Automated Mapping Methods for the IBM Transmon Devices

Kaitlin N. Smith and Mitchell A. Thornton
Quantum Informatics Research Group
Southern Methodist University
Dallas, TX, USA
{knsmith, mitch}@smu.edu

***Abstract*— Quantum computing and quantum information processing devices are becoming a reality. As a direct result of this new technology, there is a demand for methods that allow the automated translation of quantum processing algorithms into forms consistent with emerging device libraries. Quantum circuits in their original form may not always be compatible with a technology platform, so they must be transformed into technology-dependent models to be executed on a real quantum machines. This synthesis procedure must consider operational constraints of the physical quantum implementation such as the maximum number qubits, the available gates, and the connectivity between qubits for multi-qubit operations. In this work, algorithms that assist in technology-dependent quantum circuit mapping in the case of limited qubit connectivity will be explored. The IBM Q devices are the example target technology, and the CNOT operation is the multi-qubit gate with limited configurations for implementation.**

I. Introduction

Quantum information processing (QIP) will likely cause a new era of technological revolution due to all of the promise that quantum algorithms hold. Quantum algorithms have been proven theoretically to outperform classical methods in computing tasks that require large datasets or numerous iterations, but these algorithms can only be implemented when reliable quantum computers (QCs) that contain an adequate number of qubits exist. Many successful QC prototypes containing a modest number of qubits have been developed, but these devices have their differences with respect to what types of quantum algorithms, or circuits, they can execute. Because of the variety between QCs, even between machines built using the same base technology, methods must be developed that allow arbitrary quantum circuits to be mapped to technology-dependent models. Generalized quantum algorithms must be transformed into technology-dependent forms to be executed on a real quantum machine, and this transformation process must not only consider the available gates for the platform but also the connectivity that exists between qubits for multi-qubit operations.

The IBM Q machines are publically available transmon-based devices that allow users to experiment with custom quantum algorithms at the gate level. The IBM devices, however, are limited to unique connectivity maps for multi-qubit interactions with the two-qubit CNOT, or controlled-NOT, operator. Because it is common for many physical quantum realizations to have limited multi-qubit connectivity, finding generic reroute algorithms capable of implementing multi-qubit operations on uncoupled qubits for any architecture specified by a coupling map is of interest for a quantum logic synthesis tool. Mapping quantum logic transformations from one form to another is a tedious task if done manually because it quickly becomes difficult and error-prone as the number of inputs and outputs of a circuit increases. The need to perform technology-dependent mapping on quantum algorithms creates a demand for an automated synthesis tool that allows generalized quantum circuits to be transformed into designs that only include the limited library of quantum transformations that are executable on a specific platform. Two techniques that can be implemented in a technology-depentent quantum synthesis tool were developed: the adjacent SWAP qubit reroute algorithm and the connectivity tree qubit reroute algorithm. These algorithms were successfully applied to generate quantum circuits targeted for evaluation on different IBM machines with unique coupling requirements.

This paper will proceed as follows: Section II will provide a brief summary of important concepts in quantum computation while Section III introduces and describes the IBM QCs. The reroute methods are presented in Section IV and conclusions and future work are found in Section V.

II. Background on Quantum Computation

A. The Qubit

The unit of quantum information is the quantum bit, or qubit. A qubit models information as a linear combination of basis states. For example, $|0\rangle$ and $|1\rangle$ are a set of orthonormal basis states in Dirac notation that represent the two-dimensional column vectors of $|0\rangle = \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}}$ and $|1\rangle = \begin{bmatrix}0 & 1\end{bmatrix}^{\mathrm{T}}$, respectively. The qubit can theoretically represent an infinite continuum of states while in superposition of the of basis states, $|0\rangle$ and $|1\rangle$ as shown in

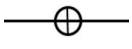
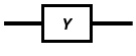
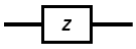
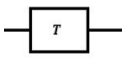
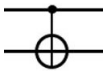
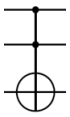
$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{1}$$

In Eqn. 1, the variables α and β are complex numbers, c, that take the form of $c = a + bi$ where i is an imaginary number that takes the value $i^2 = -1$. The probability that $|\Psi\rangle$ is measured to be $|0\rangle$ is equal to $|\alpha|^2$ and the probability that $|\Psi\rangle$ is measured to be $|1\rangle$ is equal to $|\beta|^2$. Once measured, qubits collapse into basis states, losing superposition.

