 2.10(d), and 2.10(e)) to verify that in the end, the ancilla stayed at its original superpostion value but also most importantly, the two top qubits didn't change. There are five steps for the gates in 2.10(b), 2.10(c), 2.10(d), and 2.10(e); the first step is the ancilla or the third qubit is set, then the inputs (first and second qubits) are adjusted to the four possible test cases, next the qubits go through the logical-AND computation gate and then the Computation-Reversal gate, and finally the qubits are measured.



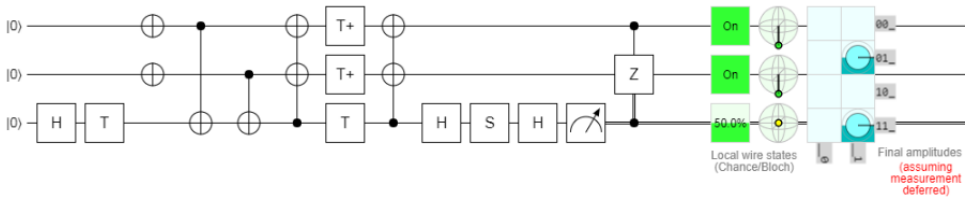
(a)



(b)



(c)

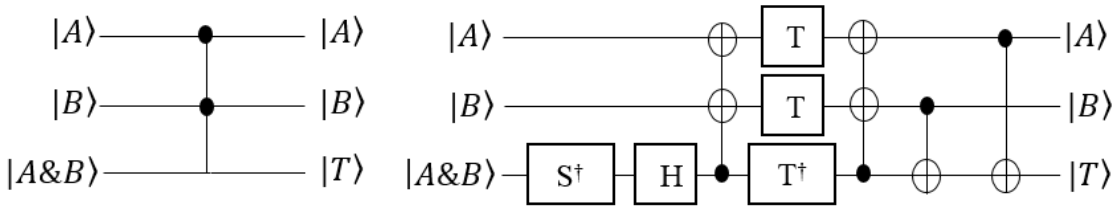


(d)

Figure 2.8: Testing Measure-and-Fixup gate
(a) Both inputs are $|0\rangle$
(b) The first input is a $|1\rangle$
(c) The second input is a $|1\rangle$
(d) Both inputs are a $|1\rangle$

2.3.3 Comparison between the two Methods

When considering these two gates, the Measure-and-Fixup uncomputation gate will be resource efficient when implemented with error correcting codes such as surface codes. Quantum circuits based on the gates of Computation-Reversal uncomputation will be resource efficient in cases where error correcting codes cannot or are not used

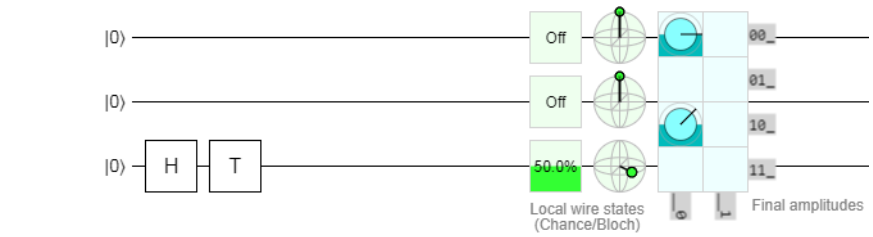


(a) (b)

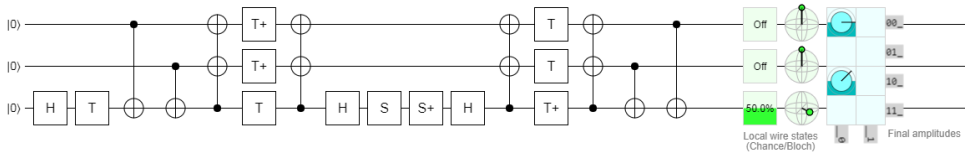
Figure 2.9: The logical-AND uncomputation gate: Computation-Reversal
(a) Graphical Representation
(b) Quantum Representation

such as with near term devices. Each gate will now be illustrated to show the best suited option for their respective implementations.

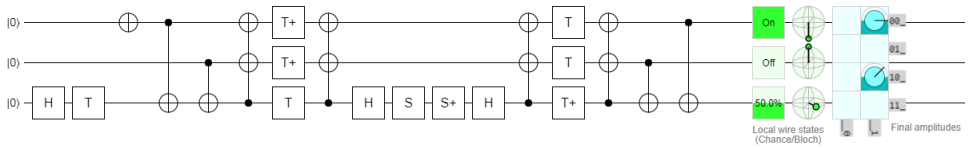
When implemented with error correcting codes such as the surface codes, the logical-AND computation gate saves resources because it uses an ancilla $|A\rangle$ set to $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$ as opposed to a logical T gate. To realize a logical T gate, ancilla set to $|A\rangle$ and $|Y\rangle$ (where $Y = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{i\pi}{2}}|1\rangle\big)$ must be created with one or more rounds of state distillation [29] [30]. According to [31] [29] [30] $|A\rangle$ distillation requires at least 15 logical qubits, 15 T gates, 15 measurements and arrays of CNOT gates and $|Y\rangle$ distillation requires at least 7 logical qubits, 7 S gates, 7 measuremnts and arrays of CNOT gates. If the surface code scheme in [31] is used, 3600 physical qubits are required per logical qubit. As a result, generating a $|Y\rangle$ state will need 25200 qubits and generating a $|A\rangle$ state will need 54000 total qubits. Thus, the logical-AND gate saves qubit and quantum gates. The uncomputation gate with Measure-and-Fixup does not require T gates avoiding the costly $|A\rangle$ and $|Y\rangle$ state distillations. Considering that a single logical T gate requires at least 22 measurements, the penalties associated with an additional single measurement in the uncomputation gate with Measure-and-Fixup will be negligible and offset by the overall resource savings from avoiding logical T gates.



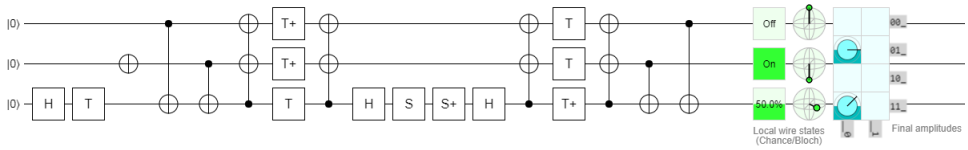
(a)



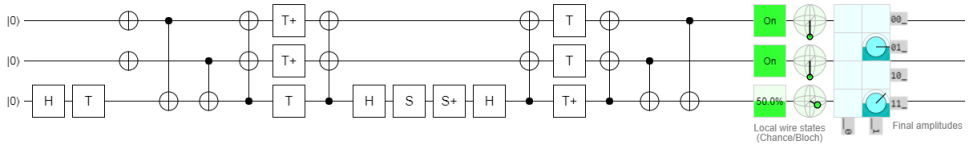
(b)



(c)



(d)



(e)

Figure 2.10: Testing Computation-Reversal gate
(a) Ancilla $|T\rangle$ result
(b) Both inputs are $|0\rangle$
(c) The first input is a $|1\rangle$
(d) The second input is a $|1\rangle$
(e) Both inputs are a $|1\rangle$

For situations where error correcting codes are not used such as with near-term quantum technologies where error correcting schemes (such as the surface codes) are not supported, the ancilla $|A\rangle$ used in the logical-AND computation gate is realized by applying a Hadamard gate then a T gate to an ancillae set to 0. The uncomputation

gate with Computation-Reversal has the advantage in terms of computation speed in cases where fault tolerant schemes are not used. Time of computation is important because qubits can only maintain superposition states for a finite time or rather they are limited by their coherence times. Coherence times for quantum computers have been reported in the literature. For instance, the IBM quantum machine has a coherence time of $100\mu s$ according to [32]. Coherence times of up to a half hour have been reported in the literature [33]. Thus, to maximize the amount of computations, time intensive operations should be used sparingly. Operation times for gates and the measurement operation have been reported in the literature [34] [35] [32]. For example, in [34], a one qubit gate (such as a T gate) has a computation time of $1\mu s$, a two qubit gate (such as a CNOT) has a computation time of $10\mu s$ while measurement has a computation time of $200\mu s$. To estimate the computation time for each gates in Figures 2.6(b), 2.7(b), and 2.9(b), a three step algorithm is used: (i) calculate the number of circuit layers (or depth), (ii) calculate maximum time to perform each gate layer, (iii) sum the results. The values in [34] for the computation time estimates. The logical-AND computation gate and the Computation-Reversal uncomputation gate will both have a depth of 7 and a total computation time of $43\mu s$. The Measure-and-Fixup uncomputation gate has a depth of 4 and a total computation time of $212\mu s$. The Computation-Reversal uncomputation gate is roughly 5 times faster and therefore will permit one to perform roughly 5 times as much computation within a given coherence time.

