MUSTANG-Q: A Technology Dependent Quantum Logic Synthesis and Compilation Tool

Kaitlin N. Smith and Dr. Mitchell A. Thornton

Abstract-As quantum information processing devices become a reality, there is increased motivation to investigate and realize algorithms to enable automated translation of quantum processing algorithms into forms consistent with emerging device libraries. In the past few years there has been some progress in devising automated methods for the conversion of classical irreversible specifications into reversible counterparts, however, the final products of these synthesis procedures are technology independent implementations of algorithms that are not specific to a target technology. In this work, a synthesis and compilation tool, MUSTANG-Q, that maps technology independent forms of quantum circuit operators to a QASM description targeted for a specific physical implementation is presented. MUSTANG-Q is a prototype tool that is capable of serving as an automated quantum circuit synthesis tool for a specific library of technologically-dependent quantum operators or "gates.'' Alternatively, MUSTANG-Q can be used as a compiler for specific implementations of general quantum computers. The IBM Q family of quantum computers are used in this work as example target technologies.

Important QIP Concepts

The Qubit, $|\Psi\rangle$:

- Theoretically an infinite set of states possible with superposition: $|\Psi\rangle = \alpha|0\rangle + \alpha|1\rangle$
- Superposition collapses after measurement or decoherence

Physical Qubit Classifications:

- Atomic Scale – particles, atoms, ions, quasi-particles, etc.
- Mesoscopic – superconducting solid-state realization

Quantum Operations:

- Qubits are transformed by unitary "gates"
- All quantum transforms can be decomposed into one- and two-qubit operations

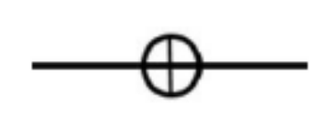
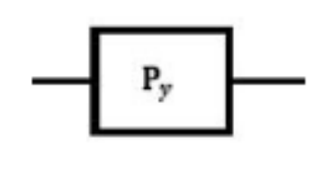
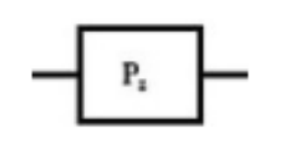
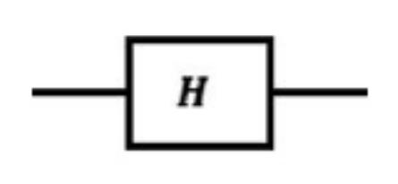
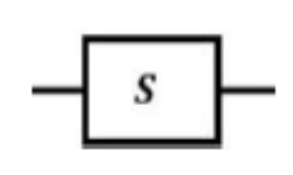
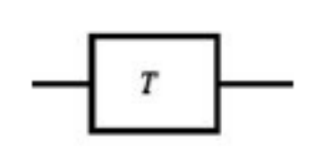
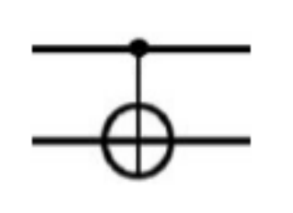
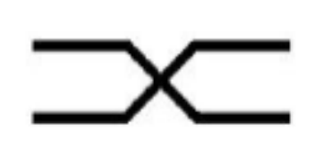
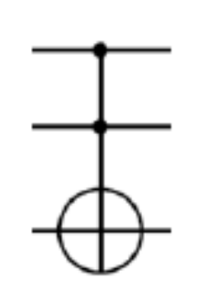
Quantum Cost:

- Costly quantum circuits have greater chance of state decoherence in qubits
- Cost of a quantum circuit typically equated to number of required CNOT and single-qubit gates

Reversible Logic:

- To implement a classical algorithm as quantum, it must be converted to a reversible form
- ESOP reversible logic synthesis algorithms can transform switching functions into Toffoli cascades

COMMON SINGLE- AND MULTI-QUBIT QUANTUM OPERATORS

Operator	Symbol	Transfer 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)		$\begin{bmatrix}1 & 0\\0 & ie^{i\pi/4}\end{bmatrix}$
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		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