Qubits can be physically realized with a variety of different physical entities. These realizations can be classified into two types: atomic-scale qubits and mesoscopic qubits. Atomic-scale qubits generally use actual particles, like atoms or ions, and also include quasi-particles such as anyons, quantum dots, and Ni vacancies in diamond. The second class of qubits, the mesoscopic forms, are realized as macroscopic circuits present in an integrated circuit (IC), yet they exhibit microscopic quantum properties at supercooled temperatures. Mesoscopic qubits are generally classified according to the information-carrying medium in the circuit such as charge, flux, and phase. Quantum phenomena in mesoscopic qubits require extremely low electrical resistance for viability and are realized in the form of electrical circuits on ICs. These devices are often referred to as superconducting solid-state qubits.

Accidental measurement of a qubit's superposition causes it to collapse into a basis state. This collapse, usually caused by an unintended interaction between the qubit and its environment, is called decoherence [1]. Some quantum systems are more resilient to decoherence as compared to others, and in QIP circuit and QC design, there is a trade-off between average decoherence times and the ability for qubits to interact. Multi-qubit coupling is necessary to enable required operations such as the CNOT or controlled-phase, but the ability for qubits to interact with each other implies that qubits also have the undesirable ability to interact with the environment, increasing the probability of decoherence. Determining qubit realizations that permit easy interaction with one another while also resisting environmental coupling is currently an active area of research that accounts for, in part, the lack of a wide consensus on the preferred implementation of a qubit. Many current QC/QIP realization efforts are focused upon the class of superconducting solid-state qubits as they are considered by many to offer the best tradeoff between qubit interaction versus average time of decoherence.

TABLE I
COMMON SINGLE- AND MULTI-QUBIT QUANTUM OPERATORS

Operator	Symbol	Transformation Matrix
Pauli-X (NOT)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

B. Quantum Operators

Quantum operators are implemented to purposely transform qubit state so that meaningful quantum computation can occur. If a quantum algorithm is modeled as a circuit, quantum operators can be viewed as quantum logic gates. These operators are represented by a unique, unitary transformation matrix, $\mathbf{U}$. Common single- and multi-qubit operations are featured in Table I.

Single quantum operators are combined to form quantum circuits, and quantum circuits can be described in a variety ways. Some of the most popular techniques include drawing the circuits as graphs, as seen in Fig. 1, or describing them with text in Quantum Assembly Language (QASM).

Understanding how information evolves in a quantum circuit requires knowledge of matrix multiplication and tensor products. As seen in Table I, quantum operators are represented by transformation matrices. To determine the resulting quantum state, $|\Psi_{out}\rangle$, after after the input state, $|\Psi_{in}\rangle$, undergoes a gate transformation, $\mathbf{U}$, the expression

$$\mathbf{U}\,|\Psi_{in}\rangle = |\Psi_{out}\rangle \tag{2}$$

must be evaluated. The individual qubit input values are combined via tensor product to create the quantum state for $|\Psi_{in}\rangle$. Consider the quantum circuit pictured in Fig. 1. In this circuit, two qubits, $|a\rangle$ and $|b\rangle$, are each represented by a single horizontal line, but these horizontal lines should not be confused with conductors like those in in electrical circuit schematics. These qubits travel from left to right as time progresses, passing though a CNOT gate. Together at the input they form the value of $|\Psi_{in}\rangle$ which is found to be

$$|\Psi_{in}\rangle = |a_{in}\rangle \otimes |b_{in}\rangle = |1\rangle \otimes |0\rangle$$

$$|\Psi_{in}\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle\,.$$

Determining the value of $|\Psi_{out}\rangle$ requires Eqn. 2 to be utilized. The transformation matrix of CNOT will take the place of the

generalized transformation matrix **U** in the equation, and the equation will be evaluated as

$$|\Psi_{out}\rangle = \mathbf{U}\,|\Psi_{in}\rangle = \mathbf{CNOT}\,|10\rangle$$

$$|\Psi_{out}\rangle = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = |11\rangle\,.$$

Finding the overall transfer function, **U**, that describes a quantum circuit containing many qubit operations is found using tensor and matrix products as well. Quantum gates in parallel are combined with a tensor product, and quantum gates in series are combined with a matrix product.