2.4 Carry Look-Ahead Addition Review

2.4.1 Classical CLA

The Full Adder is used to add two 1-bit numbers, or inputs, and a Carry in if applicable. The results of the Full Adder are two outputs: Sum and Carry out. To add

two 2-bit, 3-bit, or n-bit numbers, the Full Adders have to be in parallel [36], where the Carry out of the first value goes into the Carry in of the second value, and so on. The parallel addition of the Full Adder is called a Ripple Carry Adder [36].

The problem with Ripple Carry Adder is to get the next Sum and the next Carry out, the previous Carry out needs to be known. That means the Carry out values needs to move or propagate all the way to the last Sum and Carry out to have the right value. Therefore, Carry out propagating to the final value creates propagation delay, which is defined as n-number of Full Adders inside a Ripple Carry times the number of seconds to reach the Sum and Carry out values in each Full Adder [36].

To reduce the propagation delay of the Ripple Carry Adder, there needs to be another way, but it will increase the complexity of the circuit, a circuit called the Carry Look-Ahead Adder. With the complexity of the Carry Look-Ahead Adder, two values have to come into place, carry propagate and carry generate [36]. Carry propagate is defined as: $p_i = a_i \oplus b_i$, which is responsible for propagating Carry to Carry out. Carry generate is defined as $g_i = a_i \,\&\, b_i$, which makes the Carry out, c_{i+1} when both a_i and b_i are set to 1, and it doesn't matter what the Carry in, c_i is. Sum, s_i is defined as: $s_i = a_i \oplus b_i \oplus c_i$, therefore, plugging in p_i: $s_i = p_i \oplus c_i$. Carry out, c_{i+1} is defined as $c_{i+1} = p_i \,\&\, c_i \mid g_i$.

An example is shown to illustrate the Carry Look-Ahead addition of two 4-bit numbers.

Using $c_{i+1} = p_i \,\&\, c_i \mid g_i$:

$i = 0 : c_1 = p_0 \,\&\, c_0 \mid g_0$

$i = 1 : c_2 = p_1 \,\&\, c_1 \mid g_1 = c_2 = p_1 \,\&\, (p_0 \,\&\, c_0 \mid g_0) \mid g_1 = p_1 \,\&\, p_0 \,\&\, c_0 \mid p_1 \,\&\, g_0 \mid g_1.$

$i = 2 : c_3 = p_2 \,\&\, c_2 \mid g_2 = p_2 \,\&\, p_1 \,\&\, p_0 \,\&\, c_0 \mid p_2 \,\&\, p_1 \,\&\, g_0 \mid p_2 \,\&\, g_1 \mid g_2.$

$i = 3 : c_4 = p_3 \,\&\, c_3 \mid g_3 = p_3 \,\&\, p_2 \,\&\, p_1 \,\&\, p_0 \,\&\, c_0 \mid p_3 \,\&\, p_2 \,\&\, p_1 \,\&\, g_0 \mid p_3 \,\&\, p_2 \,\&\, g_1 \mid p_3 \,\&\, g_2 \mid g_3.$

2.4.2 Literature Review

A literature review was done for the Quantum Carry Look-Ahead Adder. In the later Chapters, three literatures in [19], [23], and [23] will be used because they are applicable for the comparison with the proposed work in Chapters 3 and 4. The remaining two articles in [21] and [22] were read to get more of an understanding into the Carry Look-Ahead addition.

For the first literature from [19], the authors that proposed this paper to improve the efficiency of the Quantum Carry Look-Ahead Adder with depth of O(logn) and O(n) of ancillary qubits. This Literature included out-of-place, in-place, and extensions of these two circuits like the comparison and subtraction. The circuit is optimized for ancillas, size, and depth.

The next literature which is from [20], the authors that proposed this paper focused on delay, gate count, and quantum cost of two circuits: out-of-place and in-place Carry Look-Ahead Adders which included reversible gates like the CNOT, Peres, TR, and Toffoli gates. The purpose was to optimize the circuits in these three parameters of delay, gate count, and quantum cost.

The third literature from [21], the authors that proposed it, focused on reducing the qubits to O(n/logn) and making the depth: O(logn) and size O(n) of the Quantum Carry Look-Ahead Adder. This circuit contains only Toffoli gates. The main issue that the author is facing with is decreasing the ancillary qubits but increasing depth and size for the Carry Look-Ahead Adder.

As for the fourth literature from [22], the main promises that this literature gave is reducing the quantum cost, delay, garbage outputs, and the number of gates on the Quantum Carry Look-Ahead Adder.

The final literature from [23], the authors of this literature decided to improve the requirements and performance for the out-of-place and in-place Quantum Carry Look-Ahead Adder using the measurement-based method. Also there is another Quantum

Carry Look-Ahead Adder described called graph-state, where the size and depth was compared to logical qubits to get a good comparison between the new Quantum Carry Look-Ahead Adders and the existing ones.

Chapter 3

Design of Proposed Out-of-place QCLA

Two designs have been proposed, the first proposed circuit called low T-count and a second proposed circuit called high speed. To make things simplistic, the first proposed circuit will be described, and where there is a difference, then the second proposed circuit will be noted down. The proposed (low T-count) out-of-place QCLA circuit is shown in Figure 3.1 for the case of adding two eight-qubit numbers $|a\rangle$ and $|b\rangle$. Out-of-place means that the Sum qubits are generated or realized from the ancilla qubits. At the end of the computation the location A with $|a\rangle$ and location B with $|b\rangle$ will be unchanged. The s_1 through s_n are realized on n ancillas initialized to the value $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$. As for s_0, it's realized on ancilla being initialized to the value of $|0\rangle$. These ancillas are stored in location Z. Another set of ancillas will be stored in location X. They will also be initialized to $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$ and will be used in the P-rounds step. The number of ancillas inside location X is n-w(n)-$\lfloor$logn$\rfloor$, where w(n) is the number of ones in the binary expansion of n [19], where w(n) = n - $\sum_{y=1}^{\infty}\big\lfloor\frac{n}{2^y}\big\rfloor$. All ancillas except the ancilla used for s_0 are initialized to $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$ because the logical-AND computation gates used in the proposed circuit require these ancillas

to function correctly. At the end of computation, n-w(n)-⌊logn⌋ ancillas are turned into classical bits and need to be reset to be used in other computations, this is the first proposed circuit. As for the second proposed circuit (high speed), the ancillas can be reused in later computations because they are restored to the initialized ancilla value (this will be described in step 6). Finally as for remaining ancillas that are from location Z, they will be in the sum.

The proposed (low T-count) out-of-place QCLA circuit is based on the NOT gate, the CNOT gate, the Toffoli gate along with the logical-AND computation gate and the logical-AND Measure-and-Fixup uncomputation gate presented in [27]. An algorithm based on the design methodology presented in [19] to implement an out-of-place QCLA from these quantum gates. By using logical-AND computation gates and logical-AND Measure-and-Fixup uncomputation gates, the proposed design methodology saves T gates.

For manipulating values in locations A and B, two 2-D arrays are required, one for propagation, p[*j*,*l*] and one for generation, g[*j*,*k*]. The variables of *i*, *j*, *k*, *l* are the indices that signify the location of the propagating, the generating and the carry value.

3.1 Procedure

The steps in designing the proposed (low T-count) out-of-place QCLA are:

1. For $i = 0$ to n-1, apply the logical-AND computation on A[i], B[i], and Z[i+1] . This will generate three outputs. The first output will be A[i], the second output will be B[i], and the third output will be A[i] & B[i] where the location will be set to g[i, i +1].

