Chapter 1

Multiple Controlled Toffoli

The quantum computing paradigm was first theorized in the 1980s with the quantum Turing machine by Paul Benioff. Over the last 40 years, there has been many significant breakthroughs in quantum algorithms that help consolidate the quantum computing model. Some of the most notable innovations include the Shor's factoring algorithm, the Grover's search algorithm, and most recently, the Variational Quantum Eigensolver. These algorithms take advantages of the unique properties of quantum systems, such as superposition, interference and entanglement, to provide non-trivial speed up to classical problems.

One big similarity that many quantum algorithms share is the use of the "black box" oracle. The oracle encodes information about the system of interest, and is physically implemented by controlled operations between different qubits in the quantum circuit.

This chapter summarizes from literature the general construction of one such controlled operations, the multiple-controlled Toffoli gate. We look at different constructions across all different circuit scales and provide a short list of the most efficient implementations when minimizing for $CNOT$ count.

1.1 Background

1.1.1 Pauli Identities

We have 3 general rotation gates: R_x, R_y, R_z. These are gates parameterized by an angle θ that rotates the statevector θ radians around the x, y, or z axis respectively. When $\theta = \pi$, these rotation gates take on special forms, called the Pauli matrices. These matrices satisfies the Pauli identities, namely:

$$\begin{gathered} X^2 = Y^2 = Z^2 = I \\ XY = -YX = iZ \\ YZ = -ZY = iX \\ ZX = -XZ = iY \\ XYZ = iI \end{gathered}$$

The R_z gate has a special form called the phase gate, denoted P, which differs from R_z by a global phase, where

$$R_z(\theta) = e^{-i\frac{\theta}{2}} P(\theta) \tag{1.1}$$

1.1.2 Hadmard Identities

The Hadamard gates can be used to transform between rotations in the z basis to the x basis and vice versa.

$$\begin{aligned} HR_z(\theta)H &= R_x(\theta) \\ HR_x(\theta)H &= R_z(\theta) \end{aligned} \tag{1.2}$$

When $\theta = \pi$, we have the special case

$$HZH = X, \quad \text{and} \quad HXH = Z$$

1.1.3 Controlled Gates

The simplest control operations is the $CNOT$ gate, lesser known as the Feynman gate. This gate acts a control bit $|q_0\rangle$ and a target bit $|q_1\rangle$. When both of the inputs are in the z basis states, $CNOT$ performs a bit-flip on the target bit if the control bit is $|1\rangle$. In other words,

$$CNOT\,|q_0 q_1\rangle = \begin{cases} |q_0 q_1\rangle & \text{if } q_0 = 0 \\ |q_0 \overline{q_1}\rangle & \text{if } q_0 = 1 \end{cases}$$

We represent $CNOT$ on a circuit with a dot on the control, and an XOR circle on the target. From now on, we will call the XOR operation "addition (modulo 2)".

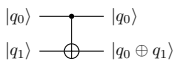


Figure 1.1: Action of $CNOT$ gate

As a unitary matrix, we have

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & X \end{array}\right) \tag{1.3}$$

In fact, we can generalize this to any two-qubits control unitary operation

$$CU\,|q_0 q_1\rangle = \begin{cases} |q_0\rangle\,|q_1\rangle & \text{if } q_0 = 0 \\ |q_0\rangle\,U\,|q_1\rangle & \text{if } q_0 = 1 \end{cases}$$

where $U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is a unitary matrix

Again, when both input are in the basis states, we can devised a clever notation that represents the controlled operations as matrix power U^k.

The unitary matrix is similar to that 1.3; now with the (2×2) U in the bottom left quadrant.

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix} = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & U \end{array}\right) \tag{1.4}$$

By Barenco et al. (1995) [1], we can decompose the CU gate into $CNOT$s and single qubits gates.

Proof: Any single qubit unitary matrix can be represented as orthogonal pairs rotations on the Bloch sphere. First, we define three rotation angles along two different axes. By Lemma 4.1,

$$U = e^{i\delta} \cdot R_z(\alpha) \cdot R_y(\theta) \cdot R_z(\beta)$$

In addition, the general 2×2 unitary matrix can also be written under the form $U = e^{i\delta} A \cdot X \cdot B \cdot X \cdot C$, where $A \cdot B \cdot C = I$. To show this, set

- $A = R_z(\alpha) \cdot R_y(\frac{\theta}{2})$
- $B = R_y(-\frac{\theta}{2}) \cdot R_z(-\frac{\alpha+\beta}{2})$
- $C = R_z(\frac{\beta-\alpha}{2})$

Now to turn this into a controlled-unitary operation, we place A, B, C in between the target of the $CNOT$ gate. If the control input is 0, then we have $A \cdot B \cdot C = I$ on the target, which has no effect on the input. Conversely, if the control input is 1, then we get $A \cdot X \cdot B \cdot X \cdot C$ on the target. Note that we are still missing a relative phase $e^i\delta$. We include this by adding a phase gate $P(\delta) = \begin{pmatrix} 1 & 0 \\ 0 & e^i\delta \end{pmatrix}$ anywhere on the control qubit. □

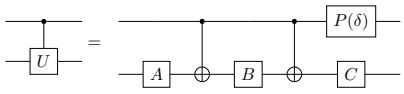


Figure 1.2: Decomposition of arbitrary controlled-unitary gate

We will now shift our attention to higher order controlled gates.