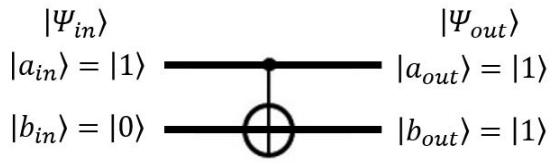


Fig. 1. Quantum circuit example 1

III. The IBM Q QC Devices

IBM has has developed multiple QCs and a quantum simulator are available to the public for experimentation [6]. Details of QCs can be found in Table II. IBM's qubit of choice is the superconducting semiconductor charge qubit known as the transmon [2], [3], [4].

TABLE II
IBM Q Device Details

Name	Release Date	Qubits Supported	Coupling Complexity
ibmqx2	Jan. 2017	5	0.3
ibmqx3	June 2017	16	$0.08\overline{3}$
ibmqx4	Sept. 2017	5	0.3
ibmqx5	Sept. 2017	16	$0.08\overline{3}$
simulator	Jan. 2017	20	1

The IBM QCs, the largest containing a total of 16 qubits, have a variety of available qubit gates that can be used within quantum circuit implementations. The following transformation operations are included in the IBM QC gate libray: NOT, Y, Z, H, S, $S^\dagger$, CNOT, $\pi/8$, $\pi/8^\dagger$, phase rotation, amplitude rotation, and measurement. It is common for qubits in physical quantum implementations to only be able to couple with other system qubits within a limited range, and this severly limits the total number of multi-qubit interactions that can occur [8]. The IBM machines demonstrate this characteristic since the CNOT gate is the only available two-qubit gate, and its implementation is restricted to a coupling map set by the connectivity of the transmon qubits on the device. The connectivity maps for the four IBM devices are as follows where a:[b] indicates CNOT operator control:[target(s)] qubit(s) that may be used in a circuit:

- ibmqx2 - 0: [1, 2], 1: [2], 3: [2, 4], 4: [2]
- ibmqx3 - 0:[1], 1: [2], 2: [3], 3: [14], 4: [3,5], 6: [7,11], 7: [10], 8: [7], 9: [8,10], 11: [10], 12: [5,11,13], 13: [4,14], 15: [0,14]
- ibmqx4 - 1: [0], 2: [0,1,4], 3: [2,4]
- ibmqx5 - 1:[2], 2:[3], 3:[4,14], 5:[4], 6:[7,11], 7:[10], 8:[7], 9:[8,10], 11:[10], 12:[5,11,13], 13:[4,14], 15:[0,2,14]

The IBM simulator can host a maximum 20 qubits that are unrestricted by a coupling map. Additional quantum gates are also included.

A metric referred to as the coupling complexity, shown in Table II, was calculated for each device to give the QC user insight to the available qubit connections within each machine. Coupling complexity is defined as the ratio of the number of allowable CNOT couplings found in the coupling map to the total number of two-qubit permutations for a quantum machine. For example, the ibmqx2 machine has 6 couplings on the coupling map and a total of 20 two-qubit permutations as shown by

$$\text{qx2 coup. cplx.} = \frac{\text{6 available couplings}}{\text{20 total coupling permutations}} = 0.3.$$

A coupling complexity close to one indicates that a high percentage of a quantum machine's qubits can be configured in arbitrary two-qubit CNOT operations. A coupling complexity close to zero indicates that a low percentage of the qubits can be coupled for a CNOT gate.

Circuits, or algorithms, can be loaded onto the IBM QCs and input to the simulator via an available Python API called QISKit (Quantum Information Software Kit) [7]. Circuits can also be made using the composer GUI on [6].

IV. CNOT Reroute Methodology

Quantum hardware synthesis must perform decomposition on general quantum algorithms so that they are transformed into forms that only use operators available in the target technology's gate library. The technology-independent quantum specification may contain multi-qubit gates that are not realizable on any physical implementation, but this does not prevent the circuit from being transformed into a platform-compatible form. This transformation can be done because all arbitrary n-qubit gates are capable of being decomposed into a collection of single qubit gates as well as the two-qubit CNOT gate [5]. Assuming that the decomposition algorithms are available, the challenge exists in placing the operators in correct locations where they will be able to execute for QIP. As described in the Section III, a serious drawback of the IBM solid-state QCs is that the qubits are in fixed positions. These stationary qubits are able to perform limited CNOT operations due to the physical device's layout and operational characteristics. For this reason, it is important to have reroute algorithms that allow the CNOT operation to be performed on uncoupled qubits on any architecture specified by a coupling map. Two algorithms were developed that satisfy this need:

the adjacent SWAP qubit reroute, and the connectivity tree qubit reroute. These algorithms were successfully applied to different IBM machines with unique coupling requirements for the CNOT operaton.