2. For $i = 1$ to n-1. At the locations of A[i] and B[i] introduce the CNOT gate in which the same value is maintained at A[i], while at location B[i], changes to B[i] = A[i] $\oplus$ B[i], and that value will be set to a new location p[i, i+1].

3. P-rounds: For the logical-AND computation gate, there are three inputs and three outputs. For the first and last outputs, the inputs pass through. The location for the first input and output is p[j,l], the location for the third input and output is p[l, k]. With the help of the second input, which is an ancilla stored at location X[d], the second output could be calculated. But in all simplicity, the second output is calculated by p[j,l] & p[l,k] and the value is saved to the output location of p[j,k]. The equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, and l for the location of array p is determined by t and m, where t=1 to $\lfloor$logn$\rfloor$-1 and where $1 \leq$ m $< \lfloor$ n/2^t $\rfloor$ which is nested in the t loop. The index d is determined by j+k, the addition of these indices are sorted from the lowest to the highest. The lowest will have an index of 0 and the highest will have $\sum_{y=1}^{\infty} \lfloor \frac{n}{2^y} \rfloor$ - $\lfloor$logn$\rfloor$ - 1.

4. G-rounds: This step includes the Toffoli gates, which have three inputs and three outputs. The locations for the three inputs are g[j, l], p[l, k], and g[l, k] respectively. The first and second outputs are the same as the inputs, so the values just pass through and the locations are unchanged. The third output has an operation done on it, g[l, k] $\oplus$ (g[j, l] & p[l, k]) and the value is saved to the output location of g[j, k]. The equation for the indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, and l for the location of arrays p and g are determined by t and m. Where t=1 to $\lfloor$logn$\rfloor$ and where $0 \leq$ m $< \lfloor$ n/2^t $\rfloor$ which is nested in the t loop.

5. C-rounds: This step includes the Toffoli gates, which have three inputs and three outputs. The locations for the three inputs are g[0, l], p[l,k], and g[l,k] respectively. The first and second outputs are the same as the inputs, so the values just pass through and the locations are unchanged. The third output has a operation done on it, where g[l, k] $\oplus$ (g[0, l] & p[l, k]) and the value is saved to the output location g[0, k]. The equation for $l = 2^t$m and $k = 2^t$m + 2^{t-1}. The indices l and k for the locations of arrays g and p are determined by t and m, where t = $\lfloor$ log(2n/3) $\rfloor$ and decreases to 1 and where $1 \leq$ m $\leq \lfloor$ (n - 2^{t-1})/2^t $\rfloor$ which is nested in the t loop.

6. P-erase-rounds: For the logical-AND Measure-and-Fixup uncomputation gate, there are three inputs and three outputs. For the first and last outputs, the inputs pass through. The location for the first input and output is p[j, l], and the location for the third input and output is p[l, k]. As for the middle input, its location is p[j, k], which has the value stored from the logical-AND, and this value is uncomputed and the output value will result in a classical bit (proposed circuit 1: low T-count). For proposed circuit 2 (high speed), by replacing the Measure-and-Fixup gates to the Computation-Reversal gates, the middle output value will result in the original X[d] ancilla value, and the location of the ancilla value will be set to Xout[d]. For both circuits, the equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, l for the location of array p are determined by t and m. Where t = $\lfloor$ logn $\rfloor$ - 1 and decreases to 1, and where $1 \leq$ m $< \lfloor$ n/2^t $\rfloor$, which is nested in the t loop. The definition for index d is the same as in P-rounds.

7. For i = 1 to n-1, apply the CNOT gate on locations p[i, i +1] and g[0, i] to get s_i. For i = 0, the CNOT gate is applied, the ancilla value of Z is set to Z = B[i] $\oplus$ Z.

8. Finally, for i = 0, s_i is equal to A[i] $\oplus$ Z. As for i = 1 to n-1, p[i, i +1] coming from location B[i] is being XORed by A[i], and the result will return the original b_i value. For i = n, s_i or the last Sum value will be extracted from g[*0*, i].

3.2 T-count

The T-count for the first proposed circuit (low T-count) Carry Look-Ahead Adder shown in Figure 3.1 is $22n - 11w(n) - 11\lfloor logn \rfloor - 7$.

- For the first step, it has n number of logical-AND computation gates, with each having a T-count of 4.
- The second step has a T-count of 0.
- The third step has a T-count of 4(n - w(n) - $\lfloor$ logn $\rfloor$).
- Step 4 has n - w(n) Toffoli gates with a T-count of 7(n - w(n)).
- Step 5 has n - $\lfloor$ logn $\rfloor$ - 1 Toffoli gates with a T-count of 7(n - $\lfloor$ logn $\rfloor$ - 1).
- As for steps 6,7, and 8, the T-count is 0.

Table 3.1 shows the equations for the T-count for the two proposed circuits and the three in literature.

Table 3.1: Equations for Out-of-place T-count

Design	T-count Equation
1	$35n - 21w(n) - 21\lfloor logn \rfloor - 7$
2	$35n - 21w(n) - 21\lfloor logn \rfloor - 7$
3	$35n - 14$
Prop_1	$22n - 11w(n) - 11\lfloor logn \rfloor - 7$
Prop_2	$25n - 14w(n) - 14\lfloor logn \rfloor - 7$

1 is the design in [19]
2 is the design in [23]
3 is the design in [20]
Prop_1 is the design in Figure 3.1
Prop_2 is the design in Figure 3.1. Step 6 replaced with Computation-Reversal gate

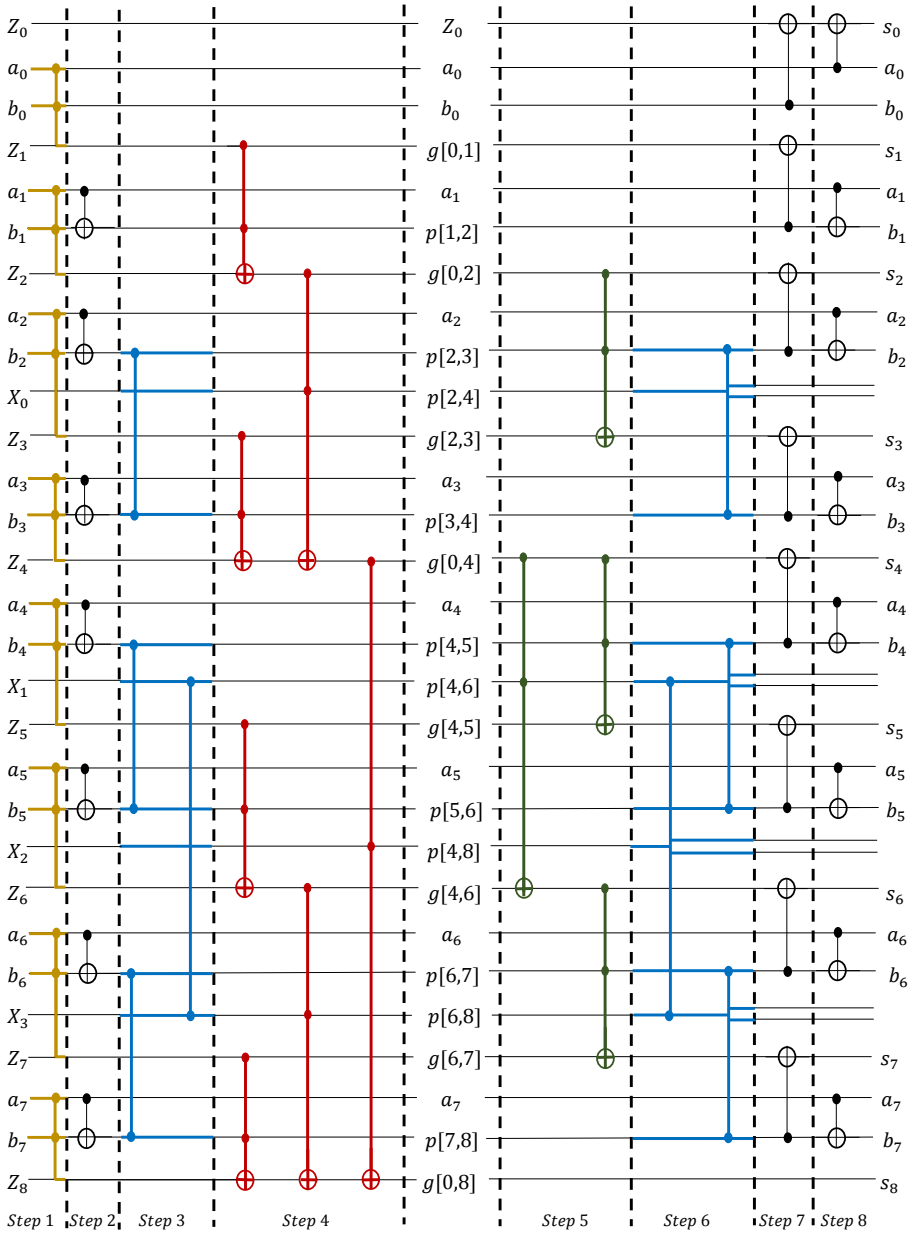


Figure 3.1: Out-of-place QCLA: low T-count
Step 1,3 Computation gate
Step 2,7,8 CNOT gate
Step 4,5 Toffoli gate
Step 6 Uncomputation gate