1.2 Notations

The controlled operation $CU(n)$ performs non-trivial transformation on an n qubits subset of a quantum circuit and used $n-1$ control bits along with 1 target bit. Note that, all permutations of the target bit below are equivalently called $CU(n)$.

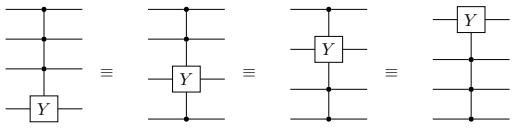


Figure 1.3: These 4 gates are equivalently $CY(4)$

When the discussion comes to discussing complexity of different technique, we will represent an instance of a gate as a scalar which denotes its $CNOT$ count. We then categorize these gates into different sets depending on which techniques are used to construct them. Conveniently, we can also define these sets as sequences parameterized by n, the number of qubits that the gate acts on. We group these techniques into different sets of their own, based on their similar construction constraints such the use of ancilla or the introduction of non-trivial relative phase. In the end, we outline the most efficient construction for any given circuit size and constraints.

For the rest of the chapter, we will be focusing on the construction $CX(n)$ and or $CZ(n)$ gates, but keep in mind that our method can be generalized to any unitary $CU(n)$ gates.

1.3 Definitions

Defintion 1. The permutation of the $CX(n)$ gate whose target is the lowest bit is defined by the matrix

$$CX(n) = \text{diag}\left\{1, 1, 1, ..., \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\right\} \tag{1.5}$$

When acting on an n-bit system, the action of $CX(n)$ is given by

$$CX(n)\,|x_1, x_2, ..., x_n\rangle = |x_1, x_2, ...x_{n-1}\rangle\, X^{\prod_{i=1}^{n-1} x_i}\,|x_n\rangle \tag{1.6}$$

Defintion 2. Given a unitary matrix U, its relative phase version is a unitary matrix V such that $|u_{i,j}| = |v_{i,j}|$ for all i, j, which implies that the magnitude of all elements of U and V are equal.

Using this definition we defined the $RCX(n)$ gate, the relative phase version of the $CX(n)$ gate, given by

$$RCX(n) = \text{diag}\left\{z_0, z_1, ..., z_{2^n-3}, \begin{pmatrix} 0 & z_{2^n-2} \\ z_{2^n-1} & 0 \end{pmatrix}\right\} \tag{1.7}$$

where $z_j = e^{i\theta_j}$

By [2], we can express the $RCX(n)$ as the matrix product of $CX(n)$ and a diagonal phased matrix $D(n)$.

$$RCX(n) = D(n) \cdot CX(n)$$

where $D(n) = \text{diag}\left\{z_0, z_1, ..., z_{2^n-1}\right\}$, with $z_j = e^{i\theta_j}$

Defintion 3. The inverse of $RCX(n)$ is also a $RCX(n)$ gate. This identity is very powerful because it allows us to reverse the presence of any relative phase when using $RCX(n)$ in our construction. We are also using without proof the identity $CX(n) = CX^{-1}(n)$.

$$\begin{aligned} RCX^{-1}(n) &= \left[D(n) \cdot CX(n)\right]^{-1} \\ &= CX^{-1}(n) \cdot D(n)^{-1} \\ &= CX(n) \cdot D^{-1}(n) \end{aligned}$$

where $D^{-1}(n) = \text{diag}\left\{\overline{z}_0, \overline{z}_1, ..., \overline{z}_{2^n-1}\right\}$, with $\overline{z}_j = e^{-i\theta_j}$

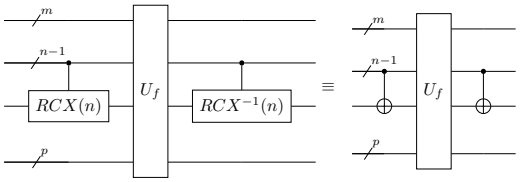
Defintion 4. An arbitrary $m+n+p$ qubits gate U_f nested between $RCX(n)$ and its inverse $RCX^{-1}(n)$ satisfies the identity

$$[I^m \otimes RCX(n) \otimes I^p] \cdot U_f \cdot [I^m \otimes RCX^{-1}(n) \otimes I^p] = [I^m \otimes CX(n) \otimes I^p] \cdot U_f \cdot [I^m \otimes CX(n) \otimes I^p]$$

