1 Overview

In my approach, I constructed the MCX gate in 3 parts: concentration, calculation, and uncomputation.

In the concentration part, I chose a set of qubits and store the logical AND state into an ancilla qubit. This process can reduce the number of variables because we can use the stored state instead of the input. Next, I calculated the logical AND state of the variables and stored it into the output. After that, I uncomputed the ancilla qubits by inverting the quantum gates used in the concentration part.

2 Approximate MCX Gates

In this section, I introduce some depth-efficient MCX gates that don't preserve the input states. These gates cannot be used in the calculation part, but are useful in the concentration and uncomputation part.

2.1 Reduced Toffoli gate

A Toffoli gate can be decomposed into 6 CNOT and 9 unitary gates. Among these gates, 2 CNOT and 3 unitary gates are used for preserving the control qubits (see Fig.1).

A reduced Toffoli gate doesn't contain the gates in red area, so this can reduce the circuit depth.

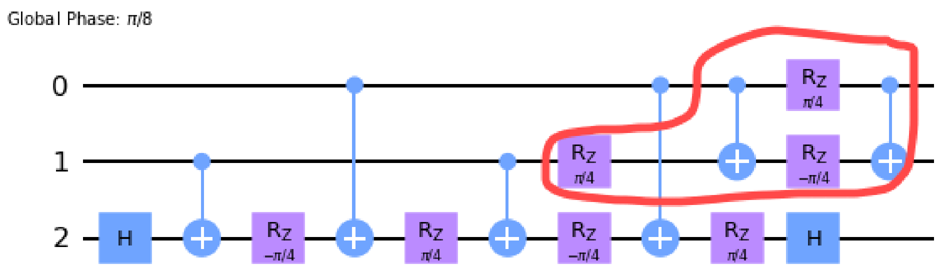

Figure 1: Reduced Toffoli gate. The gates in red area can be removed if the control qubits don't need to be preserved.

2.2 $R_Y(\theta)$-based Toffoli gate

There is another option for constructing a Toffoli gate (see Fig.2). This gate only uses 3 CNOT and 4 unitary gates, but requires the destination qubit to be clean.

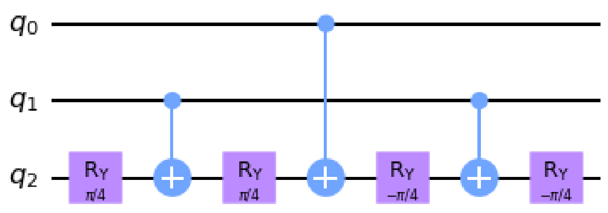

Figure 2: $R_Y(\theta)$-based Toffoli gate. This gate can properly perform the logical AND operation, but may add an extra phase.

2.3 $R_Y(\theta)$-based CCCX gate

There are 2 types of $R_Y(\theta)$-based CCCX gates (see Fig.3 and Fig.4). The latter one is constructed based on the Toffoli gate in Fig.2 and can be executed faster beacuse it needs less elementary gates.

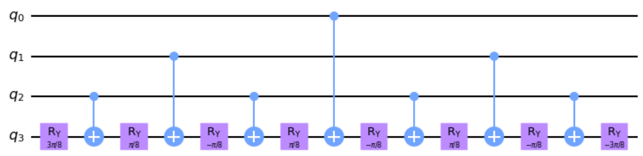

Figure 3: $R_Y(\theta)$-based CCCX gate.

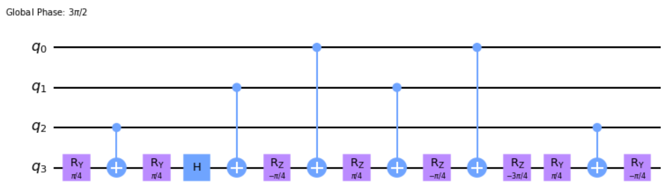

Figure 4: Another $R_Y(\theta)$-based CCCX gate. We can get this gate by replacing the second CNOT gate in Fig.2 with a reduced Toffoli gate.

2.4 $R_Y(\theta)$-based CCCCX gate

I constructed a CCCCX gate based on the CCCX gate shown in Fig.3. This gate is drawn in Fig.6 below.

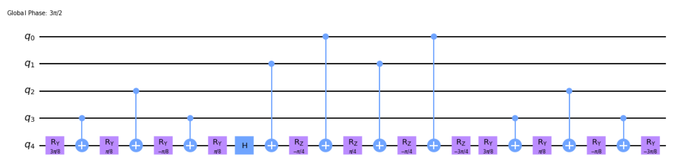
Figure 5: $R_Y(\theta)$-based CCCCX gate.

2.5 $R_Y(\theta)$-based CCCCX gate (with 1 clean ancilla)

We can construct this gate by combining 2 $R_Y(\theta)$-based Toffoli gates and 1 $R_Y(\theta)$-based CCCX gate(in Fig.3). This CCCCX gate uses a clean ancilla qubit to store the output of the Toffoli gate so that we can execute the Toffoli (and uncomputation) and CCCX in parallel.