Chapter 4

Design of Proposed In-place QCLA

Three designs have been proposed, the first proposed circuit called low T-count, the second proposed circuit which is a mixture of low T-count and high speed, and the third proposed circuit called high speed. To make things simplistic, the first proposed circuit will be described, and where there is a difference, then the second and third proposed circuits will be noted. The proposed (low T-count) in-place QCLA circuit is shown in Figure 4.1 for the case of adding two eight-qubit numbers $|a\rangle$ and $|b\rangle$. In-place means that the Sum qubit is generated or realized from an input qubit. At the end of computation, the location A with $|a\rangle$ will be unchanged and the location B with $|b\rangle$ will contain the sum bits s_0 through s_{n-1}. The proposed in-place QCLA circuit also requires n ancillas initialized to the value $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$. These ancillas will be stored in location Z. At the end of computation, n ancillas are turned into classical bits and need to be reset to be used in other computations, this is the first proposed circuit. As for the third proposed circuit (high speed), the ancillas can be reused in later computations and the value is set to the original initialized ancilla. For all the proposed circuits, another set of ancillas will be stored in location X. The ancillas will also be initialized to $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$ and will be used in the P-rounds and the reverse of P-erase-rounds steps. The number of ancillas inside location X is

n-w(n)-$\lfloor \log n \rfloor$, where w(n) is the number of ones in the binary expansion of n [1], where w(n) = n - $\sum_{y=1}^{\infty} \lfloor \frac{n}{2^y} \rfloor$. All ancillas are initialized to the value $\frac{1}{\sqrt{2}}\big(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle\big)$ because the logical-AND computation gates used in the proposed circuit require these ancillas to function properly.

The proposed (low T-count) in-place QCLA circuit is based on the NOT gate, the CNOT gate, the Toffoli gate along with the logical-AND computation and logical-AND Measure-and-Fixup uncomputation gate presented in [27]. An algorithm based on methodology presented in [19] to implement the in-place QCLA from these gates. By using the logical-AND computation and logical-AND Measure-and-Fixup uncomputation gates, the proposed design methodology saves T gates.

4.1 Procedure

The steps in designing the new (low T-count) in-place QCLA are:

1. For $i = 0$ to n-1, apply the logical-AND computation gate on three inputs that have locations A[i], B[i] and Z[i], respectively. This will generate three outputs. The first output will be A[i], the second output will be B[i], where the first and second input value will not change. The third output value will be A[i] & B[i] where the location will be set to g[i, i +1].

2. For $i = 0$ to n-1. At the locations of A[i] and B[i] introduce the CNOT gate in which the same value is maintained at A[i], while at location B[i], changes to A[i] $\oplus$ B[i], and that value will be set to a new location p[i, i+1].

3. P-rounds: For the logical-AND computation gate, there are three inputs and three outputs. For the first and last outputs, the inputs pass through. The

location for the first input and output is p[j,l], and the location for the third input and output is p[l, k]. With the help of the second input, which is an ancilla stored at location X[d], the second output could be calculated. But in all simplicity, the second output has an operation done on it, p[j,l] & p[l,k] and the value is saved to the output location of p[j,k]. The equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, and l for the location of array p are determined by t and m, where t=1 to $\lfloor$logn$\rfloor$-1 and where $1 \leq$ m $< \lfloor$ n/2^t $\rfloor$ which is nested in the t loop. The index d is determined by j+k, the addition of these indices are sorted from the lowest to the highest. The lowest will have an index of 0 and the highest will have $\sum_{y=1}^{\infty} \lfloor \frac{n}{2^y} \rfloor$ - $\lfloor$logn$\rfloor$ - 1.

4. G-rounds: This step includes the Toffoli gates, which have three inputs and three outputs. The locations for the three inputs are g[j, l], p[l, k], and g[l, k] respectively. The first and second outputs are the same as the inputs, so the values just pass through and the locations are unchanged. The third output has an operation done on it, g[l, k] $\oplus$ (g[j, l] & p[l, k]) and the value is saved to the output location of g[j, k]. The equation for $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, l for the location p ang g are determined by t and m. Where t=1 to $\lfloor$logn$\rfloor$ and where $0 \leq$ m $< \lfloor$ n/2^t $\rfloor$ which is nested in the t loop.

5. C-rounds: This step includes the Toffoli gates, which have three inputs and three outputs. The locations for the three inputs are g[0, l], p[l,k], and g[l,k] respectively. The first and second outputs are the same as the inputs, so the values just pass through and the location is unchanged. The third output has a operation done on it, where g[l, k] $\oplus$ (g[0, l] & p[l, k]) and the value is saved to

the output location g[0, k]. The equation for $l = 2^t$m and $k = 2^t$m + 2^{t-1}. The indices l and k for the location of arrays g and p are determined by t and m, where t = $\lfloor$ log(2n/3) $\rfloor$ and decreases to 1 and where $1 \leq$ m $\leq \lfloor$ (n - 2^{t-1})/2^t $\rfloor$ which is nested in the t loop.

6. P-erase-rounds: For the logical-AND Measure-and-Fixup uncomputation gate, there are three inputs and three outputs. For the first and last outputs, the inputs pass through. The location for the first input and output is p[j, l], and the location for the third input and output is p[l, k]. As for the middle input, its location is p[j, k], which has the value stored from logical-AND, and this value is uncomputed and the output value will result in a classical bit (proposed circuit 1). For proposed circuit 3 (high speed), the output value will result in the original X[d] ancilla value, and the location of the ancilla value will be set to Xout[d]. For all the proposed circuits, the equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, l for the location p are determined by t and m. Where t = $\lfloor$ logn $\rfloor$ - 1 and decreases to 1, and where $1 \leq$ m $< \lfloor$ n/2^t $\rfloor$, which is nested in the t loop. The definition for index d is the same as in P-rounds.

7. For i = 1 to n-1, apply the CNOT gate on locations p[i, i +1] and g[0, i] so that location g[0, i] would contain the same value but the value of p[i, i +1] would change to g[0, i] $\oplus$ p[i, i +1] and placed in location p[i, i +1].

8. For i = 0 to n-2, apply the NOT gate. Starting at i = 0, apply the NOT gate on location B[i] and put the value in location p[i, i +1]. As for i = 1 to n-2, apply the NOT gate on the p[i, i +1] location and put that negated value back

in location p[i, i +1].

9. For $i = 1$ to n-2, at locations A[i] and p[i, i +1], apply the CNOT gate, so that location A[i] would contain the same value but the value of location p[i, i +1] changes to A[i] $\oplus$ p[i, i +1], and placed back into the location p[i, i +1].

10. Reverse of P-erase-rounds: which has logical-AND computation gate, there are three inputs and three outputs. For the first and last outputs, the inputs pass through. The location for the first input and output is p[j,l], and the location for the third input and output is p[l,k]. With the help of the second input, which is an ancilla stored at location X[d], the second output could be calculated. But in all simplicity, the second output has an operation done on it, p[j,l] & p[l,k] and the value is saved to the output location p[j,k]. The equation for indices j = 2^tm, k = 2^tm + 2^t, and l = 2^tm + 2^{t-1}. The indices j, k, l for the location p is determined by t and m*, where t=1 to $\lfloor$log(n-1)$\rfloor$-1 and where 1 $\leq$ m < $\lfloor$ (n-1)/2^t $\rfloor$ which is nested in the t loop. The definition for index d is the same as in P-rounds.