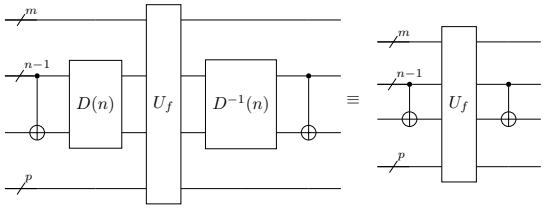
$$\iff \tag{1.8}$$

$$[I^m \otimes D(n) \otimes I^p] \cdot U_f \cdot [I^m \otimes D^{-1}(n) \otimes I^p] = U_f$$

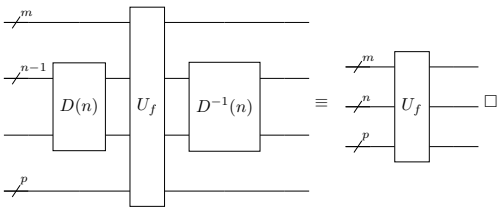
Proof. Visually, we have the circuit



Expand $RCX(n)$ and $RCX^{-1}(n)$ into its phased and non-phased components



Cancelling the two exterior $CX(n)$ from both sides, we get



□

1.4 Gray Code Decomposition

For $CX(3)$, also known as the Toffoli gate, we start with the Sleator-Weinfurter construction.

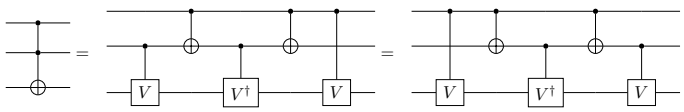


Figure 1.4: Sleator-Weinfurter Construction

where $V^2 = X$.

With $X = e^{i\frac{\pi}{2}} R_x(\pi)$, we can derive that $V = \sqrt{X} = e^{i\frac{\pi}{4}} R_x(\frac{\pi}{2})$. Using the identities from 1.2, we get

$$\begin{aligned} e^{i\frac{\pi}{4}} R_x(\frac{\pi}{2}) &= e^{i\frac{\pi}{4}} H \cdot R_z(\frac{\pi}{2}) \cdot H \\ &= H \cdot S \cdot H \end{aligned}$$

Now, we substitute H and P gates in for V and $V^\dagger$. Looking strictly at the target wire, we see the all the interior H gates cancels out, leaving only two H on the outside. This is a common method for building $CX(n)$, where we first build the $CZ(n)$ gate, then surround H gates on both sides of the target bit to form $CX(n)$.

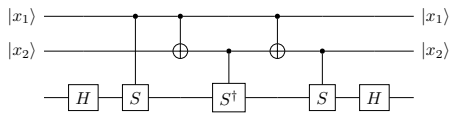


Figure 1.5: Modified Sleator-Weinfurter Construction with Controlled-Phase Gates

We can analyse the action of this circuit by looking at the operations applied on the target bit based on the the inputs x_1 and x_2. Specifically, we apply S whenever $x_1 = 1$ and also when $x_2 = 1$, but $V^\dagger$ whenever $x_1 \oplus x_2 = 1$. Overall, the gate applied on the target bit is X^p, where

$$p = x_1 + x_2 - (x_1 \oplus x_2) = 2 \cdot (x_1 \wedge x_2) = 2 \cdot (x_1 \cdot x_2)$$

This is consistent with definition of the $CX(n)$ gate in 1.5. By Figure 1.2, each controlled-phase gates, called CP, decompose into 2 $CNOT$s. In total, this construction of the Toffoli uses $3 \cdot 2 + 2 = 8$ $CNOT$ gates.

We can extend this construction technique to build $CX(4)$. The key behind this is placing $CNOT$s and controlled-phase gates in the correct places, so that the total action on the target bit gives $X^{(x_1 \cdot x_2 \cdot x_3)}$.

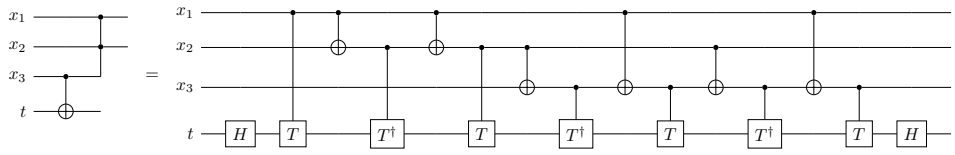


Figure 1.6: Gray code construction of $CX(4)$

We will denote the control bits, from the top down, as x_1, x_2, x_3. Again, we look at the operations on the target bit based on the value of the inputs.

$$
\begin{aligned}
T \text{ iff } x_1 = 1 \quad & (100) \\
T^\dagger \text{ iff } x_1 \oplus x_2 = 1 \quad & (110) \\
T \text{ iff } x_2 = 1 \quad & (010) \\
T^\dagger \text{ iff } x_2 \oplus x_3 = 1 \quad & (011) \\
T \text{ iff } x_1 \oplus x_2 \oplus x_3 = 1 \quad & (111) \\
T^\dagger \text{ iff } x_1 \oplus x_3 = 1 \quad & (101) \\
T \text{ iff } x_3 = 1 \quad & (001)
\end{aligned}
$$

By [3], we can reformulate this structure to describe a more efficient scheme for building $CX(3)$. First we replace all the CP gates with the same single-qubit phase gate in the position of the control bit. Finally, we move the exterior H gates one wire up. This construction uses 6 $CNOT$ gates, a reduction of 2 compared to the construction in Figure 1.5.