A. Adjacent SWAP Reroute Algorithm

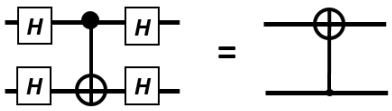


Fig. 2. CNOT orientation reversal

The adjacent SWAP reroute algorithm (ASR) allows the control qubit of the CNOT operation to be moved to a location where the CNOT can execute according coupling map of the architecture. This algorithm depends on the physical implementation supporting coupling between all adjacent qubits. If a CNOT can be performed in one direction, then it can be inverted with the use of four Hadamard gates according to the transformation from [1]. These equivalent circuits can be seen in Fig. 2. Since the transformation pictured in Fig. 2 is available, the direction of the coupling does not matter for ASR. As long as a coupling relationship exists between all adjacent device qubits, the ASR algorithm works. The path that a control qubit travels during the SWAP operations is direct and is referred to as the "SWAP path." The SWAP path proceeds from qubit to qubit in an order according to the qubit indices on the IC. As the qubit propagates from one location to another, the distance to the target qubit decreases until a pair of coupled qubits is found. As seen in Fig. 3, the SWAP operation is implemented with three CNOT gates among physically adjacent qubits, causing the interchange of quantum information. After the swapping operations occur and the CNOT operation of interest executes, the target qubit traverses the SWAP path in reverse until it returns to its original location.

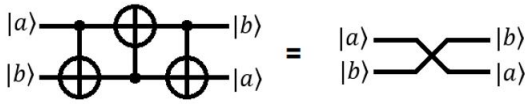


Fig. 3. Implementation of SWAP operation using CNOT

The the general expression for the transfer function that governs the SWAP path was derived for this reroute algorithm. The transfer function for the n required SWAPs to reroute a CNOT operation is generated using

$$\sum_{i=0}^{2^n-1} |2i\rangle\langle i| + \left|2^{n+1}-2i-1\right\rangle\left\langle 2^{n+1}-i-1\right|. \quad (3)$$

In Eqn. 3, the resulting integer value (denoted in bold font) in the Dirac Bra or Ket must be converted to its corresponding value in binary. Smaller values must be padded with zeros so that all vectors contain the same number of qubits. The variable n represents the number of SWAP operations needed for a desired CNOT to be implemented according to the coupling map, and it must be greater than or equal to 1. This minimum boundary case effectively calculates the SWAP operation's transfer function between two qubits. As a demonstration, the SWAP gate transformation matrix will be derived using Eqn. 3 for the situation where a qubit must be swapped once, $n = 1$, to reach a position where a CNOT operation can be performed.

$$\sum_{i=0}^{2^1-1} |2i\rangle\langle i| + \left|2^{1+1}-2i-1\right\rangle\left\langle 2^{1+1}-i-1\right|$$

$$|\mathbf{0}\rangle\langle\mathbf{0}| + |\mathbf{3}\rangle\langle\mathbf{3}| + |\mathbf{2}\rangle\langle\mathbf{1}| + |\mathbf{1}\rangle\langle\mathbf{2}| =$$

$$|00\rangle\langle 00| + |11\rangle\langle 11| + |10\rangle\langle 01| + |01\rangle\langle 10| =$$

$$\begin{bmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix} + \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&0\end{bmatrix} +$$

$$\begin{bmatrix}0&0&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&0\end{bmatrix} = \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} = \mathbf{SWAP}$$

This example is illustrated in Fig. 4.

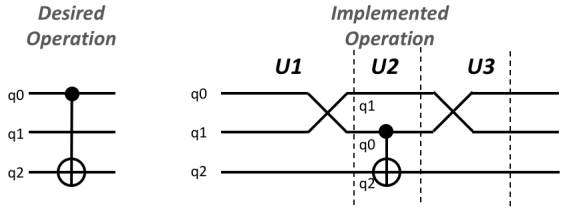


Fig. 4. Example of CNOT implementation using SWAP operation for $n = 1$

The general form of the SWAP path transfer function in Eqn. 3 can also be used to calculate more complex transformation that represents multiple SWAPs needed for CNOT execution. For example, consider the case in Fig. 5 where a qubit must be swapped twice for a CNOT operation. In this example, q0 and q1 cannot be coupled with q3, but q2 can.