IBM Q: A Physical QC Implementation

IMB Q MACHINE DETAILS

Name	Release Date	# Qubits	Coupling Map (controls:[target(s)])
ibmqx2	Jan. 2017	5	0: [1, 2], 1: [2], 3: [2, 4], 4: [2]
ibmqx3	June 2017	16	1: [2], 2: [3], 4: [3], 4: [5], 6: [7], 8: [7], 9:[10], 11: [10], 12: [11], 12: [13], 13:[14], 15: [14], 15:[0], 0: [1], 3: [14], 13: [4], 12: [5], 6: [11], 7: [10], 9: [8]
ibmqx4	Sept. 2017	5	1: [0], 2: [0, 1, 4], 3: [2, 4]
ibmqx5	Sept. 2017	16	1:[2], 2:[3], 3:[4], 3:[14], 5:[4], 6:[7], 6:[11], 7:[10], 8:[7], 9:[8], 9:[10], 11:[10], 12:[5], 12:[11], 12:[13], 13:[4], 13:[14], 15:[0], 15:[2], 15:[14]
simulator	Jan. 2017	20	n/a

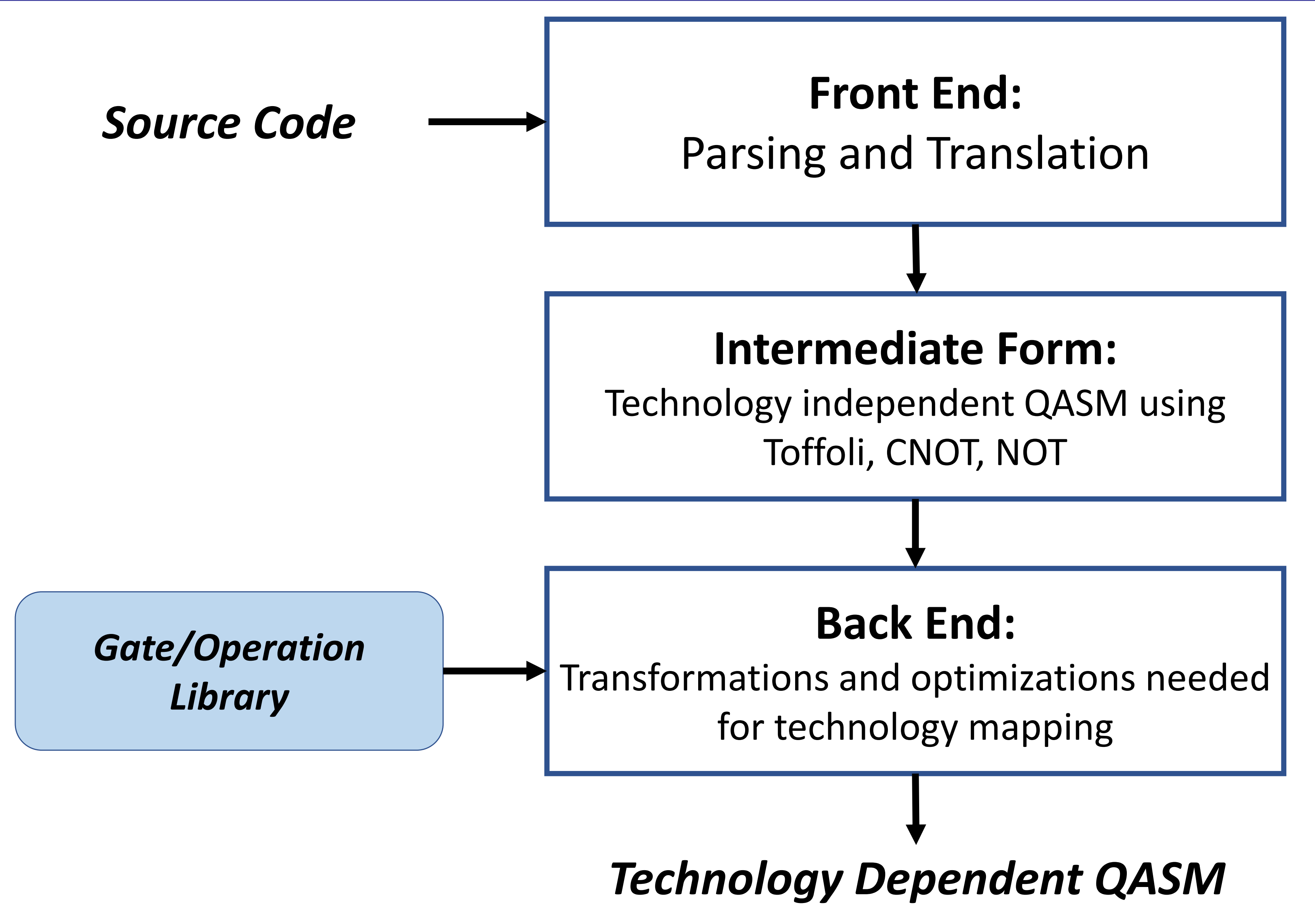
Transmons

- IBM's qubit of choice
- A type of superconducting qubit that uses charge as the information carrier
- Operational characteristics and proximity of qubits allows coupling, restricting multi-qubit operations to coupling maps