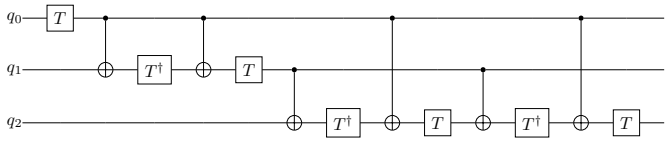


Figure 1.7: Gray code construction of $CX(3)$ using $CNOT$ and phase gates

We will now describe the procedure for constructing general $CX(n)$ gate.

1. Generate the Gray code sequence of length $2^n - 1$ starting with 0...1. Then, invert the binary number so that for example 10011 becomes 11001. Note that the most significant bit of the number correspond to the qubit x_1.
2. For each number in the sequence, we assign a phase gate to it, alternating between $P(\frac{\pi}{2^{n-1}})$ and its inverse.
3. We divide the sequence into n consecutive groups, where the ith group (starting from $i = 1$) contains 2^{i-1} element. For example, with $n = 3$, we split 7 elements into 3 different groups.

$$\{\underbrace{100}_{1}, \underbrace{110, 010}_{2}, \underbrace{011, 111, 101, 001}_{3}\}$$

4. The first number in the sequence, we apply the $P(\frac{\pi}{2^{n-1}})$ gate to the qubit x_1.
5. The second number in the sequence, we apply the gate $CNOT$ to qubit x_1 (control) and x_2 (target). Then, we apply $P(-\frac{\pi}{2^{n-1}})$ gate to qubit x_2.
6. Then iterating through each number in group ith, we apply the $CNOT$ gate with the control on the qubit that correspond to the newly changed bit of the sequence, and the target on qubit x_i. Then, we apply the phase gates assign in step 2 to the qubit x_i.
7. Finally, place one H gate at the beginning and the end of qubit x_n

1.4.1 CNOT Count

We can start by counting the number of $CNOT$ used in our circuit. For each wire, we count the number $CNOT$ targets that it contains. For x_1, we have 0 targets, for x_2, we have 2, for x_3, we have

4. In general, for the wire x_k, we have 2^{k-1} target bits. Summing the $CNOT$ target in all wires x_2 to x_n gives us the expression

$$\begin{aligned} CX(n) &= \sum_{k=2}^{n} 2^{k-1} \sum_{k=1}^{n-1} 2^k \\ &= 2^{n-1+1} - 1 - 1 \\ &= 2^n - 2 \end{aligned}$$

Overall, this construction for $CX(n)$ uses $2^n - 2$ $CNOT$s and $2^n - 1$ phase gates. We found that this construction is optimal in terms of $CNOT$s for $n = 3, 4, 5$. After that we will use other more efficient construction technique.

1.5 Recursive Relative Phase Decomposition

We will take a detour from the normal $CX(n)$ construction and discuss $RCX(n)$ construction. Recall that this is equivalent to $CX(n)$ except for the possibility of relative phase whenever its matrix elements are non-zero (1.7). We will look at a recursive construction proposed by Pelaez, E. and Pham, M. (2021).

1.5.1 Construction

We outlined a scheme to recursively construct $RCX(n)$ gates. We start with our three bases cases: $CNOT$, $RCX(3)$ and $RCX(4)$. $RCX(3)$ uses 3 $CNOT$s and 4 phase gates. $RCX(4)$ uses 6 $CNOT$s and 8 phase gates.

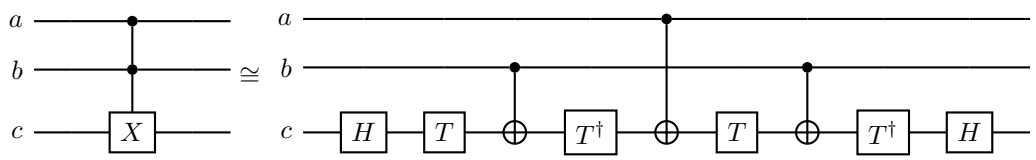


Figure 1.8: $RCX(3)$ base construction

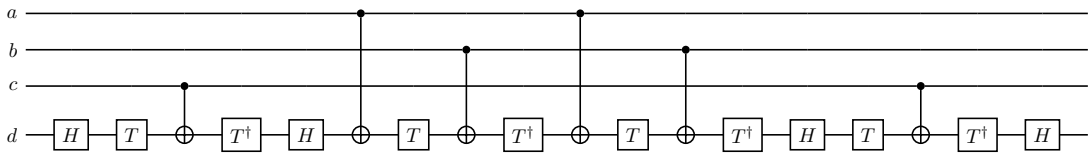


Figure 1.9: $RCX(4)$ base construction

Note that ≅ means that the circuit on the right is a relative phase version of the circuit on the left. In the final decomposition and gate count, the circuit in the right is considered.