11. Reverse of C-rounds: which has Toffoli gates, there are three inputs and three outputs. The locations for the 3 inputs are g[0,l], p[l,k], and g[0,k] respectively. The first and second output values are the same as inputs, so the values just pass through and the location is unchanged. The output has an operation done on it, where g[0, k] $\oplus$ (g[0, l] & p[l, k]) and the value is saved to the output location g[l,k]. The equation for l = 2^tm and k = 2^tm + 2^{t-1}. The indices k and l are determined by t and m*, where t = 1 and goes to t = $\lfloor$ log(2(n-1)/3) $\rfloor$ and where 1 $\leq$ m $\leq$ $\lfloor$ ((n-1)-2^{t-1})/2^t $\rfloor$ which is nested in the t loop.

12. Reverse of G-rounds: which have the Toffoli gates, it includes three inputs and three outputs. The locations for the three inputs are g[j,l], p[l,k] and g[j,k] respectively. The first and second outputs are the same as the inputs, so the values just pass through and the location is unchanged. The third output has an operation done on it, g[j,k] $\oplus$ (g[j,l] & p[l,k]) and the value is saved to the output location of g[l,k]. The equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, l for the location p and g are determined by t and m*. Where t = $\lfloor$ log(n-1) $\rfloor$ which decreases to 1 and where 0 $\leq$ m < $\lfloor$ (n-1)/2^t $\rfloor$ which is nested in the t loop.

13. Reverse of P-rounds: apply the logical-AND Measure-and-Fixup uncomputation gate on three inputs that have locations p[j,l], p[j,k], and p[l,k], respectively. This will generate three outputs. The first output will be p[j,l], the third output will be p[l,k], where the first and third output values will not change. The second input value will be uncomputed, and the output value will result in a classical bit (proposed circuit 1). For proposed circuit 3 (high speed), the output value will result in the original X[d] ancilla value, and the location of the ancilla value will be set to Xout[d]. For all the proposed circuits, the equation for indices $j = 2^t$m, $k = 2^t$m + 2^t, and $l = 2^t$m + 2^{t-1}. The indices j, k, l for the location of array p are determined by t and m*. Where t = $\lfloor$ log(n-1) $\rfloor$ - 1 and decreases to 1, and where 1 $\leq$ m < $\lfloor$ (n-1)/2^t $\rfloor$ which is nested in the t loop. The definition for index d is the same as in P-rounds.

14. For i=1 to n-2, at locations A[i] and p[i, i +1], apply the CNOT gate so that location A[i] would contain the same value but the value of location p[i, i +1]

changes to A[i] $\oplus$ p[i, i +1], and placed back into the location p[i, i +1].

15. For i=0 to n-2, at locations A[i], p[i, i +1], and g[i, i +1], apply the Measure-and-Fixup uncomputation gate so that locations a[i] and p[i, i +1] would contain the same value. The third input value will be uncomputed, and the output value will result in a classical bit (proposed circuit 1). For proposed circuit 3 (high speed), the output value will result in the original Z[i] ancilla value, and the location of the ancilla value will be set to Zout[i].

16. For i=0 to n-2, at location p[i, i +1], apply the NOT gate, so that the value would be inverted and saved in location s_i, where the Sum is stored. For i=n-1, at location p[i, i +1], s_i will be stored from, which is the second last value of Sum. For i = n, at location g[*0*, i], the last value of sum will be stored.

*To preserve the last carry value or s_n, gates and circuits in steps 10 to 13 that do an operation on the last carry value needs to be deleted so they would not interfere with the last carry value. Therefore the equations for t and m were adjusted from the index ending at n to n-1.

Two more proposed Carry Look-Ahead Adders can be extracted by manipulating the design steps. The second proposed (low T-count + high speed) circuit can be made by replacing the Computation-Reversal gates in steps 13 and 15 from Figure 4.2 and putting the Measure-and-Fixup gates. The third proposed (high speed) Carry Look-Ahead Adder is shown in Figure 4.2, where it can be made by replacing all the uncomputation gates to be Computation-Reversal gates.

4.2 T-count

The T-count for the proposed (low T-count) In-place QCLA shown in Figure 4.1 is $40n - 11w(n) - 11\lfloor logn \rfloor - 11w(n-1) - 11\lfloor log(n-1) \rfloor - 32$.

- For the first step, it has n number of logical-AND computation gates, with each having a T-count of 4.
- The second step has a T-count of 0.
- The third step has a T-count of 4(n - w(n) - ⌊ logn ⌋).
- Step 4 has n - w(n) Toffoli gates with a T-count of 7(n - w(n)).
- Step 5 has n - ⌊ logn ⌋ - 1 Toffoli gates with a T-count of 7(n - ⌊ logn ⌋ - 1).
- As for steps 6,7, 8, 9 the T-count is 0.
- Step 10 has 4((n-1) - w(n-1) - ⌊ log(n-1) ⌋)
- Step 11 has (n-1) - ⌊ log(n-1) ⌋ - 1 Toffoli gates with a T-count of 7((n-1) - ⌊ log(n-1) ⌋ - 1).
- Step 12 has (n-1) - w(n-1) Toffoli gates with a T-count of 7((n-1) - w(n-1)).
- As for Steps 13, 14, 15, 16, the T-count is 0.

Table 4.1 shows the equations for the T-count for the three proposed circuits and the three existing works.

Table 4.1: Equations for In-place T-count

Design	T-count Equation
1	$70n - 21w(n) - 21\lfloor log(n) \rfloor - 21w(n-1) - 21\lfloor log(n-1) \rfloor - 49$
2	$70n - 21w(n) - 21\lfloor log(n) \rfloor - 21w(n-1) - 21\lfloor log(n-1) \rfloor - 49$
3	$\frac{203}{4}n - 28$
Prop_1	$40n - 11w(n) - 11\lfloor logn \rfloor - 11w(n-1) - 11\lfloor log(n-1) \rfloor - 32$
Prop_2	$43n - 14w(n) - 14\lfloor logn \rfloor - 11w(n-1) - 11\lfloor log(n-1) \rfloor - 32$
Prop_3	$49n - 14w(n) - 11\lfloor logn \rfloor - 14w(n-1) - 14\lfloor log(n-1) \rfloor - 38$

1 is the design in [19]
2 is the design in [23]
3 is the design in [20]
Prop_1 is the design in Figure 4.1
Prop_2 is the design in Figure 4.2. Steps 13, 15 replaced with Measure-and-Fixup gate
Prop_3 is the design in Figure 4.2