The IBM QCs

- Available gates: Identity, Pauli-X, Pauli-Y, Pauli-Z, Hadamard, Phase, Phase †, CNOT, $\pi/8$, $\pi/8^\dagger$, phase rotation, amplitude rotation, and measurement
- All devices characterized by unique coupling map for where CNOT can be implemented
- Circuits can be loaded into the IBM QCs with a Python API known as QISKit or with the composer GUI on the IBM web page

MUSTANG-Q Architecture

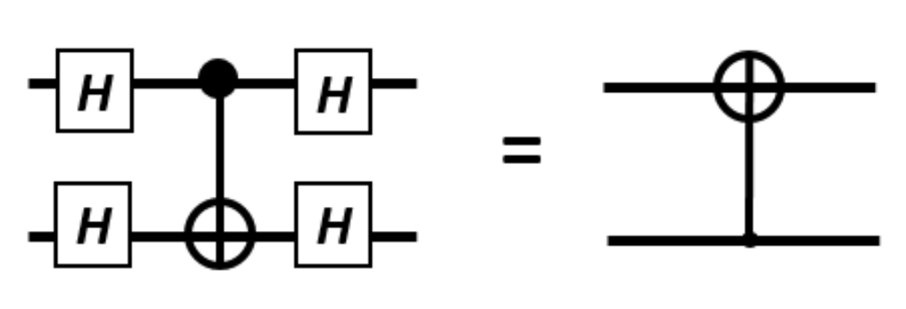

Source Code
Front End:
Parsing and Translation
Intermediate Form:
Technology independent QASM using Toffoli, CNOT, NOT
Gate/Operation Library
Back End:
Transformations and optimizations needed for technology mapping
Technology Dependent QASM

Quantum Logic Synthesis Methodology

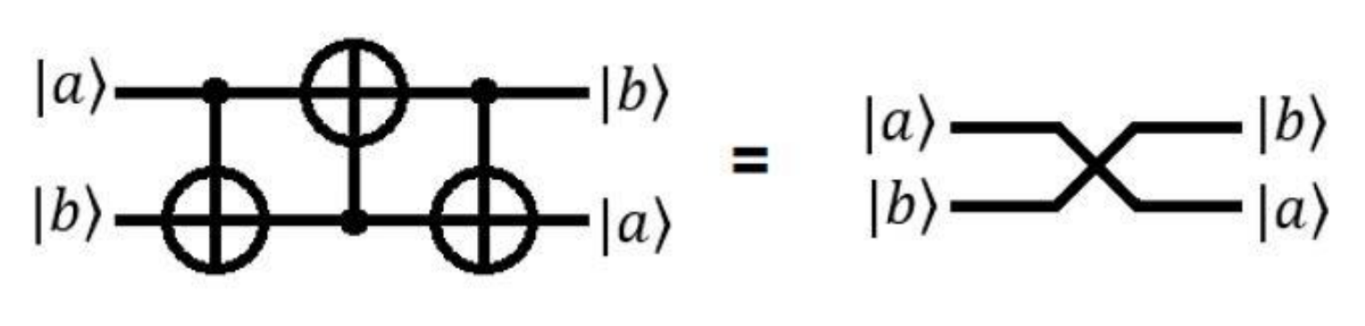
Currently Implemented Back-End Optimizations:

1. If a CNOT can be performed in one direction, then it can be inverted with the use of four Hadamard gates
2. If a CNOT operation is performed on two qubits not included directly or in reverse on the coupling map, a reroute algorithm implementing SWAP operations between coupled qubits will occur until the two qubits are in a position where the CNOT operation can evaluate. After evaluation, the qubits are moved back to their original position
3. Toffoli operations on 3 qubits can be decomposed into the Clifford+T, S library in order to be compatible with the available operators

Useful Identities:



CNOT direction reversal



CNOT implementation of SWAP

CNOT Reroute Algorithms using SWAP

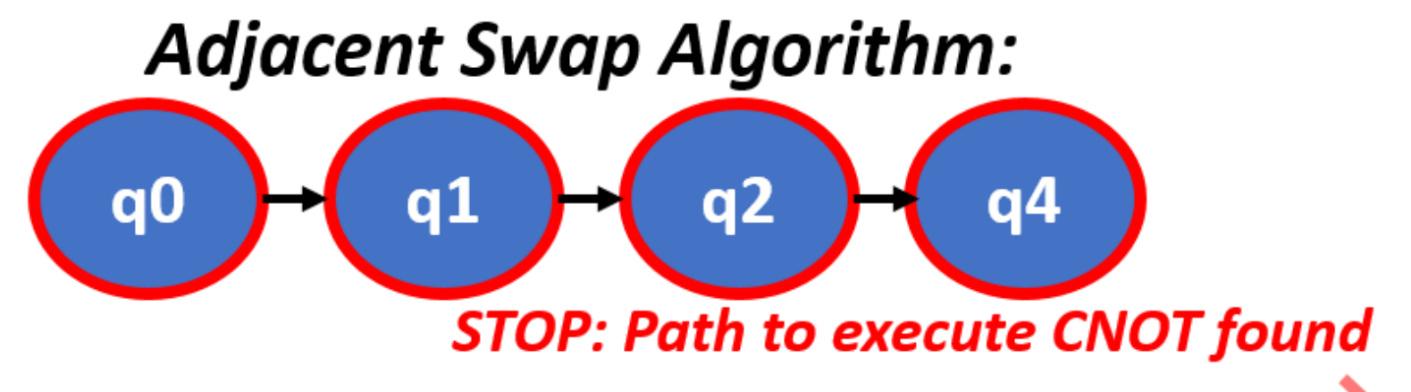
We have developed two algorithms that allow a CNOT operation between uncoupled qubits to be realized. Algorithms can be used on any architecture where a coupling map is provided. A general expression for the transfer function that governs the SWAP path was derived:

$$\sum_{i=0}^{2^n-1} |2i\rangle\langle i| + |2^{n+1}-2i-1\rangle\langle 2^{n+1}-2i-1|, n = \#\, req.SWAPs\ for\ CNOT\ reroute$$

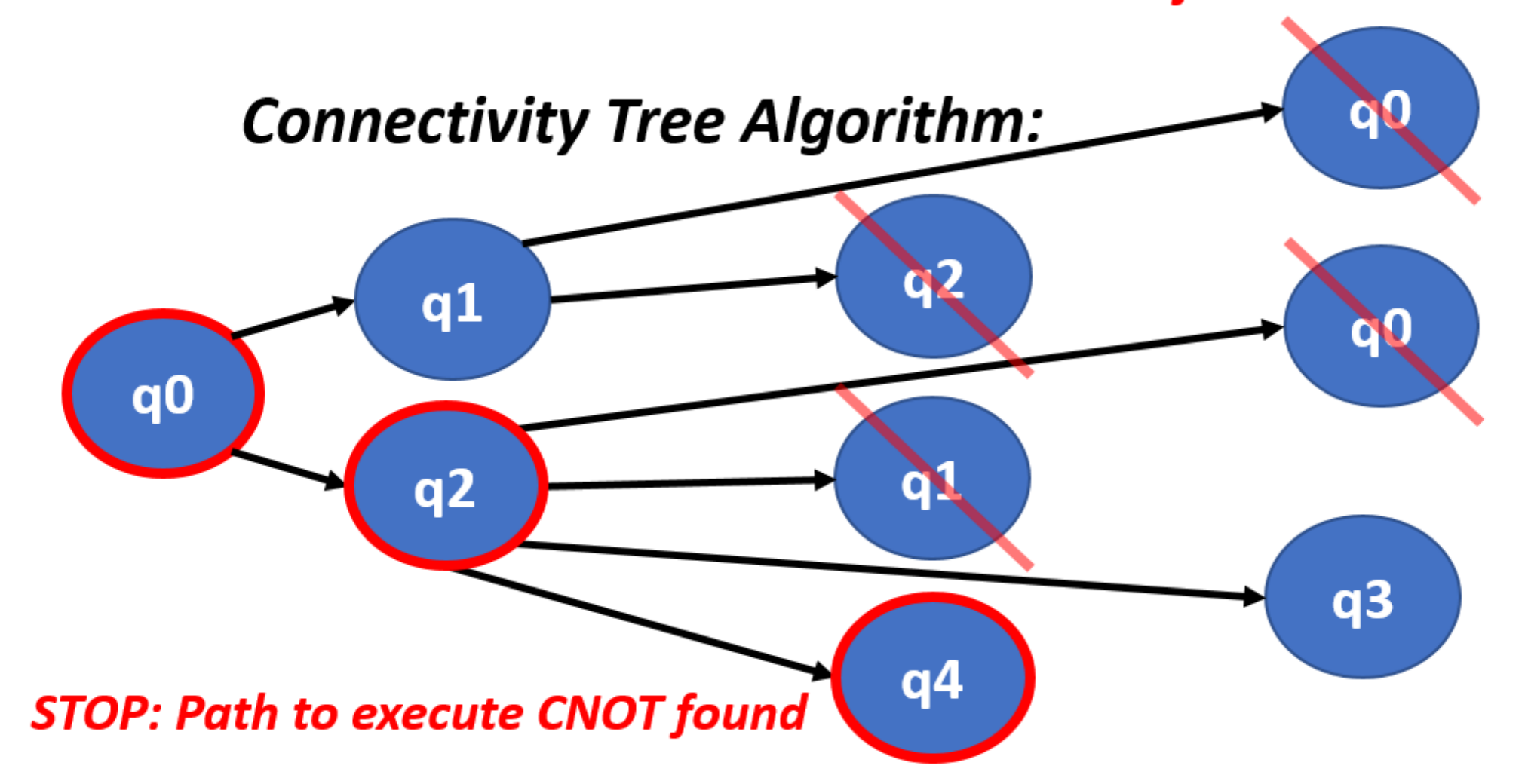
Adjacent SWAP Reroute – Control qubit swaps location with adjacent qubits, moving towards the target qubit until CNOT can evaluate. Algorithm depends on physical implementation supporting coupling between all adjacent qubits.