Notice how $RCX(4)$ (Figure 1.9) has 3 $CNOT$ pairs, one on qubit a, one on qubit b, and one on qubit c. We can build $RCX(5)$ by first extending the $CNOT$ to a $CX(3)$.

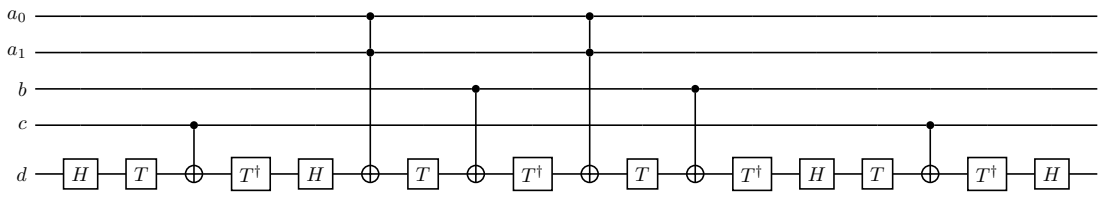


Figure 1.10: $RCX(5)$ from $RCX(4)$

Then, we can replace the normal $CX(3)$ with the $RCX(3)$ base case shown in Figure 1.8.

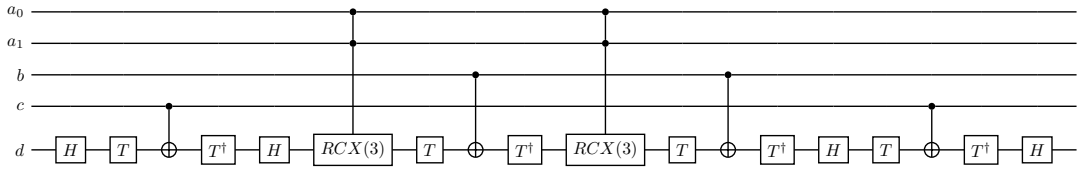


Figure 1.11: Replacing normal $CX(3)$ with $RCX(3)$

Similarly for $RCX(6)$, and $RCX(7)$, we can extend the $CNOT$ pairs into $CX(3)$, and replace these with $RCX(3)$. For $RCX(8)$, $RCX(9)$, and $RCX(10)$, we repeat the same process as above, however this time extend the $CNOT$ pairs into $CX(4)$, and replace these with $RCX(4)$. For $RCX(11)$, $RCX(12)$, and $RCX(13)$, we extend the CX pairs into $CX(5)$, and replace these with the $RCX(5)$ gates that we previously built.

In general, for an arbitrary $RCX(n)$ where $n > 4$,

1. Start with the structure of $RCX(4)$
2. Extend the a qubit $CNOT$ pairs to $CX(\lceil \frac{n-1}{3} \rceil)$,
3. Extend the b qubit CX pairs to $CX(\lfloor \frac{n-1}{3} \rfloor)$
4. Extend the c qubit $CNOT$ pairs to $CX(n-1-\lceil \frac{n-1}{3} \rceil - \lfloor \frac{n-1}{3} \rfloor)$.
5. Then, decompose each of the last three gates using the same process as outlined by step $1-3$. We repeat step $1-5$ until all the gates decompose to the gates in the base cases. At the end, we replace these gates with their corresponding phase gates.

1.5.2 CNOT Count and Applications

When we goes from $RCX(n-1)$ to $RCX(n)$, we replace two $RCX(\lceil \frac{n-1}{3} \rceil - 1)$ with two $RCX(\lceil \frac{n-1}{3} \rceil)$ gates. The cost of any $RCX(n)$ then, where $n \geq 5$, is given by the relation

$$RCX_P(n) = RCX_P(n-1) + 2 \cdot RCX\big(\lceil \frac{n-1}{3} \rceil\big) - 2 \cdot RCX\big(\lceil \frac{n-1}{3} \rceil - 1\big) \tag{1.9}$$

Asymptotically, this relation grows by $\Theta(n^{\log_6 3})$ $CNOT$s.

The application for RCX gates are given by 1.8, whenever RCX and RCX^{-1} are used concurrently. For example, we will encounter the following configuration when we discuss the quadratic construction by Barenco et al. (1995) [1] (Figure 1.18). The replacement of the CX by the RCX gates here will significantly decrease the number $CNOT$ used.