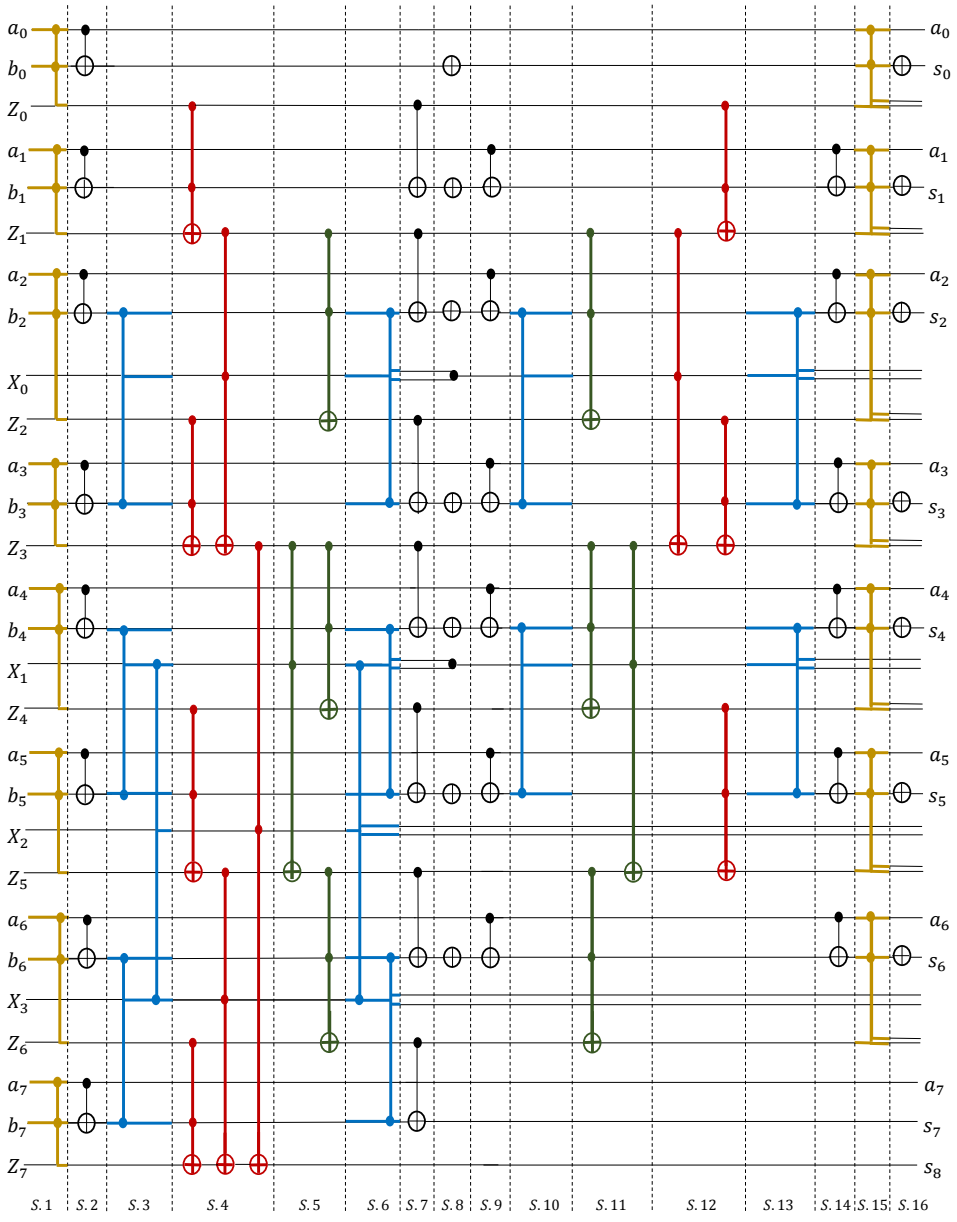


Figure 4.1: In-place QCLA: low T-count

Step 1,3,10 Computation gate
Step 2,7,9,14 CNOT gate
Step 4,5,11,12 Toffoli gate
Step 6,13,15 Uncomputation gate
Step 8,16 NOT gate

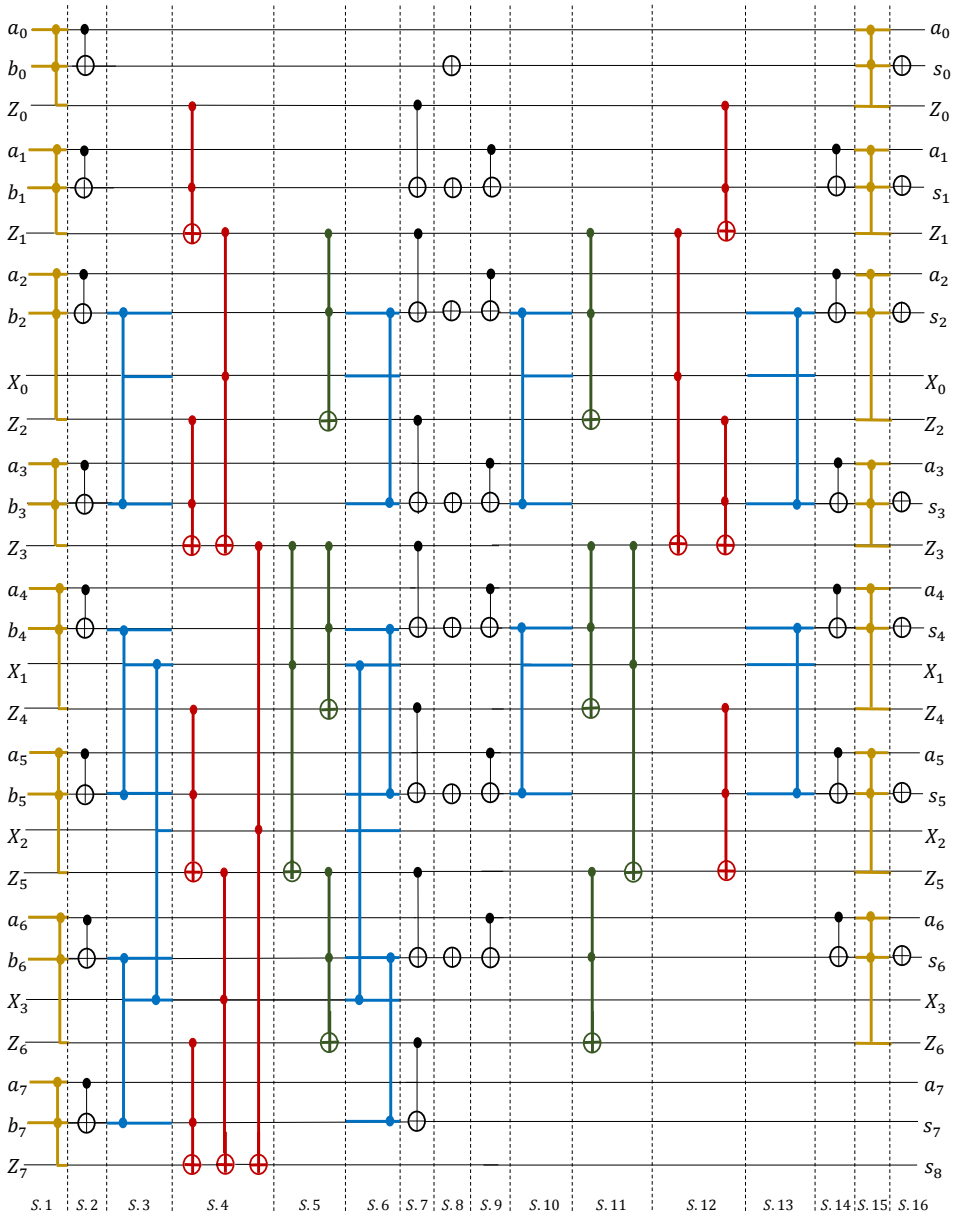


Figure 4.2: In-place QCLA: high speed

Step 1,3,10 Computation gate
Step 2,7,9,14 CNOT gate
Step 4,5,11,12 Toffoli gate
Step 6,13,15 Uncomputation gate
Step 8,16 NOT gate

Chapter 5

Simulation Results

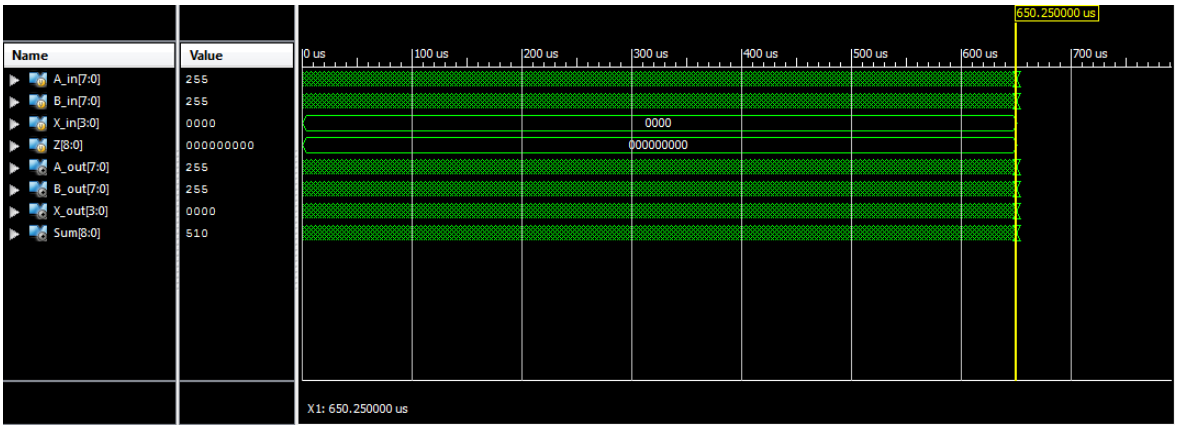
For the simulation results of the out-of-place and in-place QCLA circuits, the hardware description language or HDL that was used is Verilog 2001. With Verilog, quantum circuits could be simulated when they are decomposed into logical gates like AND and XOR. The simulation output can be seen on Xilinx's ISE Simulator called ISim. For both of the circuits, the inputs changed every 10ns.