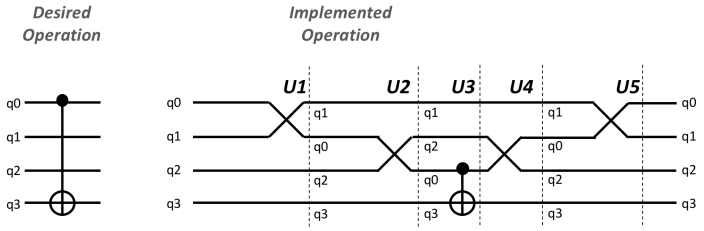


Fig. 5. Example of CNOT implementation using SWAP operation for $n = 2$

$$\sum_{i=0}^{2^n-1} |2i\rangle\langle i| + \left|2^{n+1}-2i-1\right\rangle\left\langle 2^{n+1}-i-1\right|; n = 2$$

$$\sum_{i=0}^{2^2-1} |2i\rangle\langle i| + |2^3 - 2i - 1\rangle\langle 2^3 - i - 1| =$$

$$|\boldsymbol{0}\rangle\langle\boldsymbol{0}| + |\boldsymbol{7}\rangle\langle\boldsymbol{7}| + |\boldsymbol{2}\rangle\langle\boldsymbol{1}| + |\boldsymbol{5}\rangle\langle\boldsymbol{6}| +$$

$$|\boldsymbol{4}\rangle\langle\boldsymbol{2}| + |\boldsymbol{3}\rangle\langle\boldsymbol{5}| + |\boldsymbol{6}\rangle\langle\boldsymbol{3}| + |\boldsymbol{1}\rangle\langle\boldsymbol{4}| =$$

$$|000\rangle\langle 000| + |111\rangle\langle 111| + |010\rangle\langle 001| + |101\rangle\langle 110| +$$

$$|100\rangle\langle 010| + |011\rangle\langle 101| + |110\rangle\langle 011| + |001\rangle\langle 100| =$$

$$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\end{bmatrix} +$$

$$\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix} +$$

$$\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix} +$$

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

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

This solution can be confirmed by finding the transfer function for the swapping function pictured in Fig. 5 manually. The transfer function that models the movement of qubit q0 down to the position of q2 is desired. Therefore, the top three qubits will be included from the circuit in the transfer function derivation. To begin the derivation, the matrices for **U1** and **U2** must be found.

$$\mathbf{U1} = \mathbf{SWAP} \otimes \mathbf{I} = \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} \otimes \begin{bmatrix}1&0\\0&1\end{bmatrix} =$$

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

$$\mathbf{U2} = \mathbf{I} \otimes \mathbf{SWAP} = \begin{bmatrix}1&0\\0&1\end{bmatrix} \otimes \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix} =$$

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

Now **U2** and **U1** must be combined with a matrix product from right to left to determine the transfer function needed to swap q0 two places:

$$\mathbf{U2U1} =$$

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

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

The transfer function general form evaluation for $n = 2$ matches the manual transformation matrix derivation, showing that Eqn. 3 is scalable.

B. Connectivity Tree Reroute Algorithm

Smaller QCs are likely to offer adjacent coupling between all qubits for two-qubit operations, such as with the case of the 5-qubit IBM QCs, but as the machines grow in size, the likelihood of available coupling between all neighboring qubits decreases. For example, ibmqx5 prohibits CNOT coupling between q0 and q1 and between q5 and q6. This lack of connectivity prevents the ASR from working if a CNOT operation spans these qubits. For this reason, an alternative CNOT rerouting algorithm, connectivity tree reroute (CTR), was developed. In this method, tree structures based on the coupling map for the QC determine the SWAP path that the control qubit travels to reposition for a CNOT. The tree root vertex is the control qubit, and edges leading to other qubit vertices are generated according to available coupling configurations. The tree continues to grow, describing all possible paths that the qubit can take, until a path to a position

coupled with the target qubit is found. During tree creation, if a vertex is reached that is already represented in the tree, it is discarded.