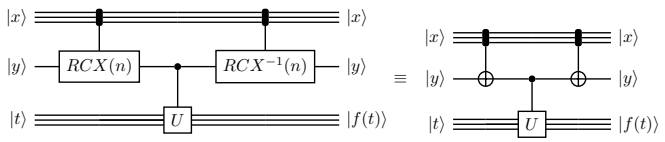


Figure 1.12: Application of RCX

1.6 V-Chain Decomposition

We can construct $CX(n)$ gates using a linear number of $CNOT$s if we allow for the possibility of ancilla bits. Whenever necessary, we denote the number of ancilla used / allowed with a superscript next to the gate. We will discuss the V-Chain decomposition by Maslov (2016) [2], which is an improvement upon Lemma 7.2 of [1]. We can improve this technique further by introducing the RCX gates discuss in section 1.5. This technique builds the $CX(n)$ gates using $\lceil \frac{n-6}{2} \rceil$ borrowed bits. This means that the ancillae could start in any state, but must be returned to that same state afterwards.

1.6.1 Construction

From now, we refer to the set of qubits that our gate acts on in square bracket, with the last element being the target bit. The top-most qubit are indexed 1, and the rest are in ascending order going down the circuit. For $n \geq 8$, the construction is defined as follows:

1. Start with a circuit of size $m = n + \lceil \frac{n-6}{2} \rceil$. The top $n-1$ bits are controls, the next $\lceil \frac{n-6}{2} \rceil$ are ancillae, and the bottom bit is the target.
2. The first gate is $CX(3)[n-1, m-1, m]$
3. If n is odd, place $CX(3)[n-2, m-2, m-1]$. Else, skip this step
4. When $n > 8$, for each k in $\lfloor \frac{n-8}{2} \rfloor, ..., 1$, place $CX(4)[2k+5, 2k+6, n+k-1, n+k]$
5. Next is $CX(7)[1, 2, 3, 4, 5, 6, n]$
6. When $n > 8$, for each k in $1, ..., \lfloor \frac{n-8}{2} \rfloor$, place $CX(4)[2k+5, 2k+6, n+k-1, n+k]$ (reverse order of step 4)
7. If n is odd, place $CX(3)[n-2, m-2, m-1]$. Else, skip this step
8. The next gate is $CX(3)[n-1, m-1, m]$
9. Implement step 3.
10. Implement step 4.
11. Next is $CX(7)[1, 2, 3, 4, 5, 6, n]$
12. Implement step 6.
13. Implement step 7.

We will explain the motivation behind this construction with an example for $n = 11$ shown by Figure 1.13. In this example, there are 10 control bits, labeled A to J. The left mountain (step 2-8) accumulates the product of the control bits into the target. This is done iteratively by storing part of the product in the ancilla bits. First, a_1 stores $a_1 \oplus ABCDEF$. Then a_2 stores $a_2 \oplus ABCDEFGH$. After

that a_3 stores $a_3 \oplus ABCDEFGHI$. Finally, the target t becomes $t \oplus ABCDEFGHIJ$. However, recall that we must return the ancilla to its original state. The right mountain (step 9-13) achieves this by adding (modulo 2) the same product as the first mountain to each ancilla bit. Since $(x \oplus y) \oplus y = x$, we successfully restore the original values of the ancilla bits.

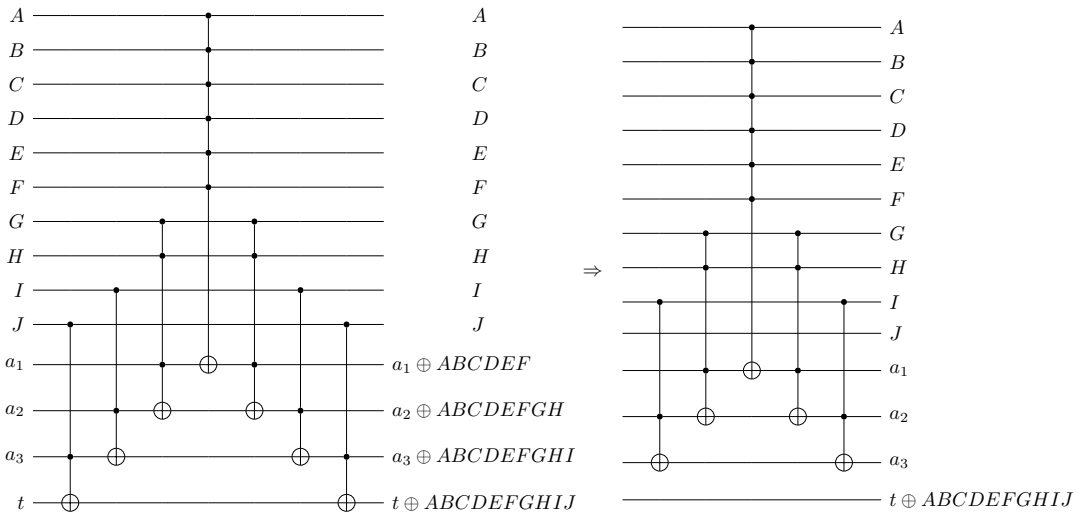


Figure 1.13: V-chain construction of $CX(11)$

The key in the efficient construction of the V-chain is replacing all CX gates with RCX. We introduce four gates that are less expensive to implement in the context of this construction, three of which comes from Maslov (2016) [2].