5.1 Out-of-place QCLA

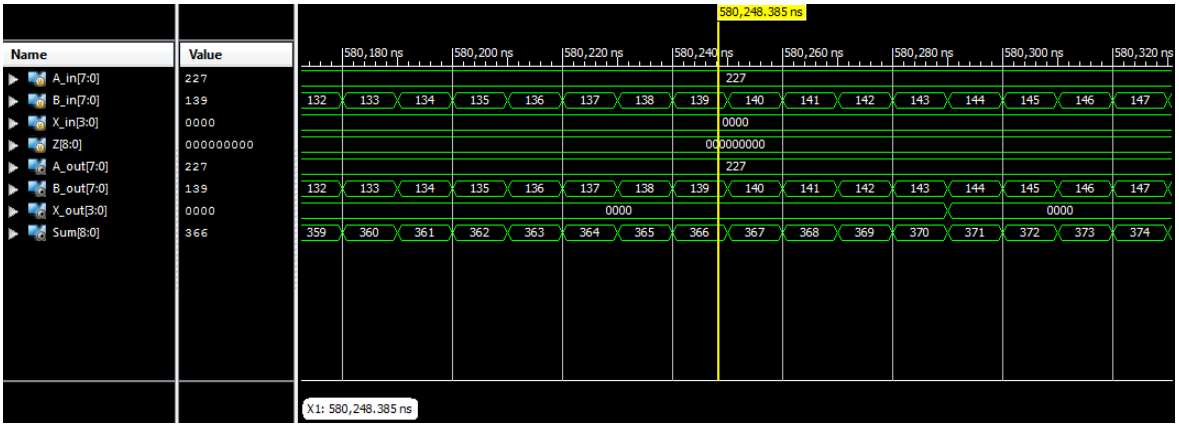
The first quantum circuit that was designed was the the out-of-place QCLA. The out-of-place QCLA was tested from 4-bit case to the 32-bit for correctness, it was also exhaustively tested, meaning tested using all possible cases, 2^n where n is dependent from n = 4-bit to n = 32-bit.

As it can be seen in Figure 5.1(a), a simulation of an 8-bit out-of-place QCLA. The inputs are the first four variables: A_in, B_in, X_in, and Z. The outputs are the last four variables: A_out, B_out, X_out, and Sum. This simulation shows all possible test cases of each input A_in and B_in from 0 to 2^8-1 or 255. As for Figure 5.1(b), a specific time frame was selected to show a closer look at the correct output for A_out, B_out, X_out, and Sum. One thing to note that the Measure-and-fixup and

Computation-Reversal uncomputation gates function the same when implemented in Verilog because Verilog is in a Classical domain not Quantum, therefore only one simulation was required to test the low T-count and the high speed quantum circuits.



(a)



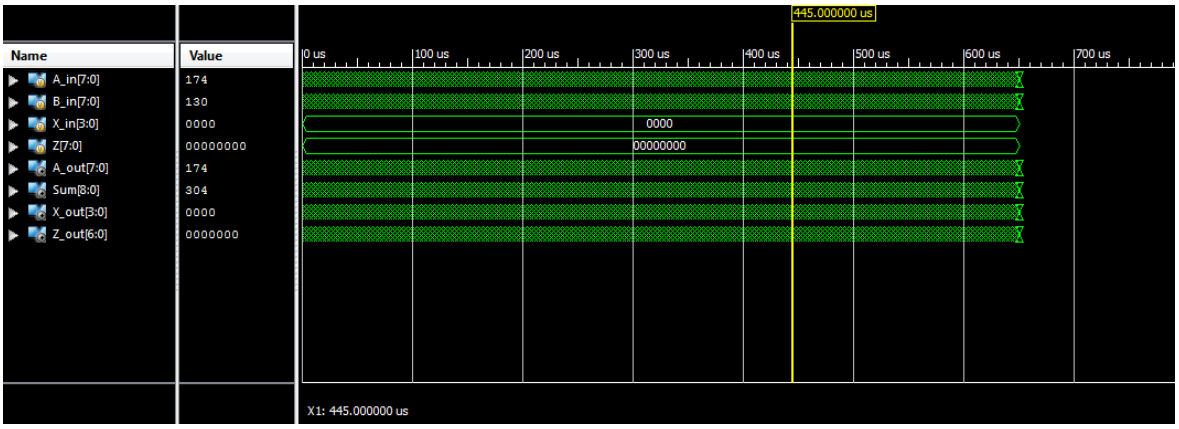
(b)

Figure 5.1: Out-of-place QCLA simulation
(a) Full View
(b) Specified View

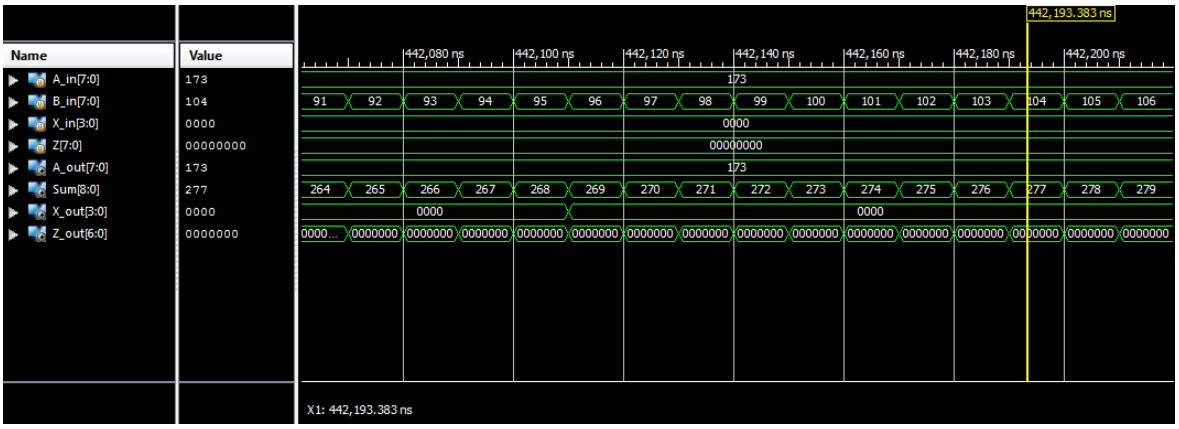
5.2 In-place QCLA

Just like the out-of-place QCLA, the in-place QCLAs were tested from 4-bit case to the 32-bit case for correctness, and tested exhaustively, 2^n possible test cases.

As it can be seen in Figure 5.2(a), a simulation of an 8-bit in-place QCLA. The inputs are the first four variables: A_in, B_in, X_in, and Z. The outputs are the last four variables: A_out, Z_out, X_out, and Sum. This simulation shows all the possible test cases of each input A_in and B_in from 0 to 2^8-1 or 255. For Figure 5.2(b), a specific time frame was selected to show a closer look at the correct output for A_out, Z_out, X_out, and Sum. One thing to note that the Measure-and-fixup and Computation-Reversal uncomputation gates function the same when implemented in Verilog because Verilog is in a Classical domain not Quantum, therefore only one simulation was required to test the low T-count, the high speed, and a mixture of both in-place QCLAs.



(a)



(b)

Figure 5.2: In-place QCLA simulation
(a) Full View
(b) Specified View

Chapter 6

Conclusion

In this thesis, five novel designs of quantum Carry Look-Ahead addition circuits having logarithmic depth were introduced. The proposed designs of the quantum Carry Look-Ahead Adder consisted of logical-AND computation, Measure-and-Fixup and Computation-Reversal uncomputation gates. These logical-AND computation and uncomputation gates were verified by hand or by using Quirk Simulator from [28]. As for the actual QCLA circuits, they ware simulated and tested with all possible test cases using Verilog HDL. But the main focus of this thesis was T-count efficient circuits for the out-of-place and in-place QCLAs.