Compared to the ASR technique, the connectivity tree, as illustrated in Fig. 6, often reduces the number of SWAP operations needed to implement a CNOT operation between originally uncoupled qubits. In Fig. 6, the desired operation is a CNOT with q0 as the control and q4 as the target on the ibmqx2 machine. CTR finds a shorter SWAP path, leading to fewer gates and hence, reduced quantum cost, in the final mapping. Because of the way the connectivity tree grows according to the connectivity map, the tree yields a path to connect qubits that may not be possible to connect with the ASR algorithm.

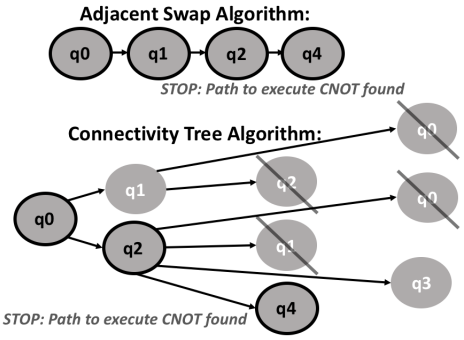


Fig. 6. Adjacent SWAP vs. Connectivity Tree for CNOT with q0 control and q4 target on the ibmqx2 machine

Another example of implementing a CNOT operation with connectivity trees can be seen in Fig. 7. In this second example, the desired operation is a CNOT with q5 as the control and q10 as the target on the 16-qubit ibmqx5 machine. The operation is able to be performed after 2 swaps of q5 with q12 then q12 with q11 since a connection between q11 and q10 exists on the coupling map. The adjacent swap algorithm would not work in this for this operation. According to the ibmqx5 coupling map, q5 and q6 cannot perform a CNOT operation together. As a result, q5 cannot swap in a direct path of increasing qubit index to find a position where a CNOT operation with q10 as the target can execute.

V. Conclusion

Many of the emerging QC machines are only able to process circuits that include operations from a fixed set of gates. In addition to this constraint, these devices have limitations with respect to interactions between qubits for multi-qubit operations. For example, the IBM Q devices can implement the CNOT operation, but only certain qubits can control others according to the coupling map. Coupling maps make the process of mapping technology-independent circuits into device-compatable forms tedious, creating a demand for automated mapping methods. In this work, methods for rerouting arbitrarily-placed CNOT operations on quantum devices with fixed coupling maps were explored. The ASR and CTR techniques assist in the process of mapping technology-independent circuits into

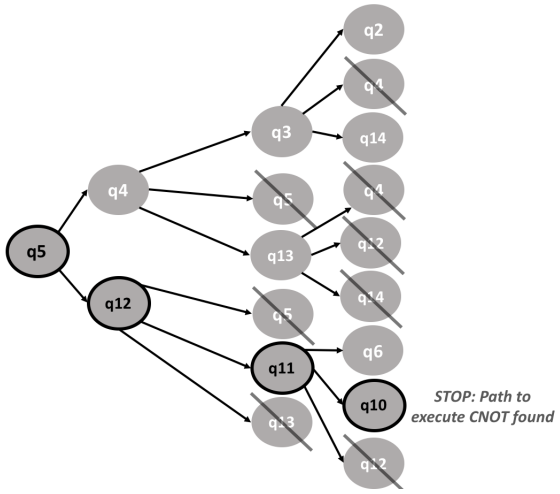


Fig. 7. Connectivity Tree for CNOT with q5 control and q10 target on the ibmqx5 machine

forms that are executable on real machines, and these methods can be included in the flow of quantum logic synthesis. Future work for the automated mapping methods presented in this paper includes their implementation in a technology-dependent quantum logic synthesis and compilation tool.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

[2] J. M. Chow, J. M. Gambetta, E. Magesan, S. J. Srinivasan, A. W. Cross, D. W. Abraham, N. A. Masluk, B. R. Johnson, C. A. Ryan, J. A. Smolin, and M. Steffen, "Implementing a strand of scalable fault-tolerant quantum computing fabric," *Nature Communications*, vol. 5, June 2014.

[3] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square latice of four superconducting qubits," *Nature Communications*, vol. 5, 2015.

[4] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, "Experimental demonstration of fault-tolerant state preparation with superconducting qubits," *arXiv preprint arXiv:1705.09259*, 2017.

[5] 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, no. 5, 1995.

[6] IBM Quantum Experience. http://research.ibm.com/quantum/.

[7] https://github.com/QISKit

[8] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Information Processing*, vol. 10, no. 3, pp. 355-377, 2011.