1. **SRTS**. This is a permutation of the Toffoli in Figure 1.7. When used with its inverse $SRTS^{-1}$, we can cancel out 2 $CNOT$s. In total, it contains 4 $CNOT$ gates. We substitute $SRTS$ in step 2 and $SRTS^{-1}$ in step 8.

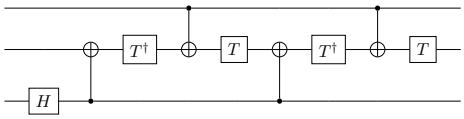


Figure 1.14: SRTS circuit: 4 $CNOT$s

2. **RTS**. When $RCX(3)$ is used with $RCX^{-1}(3)$, we can partly cancel out a $CNOT$ gate. The result is RTS. The circuit contains 2 $CNOT$ gates. We substitute RTS in step 3 and RTS^{-1} in step 7.

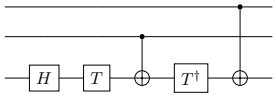


Figure 1.15: RTS circuit: 2 $CNOT$s

3. **RT4S**. This is the remainder of the $RCX(4)$ circuit after partial-cancellation with its inverse. The implementation uses 4 $CNOT$s. We substitute $RT4S$ in step 4 and $RT4S^{-1}$ in step 6.

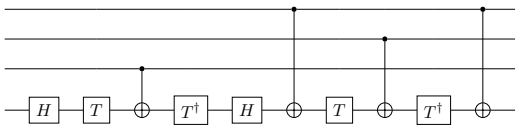


Figure 1.16: RT4S circuit: 4 $CNOT$s

4. **RT7L**. This circuit is equivalent to $RCX(7)$. However, we will call it $RT7L$ to be consistent with the original literature. $RT7L$ is composed of 18 $CNOT$s. We substitute $RT7L$ in step 5 and $RT7L^{-1}$ in step 11.

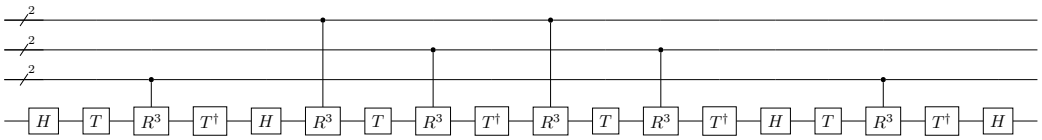


Figure 1.17: RT7L circuit: 18 $CNOT$s

To create the inverse of any of the gates above, simply flip the order of the $CNOT$, T, $T^\dagger$ and H gates, then inverse all the phase gates. Substituting the above gates into the circuit leads to the construction of $CX(n)$ with $\lceil \frac{n-6}{2} \rceil$ ancillae, using a linear number of $CNOT$ gates.

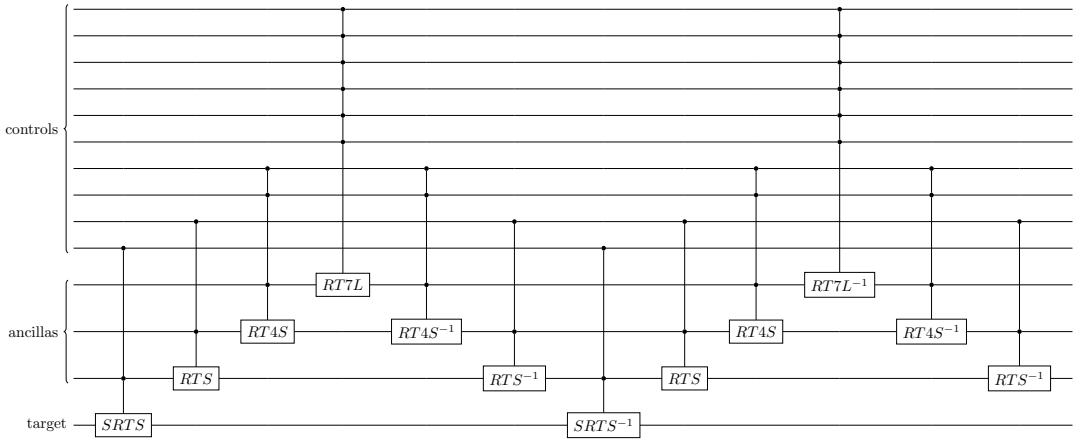


Figure 1.18: Substitution example of $CX(11)$

1.6.2 CNOT Count

We start with $n = 8$, which requires 2 $RT7L$ and 2 $SRTS$ gates, for a total of 44 $CNOT$s. For larger values of n, we either add 4 RTS gates to build $n+1$ or 4 $RT4S$ gates to build $n+2$. Both increases the $CNOT$ count by 8 per qubit. Thus, our construction of $CX(n)$ uses $8n - 20$ $CNOT$ gates.