The proposed designs are compared and achieve significant T-count savings compared to the existing works. The proposed QCLA circuits can be used in a larger quantum circuit where T-count is of primary concern.

The five designs that were introduced in this thesis can be the ground work for future improvements. One improvement to the QCLA circuits is changing all the Toffoli gates (in G-rounds, C-rounds, and their reverses) to the gates proposed by Jones in [37]. This will make each Toffoli gate have a T-count of 4 instead of 7. But the drawback of the Jones proposition is that there will be more ancilla inputs and outputs to the proposed QCLA circuits. Second improvement or proposition is the

proposed designs can be used in quantum circuits like multiplication and division to improve the efficiency. The last proposition is to apply the logical-AND computation and uncomputation gates into other quantum circuits, other than the QCLAs, where T-count is a major concern.

References

[1] T. S. Humble, H. Thapliyal, E. Muoz-Coreas, F. A. Mohiyaddin, and R. S. Bennink. Quantum computing circuits and devices. *IEEE Design Test*, pages 1–1, 2019.

[2] H.R. Bhagyalakshmi and M Venkatesha. Toffoli cascade synthesis of an optimized two-bit comparator. *Lecture Notes in Electrical Engineering*, 248:779–787, 09 2014.

[3] Eleanor Rieffel and Wolfgang Polak. *Quantum computing a gentle introduction*. The MIT Press, 2014.

[4] H. Thapliyal, E. Munoz-Coreas, T. S. S. Varun, and T. Humble. Quantum circuit designs of integer division optimizing t-count and t-depth. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019.

[5] Edgard Munoz-Coreas and Himanshu Thapliyal. T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 14(3):36, 2018.

[6] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the t-count. *Quantum Info. Comput.*, 14(15-16):1261–1276, November 2014.

[7] Austin Fowler, Ashley M. Stephens, and Peter Groszkowski. High threshold universal quantum computation on the surface code. *Physical Review A*, 80, 03 2008.

[8] N. Cody Jones, Rodney Van Meter, Austin G. Fowler, Peter L. McMahon, Jungsang Kim, Thaddeus D. Ladd, and Yoshihisa Yamamoto. Layered architecture for quantum computing. *Phys. Rev. X*, 2:031007, Jul 2012.

[9] K.-A. Brickman, P. C. Haljan, P. J. Lee, M. Acton, L. Deslauriers, and C. Monroe. Implementation of grover's quantum search algorithm in a scalable system. *Phys. Rev. A*, 72:050306, Nov 2005.

[10] Y. S. Weinstein, M. A. Pravia, E. M. Fortunato, S. Lloyd, and D. G. Cory. Implementation of the quantum fourier transform. *Phys. Rev. Lett.*, 86:1889–1891, Feb 2001.

[11] M. S. Tame, B. A. Bell, C. Di Franco, W. J. Wadsworth, and J. G. Rarity. Experimental realization of a one-way quantum computer algorithm solving simon's problem. *Phys. Rev. Lett.*, 113:200501, Nov 2014.

[12] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.

[13] Himanshu Thapliyal and Nagarajan Ranganathan. Design of efficient reversible logic-based binary and bcd adder circuits. *J. Emerg. Technol. Comput. Syst.*, 9(3):17:1–17:31, October 2013.

[14] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible floating-point adder architecture. In *2011 11th IEEE International Conference on Nanotechnology*, pages 451–456, Aug 2011.

[15] Rasha Montaser, Ahmed Younes, and Mahmoud Abdel-Aty. New design of reversible full adder/subtractor using r gate. *International Journal of Theoretical Physics*, 58(1):167–183, 2019.

[16] E. Muoz-Coreas and H. Thapliyal. Quantum circuit design of a t-count optimized integer multiplier. *IEEE Transactions on Computers*, 68(5):729–739, May 2019.

[17] Rich Rines and Isaac Chuang. High performance quantum modular multipliers. *arXiv preprint arXiv:1801.01081*, 2018.

[18] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible single precision floating point multiplier based on operand decomposition. In *10th IEEE International Conference on Nanotechnology*, pages 233–237, Aug 2010.

[19] Thomas G Draper, Samuel A Kutin, Eric M Rains, and Krysta M Svore. A logarithmic-depth quantum carry-lookahead adder. *arXiv preprint quant-ph/0406142*, 2004.

[20] Himanshu Thapliyal, HV Jayashree, AN Nagamani, and Hamid R Arabnia. Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder. In *Transactions on Computational Science XVII*, pages 73–97. Springer, 2013.

[21] Yasuhiro Takahashi and Noboru Kunihiro. A fast quantum circuit for addition with few qubits. *Quantum Information & Computation*, 8(6):636–649, 2008.

[22] Neela Shirisha, M Tech, P Kalyani, and D Nageshwar Rao. Design of a reversible carry look-ahead adder using reversible gates.

[23] Agung Trisetyarso and Rodney Van Meter. Circuit design for a measurement-based quantum carry-lookahead adder. *International Journal of Quantum Information*, 8(05):843–867, 2010.

[24] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[25] Giuliano Benenti, Guilio Casati, and Giuliano Strini. *Basic concepts*. World Scientific, 2008.

[26] D Michael Miller, Mathias Soeken, and Rolf Drechsler. Mapping ncv circuits to optimized clifford+t circuits. In *International Conference on Reversible Computation*, pages 163–175. Springer, 2014.

[27] Craig Gidney. Halving the cost of quantum addition. *arXiv preprint arXiv:1709.06648*, 2017.

[28] Craig Gidney. Quirk: Quantum circuit simulator, https://algassert.com/quirk.

[29] Austin G. Fowler and Simon J. Devitt. A bridge to lower overhead quantum computation. *arXiv e-prints*, Sep 2012.

[30] Simon J. Devitt, Ashley M. Stephens, William J. Munro, and Kae Nemoto. Requirements for fault-tolerant factoring on an atom-optics quantum computer. *Nature Communications*, 4, 2013.

[31] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

[32] Kristel Michielsen, Madita Nocon, Dennis Willsch, Fengping Jin, Thomas Lippert, and Hans De Raedt. Benchmarking gate-based quantum computers. *Computer Physics Communications*, 220:44 – 55, 2017.

[33] Kamyar Saeedi, Stephanie Simmons, Jeff Z Salvail, Phillip Dluhy, Helge Riemann, Nikolai V Abrosimov, Peter Becker, Hans-Joachim Pohl, John J L Morton, and Mike L W Thewalt. Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28. *Science (New York, N.Y.)*, 342(6160):830–833, 2013.

[34] Darshan Thaker, Tzvetan Metodi, Andrew Cross, Isaac Chuang, and Frederic Chong. Quantum memory hierarchies: Efficient designs to match available parallelism in quantum computing. In *Proceedings of the 33rd annual international symposium on computer architecture*, volume 2006 of *ISCA '06*, pages 378–390. IEEE Computer Society, 2006.

[35] Daniel Kudrow, Kenneth Bier, Zhaoxia Deng, Diana Franklin, Yu Tomita, Kenneth Brown, and Frederic Chong. Quantum rotations: a case study in static and dynamic machine-code generation for quantum computers. In *Proceedings of the 40th Annual International Symposium on computer architecture*, ISCA '13. ACM, 2013.

[36] Stephen D. Brown and Zvonko G. Vranesic. *Fundamentals of digital logic with Verilog design*. McGraw-Hill Higher Education, 2008.

[37] Cody Jones. Novel constructions for the fault-tolerant toffoli gate. *Physical Review A*, 87, 12 2012.

Vita

Vladislav Ivanovich Khalus

Education

University of Kentucky
Bachelor of Science in Electrical Engineering, May 2016
Bachelor of Science in Computer Engineering, May 2016

Experience

Embedded Software Engineer
June 2018 - Current
KPIT Technologies Inc.
Novi, MI
Graduate Research Assistant
August 2016-May 2017
University of Kentucky
Lexington, KY

Publication

Vladislav Khalus, Edgard Muñoz-Coreas, and Himanshu Thapliyal. "T-count Optimized Quantum Circuit for Logarithmic Addition" 22nd Annual Conference on Quantum Information Processing, Boulder, January 2019.