Connectivity Tree Reroute – SWAP path for control qubit found with a tree based on the coupling map. Root is the control, and edges leading to other qubits are generated according to available couplings, describing all paths the qubit can take until shortest path to a CNOT coupling with target is found.

Adjacent Swap Algorithm:



Connectivity Tree Algorithm:



Experimental Results

CNOT CIRCUITS MAPPED FOR 5-QUBIT IBMQX2 DEVICE

Circuit	# CNOT w/ Adj. SWAP	# Gates w/ Adj. SWAP	# CNOT w/ Conn. Tree	# Gates w/ Conn. Tree
cx q0 q1	1	1	1	1
cx q0 q2	1	1	1	1
cx q0 q3	13	33	7	19
cx q0 q4	13	33	7	19
cx q1 q0	1	5	1	5
cx q1 q2	1	1	1	1
cx q1 q3	7	19	7	19
cx q1 q4	7	19	7	19
cx q2 q0	1	5	1	5
cx q2 q1	1	5	1	5
cx q2 q3	1	5	1	5
cx q2 q4	1	5	1	5
cx q3 q0	7	19	7	19
cx q3 q1	7	19	7	19
cx q3 q2	1	1	1	1
cx q3 q4	1	1	1	1
cx q4 q0	13	41	7	19
cx q4 q1	13	41	7	19
cx q4 q2	1	1	1	1
cx q4 q3	1	5	1	5

ORIGINAL COMPOSITION OF TOFFOLI CASCADE BENCHMARKS

Circuit	# Qubits	# Toffoli	# CNOT	# NOT	# Gates
3-17-14.real	3	2	3	1	6
4-49-17.real	4	5	5	2	12
fredkin-6.real	3	3	0	0	3
half-adder	3	1	1	0	2
full-adder	4	2	2	0	4

TOFF. CASCADE BECH. METRICS AFTER MUSTANG-Q MAPPING TO IBM DEVICES (#CNOT/#GATES & AVG. TIME/10000 RUNS)

Circuit	ibmqx2	ibmqx3	ibmqx4	ibmqx5
3-17-14.real	45/142	45/190	45/122	45/190
	3.826ms	4.974ms	3.694ms	4.854ms
4-49-17.real	113/365	203/607	113/405	203/599
	8.293ms	15.63ms	8.504ms	15.46ms
fredkin-6.real	54/172	54/156	54/228	54/156
	4.144ms	4.483ms	4.59ms	3.919ms
half-adder	19/61	19/45	19/89	19/45
	1.902ms	2.074ms	2.286ms	1.742ms
full-adder	26/98	62/178	26/102	62/146
	2.622ms	5.124ms	2.338ms	5.109ms

Algorithm Implementation and Test:

- Back-end of MUSTANG-Q compiler developed in Python
- CNOT and reversible Toffoli cascade circuits used as benchmarks for logic synthesis to test CNOT reroute and Toffoli decomposition capabilities
- Technology dependent synthesized circuits verified for correct logic on IBM QC machines
- Connectivity tree reroute found to be most effective algorithm for finding SWAP path for arbitrarily placed CNOT, especially on larger architectures where adjacent coupling does not always exist

MUSTANG-Q Future Work:

- Addition of coupling maps for new architectures as they become available
- Addition of decompositions for other controlled gates
- Improved optimizations to reduce quantum cost