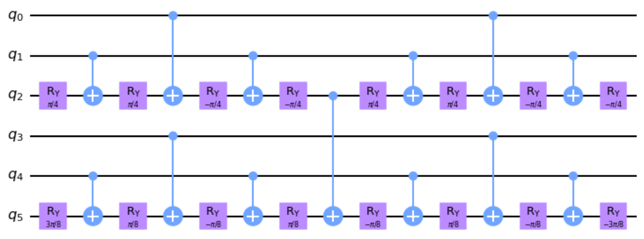
Figure 6: $R_Y(\theta)$-based CCCCX gate (with 1 clean ancilla).

3 Exact MCX gates

In the calculation part, an exact MCX gate is needed because we have to keep the control qubits to retain the same.

3.1 Exact Toffoli gate

Fig.9 shows a decomposed Toffoli gate obtained from the default transpiler. This decomposition is not optimal, so I changed the gate execution order in the manner below and parallelized the CNOT and unitary gates.

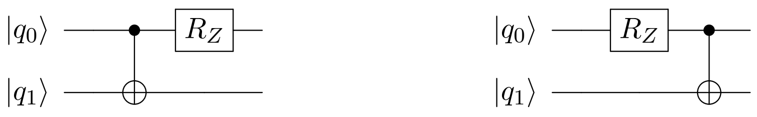

Figure 7: R_Z-control commutation.

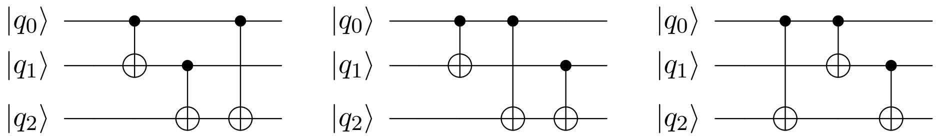

Figure 8: control-control, target-target commutation.

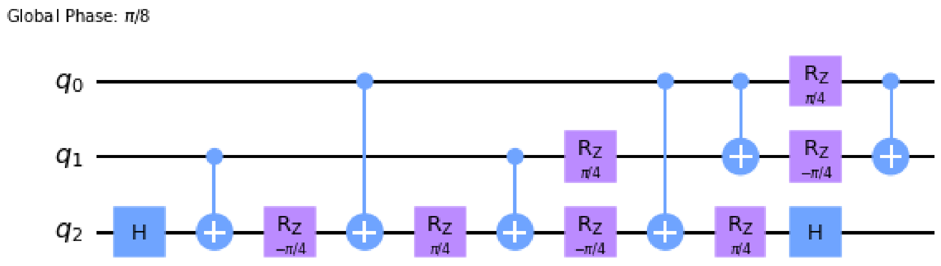

Figure 9: Exact Toffoli gate. This requires 11 time steps for execution.

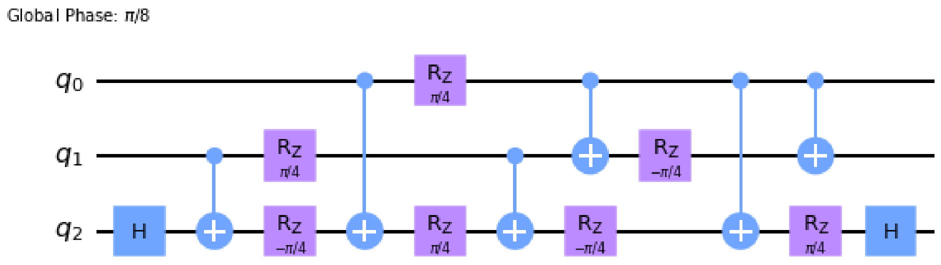

Figure 10: Depth-optimized Toffoli gate. This requires 2 less time steps for execution.

3.2 Exact CCCX gate

Using the same method, I changed the gate execution order and reduced the circuit depth.

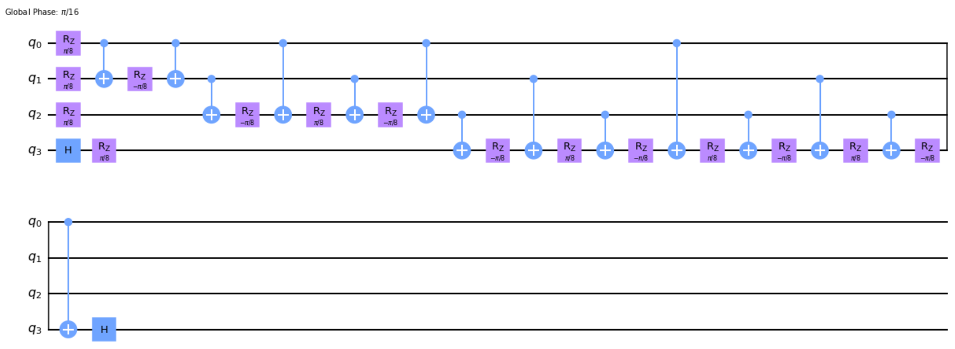

Figure 11: Exact CCCX gate. This requires 27 time steps for execution.