1.7 Quadratic Decomposition

This decomposition scheme comes from Barenco et al. (1995) [1]. In a way, it is an extension of the structure in Figure 1.5, where the $CX(1)$ and $CS(1)$ gates turns into $CX(n-1)$ and $CS(n-1)$ gates.

1.7.1 Construction

Here, we will describe the construction of the $CZ(n)$ gates. However, the conversion between this and $CX(n)$ gates uses two H gates around the target bit of the gate. For an arbitrary $CZ(n)$ gate, we can decompose it into 2 $CX(n-1)$, CS , $CS^\dagger$ and $CS(n-1)$.

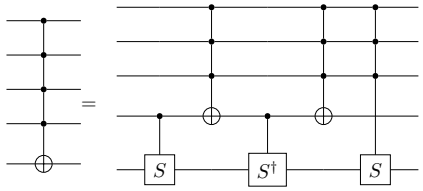


Figure 1.19: Example: $CZ(5)$

Then, we can further decompose $CS(n-1)$ into 2 $CX(n-2)$, CT , $CT^\dagger$ and $CT(n-2)$.

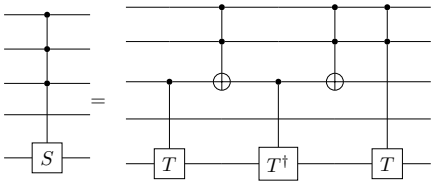


Figure 1.20: Example: $CS^1(4)$

From now, the number in bracket refers to the angle of the controlled-phase gates whenever it has the parameters θ inside, else it denotes size of the control gates. In general for an arbitrary $CP_\theta(n)$ gate, we decompose it into 2 $CX(n-1)$, $CP(\frac{\theta}{2})$, $CP^\dagger(\frac{\theta}{2})$ and $CP_{\frac{\theta}{2}}(n-1)$. We repeat this process for $CP_{\theta/2}(n-1)$ until we hit the base case $CP(\frac{\theta}{2^{n-2}})$.

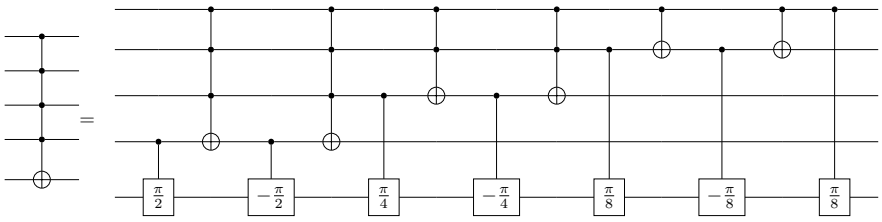


Figure 1.21: Example: Full Decomposition of $CX(5)$

where $\boxed{\theta}$ denotes the phase gate $P(\theta)$

What's left to do is to decompose all the CX gates inside our circuit. There are a few methods for doing this. The simplest way is to replace CX with the RCX gates described by section 1.5. This substitution is optimal for $n \leq 21$. Past this point, we can use the technique of Maslov (2016) [2] if we have enough ancilla bits. More specifically, for an arbitrary $CZ(n)$, we can implement $CX(m)$ as described in section 1.6 whenever $m + \lceil \frac{m-6}{2} \rceil \leq n$. If this is not the case, we can use the construction

of [4]. This method also builds RCX with the $CNOT$ counts growing linearly with the number of qubits. The technicality of this construction is outside the scope of this chapter.

1.7.2 CNOT Count

Since the construction is recursive, we can define a simple recursion relation for the $CNOT$ count.

$$CZ(n) = 2CX(n-1) + CP(n-1) + 2CP(2)$$

The asymptotic complexity of this expression depends on our choice of $CX(n-1)$. If we uses a linearly construction then we have

$$CZ(n) = CP(n-1) + \Theta(n)$$

which implies that $CZ(n) \in \Theta(n^2)$.

On the other hand, if we use the $\Theta(n^{\log_3 6})$ construction of 1.9, then our complexity is super-quadratic.

1.8 Summary

Construction	Relative Phase	# ancillae	# $CNOT$
Gray Code	No	N / A	$2^n - 2$
Recursive Relative Phase	Yes	N / A	$\Theta(n^{log_3 6})$
V-Chain	No	$\lceil \frac{n-6}{2} \rceil$	$8n - 20$
Quadratic	No	N / A	$\geq \Theta(n^2)$

Table 1.1: Summary of the constructions presented

Bibliography

[1] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995.

[2] D. Maslov, "Advantages of using relative-phase toffoli gates with an application to multiple control toffoli optimization," *Phys. Rev. A*, vol. 93, p. 022311, Feb 2016.

[3] N. Schuch and J. Siewert, "Programmable networks for quantum algorithms," *Phys. Rev. Lett.*, vol. 91, p. 027902, Jul 2003.

[4] M. Amy and N. J. Ross, "The phase/state duality in reversible circuit design," 2021.