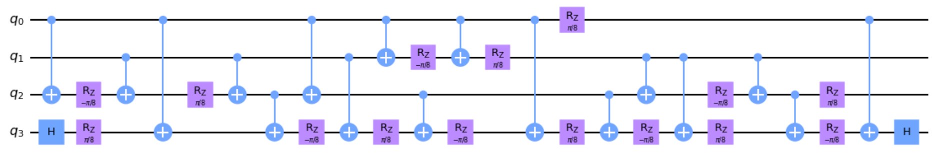

Figure 12: Depth-optimized CCCX gate. This requires 6 less time steps for execution.

4 Finding the minimum depth

To find the best solution for depth minimization, I wrote a Python script that can test any possible circuit that uses approximate/exact MCX gates.

When searching the optimal circuit, a MCX gate can be drawn as a block shown in Fig.13 and 14. The algorithm virtually connects these blocks, and calculates the minimum circuit depth. After connecting these components, we can calculate the depth of each qubit using the equation below:

$$\mathrm{depth}(q_{\mathrm{out}}) = \max_{q_{\mathrm{in}}} \left(\mathrm{depth}(q_{\mathrm{in}}) + \mathrm{width}(q_{\mathrm{in}}, q_{\mathrm{out}})\right) \tag{1}$$

where the set of $\mathrm{width}(q_{\mathrm{in}}, q_{\mathrm{out}})$ is determined by the shape of MCX gates. For example, $\{4, 6, 7\}$ for Toffoli gate, $\{8, 10, 12, 13\}$ for CCCX gate (shown in Fig.13), and $\{12, 14, 18, 20, 21\}$ for CCCCX gate (shown in Fig.14).

When calculating the circuit depth, we can arrange the order of input qubits so that the depth will be minimized.

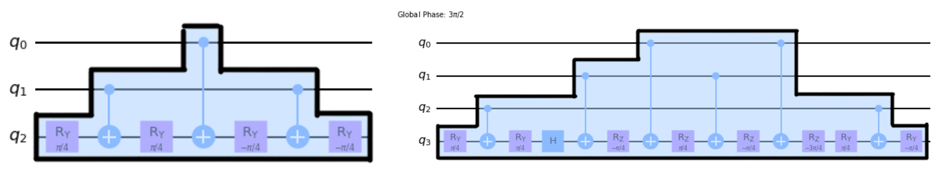

Figure 13: The block representation of a $R_Y(\theta)$-based Toffoli/CCCX gate. The Toffoli gate requires 4 and 6 time steps for each input, and 7 for the output. The CCCX gate requires 8, 10, and 12 time steps for each input, and 13 for the output.

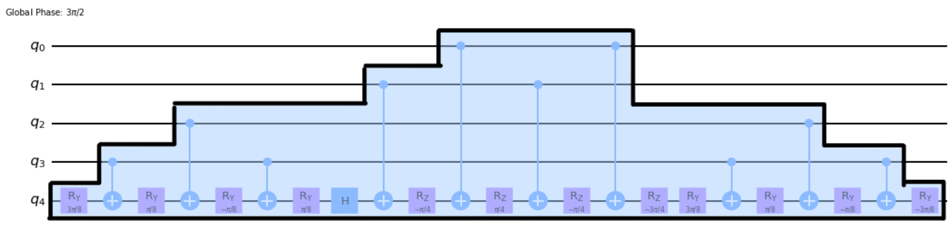

Figure 14: The block representation of a $R_Y(\theta)$-based CCCCX gate. This gate requires 12, 14, 18, and 20 time steps for each input, and 21 for the output.

5 The circuit

Fig.15 shows my solution. In this circuit, 2 $R_Y(\theta)$-based Toffoli gates, 2 $R_Y(\theta)$-based CCCX gates, 4 $R_Y(\theta)$-based CCCCX gates, 2 $R_Y(\theta)$-based CCCCX gate (with 1 clean ancilla), and an exact Toffoli gate are used to perform as a 14-controlled MCX gate.

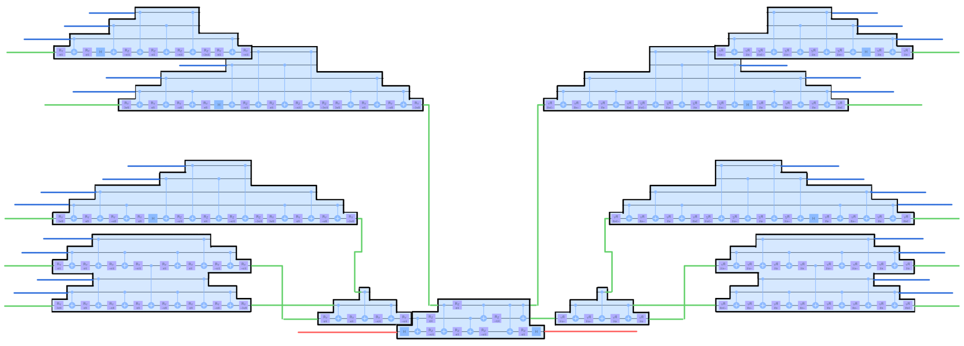

Figure 15: The 14 input qubits, 5 ancilla qubits, and 1 output qubit are drawn as blue, green, and red line(s).