

Quantum Logic Synthesis with Formal Verification

Kaitlin N. Smith and Mitchell A. Thornton
Department of Electrical and Computer Engineering
Southern Methodist University
Dallas, TX USA
{knsmith,mitch}@smu.edu

Abstract—Quantum computers capable of practical information processing are emerging rapidly. As these devices become more advanced, tools will be needed for converting generalized quantum algorithms into formally-verified forms that are executable on real quantum machines. In this work, a prototype tool is presented that transforms quantum algorithms into executable specifications where optimization procedures yield 9-24 % cost improvement on a range of benchmarks. Additionally, the tool incorporates formal verification internally with Quantum Multiple-valued Decision Diagrams to confirm that the generated technology-dependent executable is functionally equivalent to the original, technology-independent algorithm. Experimental results are provided that target the Rigetti family of quantum processing units although the tool may also target other architectures.

Index Terms—quantum information processing, quantum computing, quantum logic synthesis, technology mapping

I. INTRODUCTION AND BACKGROUND

Many quantum processing units (QPUs) have been developed, but these devices have differences in the types of quantum circuits they can execute. Because of the variety between QPUs, even between those of the same technology, methods must be developed that allow arbitrary quantum circuits to be mapped to technology-dependent models. When mapping circuits into a technology-dependent form, restrictions with respect to available operators and qubit topology must be considered. In this work, a quantum logic synthesis tool that compiles circuits to the Rigetti QPUs will be discussed. Output circuits are optimized for the target architectures through the minimization of a user-specified cost function. Although the problem of transforming quantum circuits into primitive gate sets required for QPU execution [1] as well as quantum mapping algorithms [2], [3] have been explored in the past, this work differs by performing compilation procedures for the Rigetti devices along with implementing formal verification. Accuracy of the output specifications is confirmed with equivalence checking via the Quantum Multiple-valued Decision Diagram.

A. Quantum Computation

Quantum units, qubits, are modeled as a combination of the basis states $|0\rangle = [1 \ 0]^T$ and $|1\rangle = [0 \ 1]^T$ as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \quad (1)$$

In Eqn. (1), α and β are complex-valued probability amplitudes in the state vector. During measurement, the probability

that $|\psi\rangle = |0\rangle$ is $\alpha^*\alpha = |\alpha|^2$ and the probability that $|\psi\rangle = |1\rangle$ after measurement is $\beta^*\beta = |\beta|^2$ where * indicates the complex conjugate and $|\alpha|^2 + |\beta|^2 = 1$. Qubits collapse into a basis state after measurement, losing all superposition and entanglement properties [4].










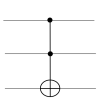
An operator for quantum information processing (QIP) is represented by a unitary transformation matrix, U of dimension $2^n \times 2^n$ where n is the number of transformed qubits. To create a transfer function that describes an entire circuit, parallel quantum operators are combined with a tensor product while operators in series are combined with a matrix product. A common set of single- and multi-qubit operations are provided in Table I.

Some quantum computers (QCs), depending on the technology platform, are more resilient to decoherence, or accidental measurement, as compared to others. In QIP circuits and QC design, there is a trade-off between average decoherence times and qubit coupling. Multi-qubit interaction is necessary to enable required operations for QIP such as the **CX** or **CZ**. The ability for qubits to couple, however, implies that qubits also have the undesirable capability to interact with the environment thus increasing the probability of decoherence. Determining qubit realizations that allow for 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 QIP realization efforts are focused upon the class of superconducting solid-state qubits as they are considered by many to offer the best trade off between qubit interaction versus average time of decoherence. At the time of this writing, the community considers that we are in the era of noisy intermediate-scale quantum (NISQ) computing [5].

B. The Rigetti Corporation Quantum Computers

Rigetti has developed three solid-state-circuit-based QCs named Agave, Aspen, and Acorn [6]–[8]. The Python library PyQuil as well as a software development kit (SDK) ForestTM are used for writing quantum algorithms, interacting with the quantum devices, and simulating quantum computing [9]. To specify an algorithm for the Rigetti QCs, quantum operators must be specified in quantum instruction language, or Quil [10]. Although Quil allows for the specification of algorithms using many of the standard quantum gates, for algorithm to be executable on a real QPU, a native gate set

TABLE I
COMMON SINGLE- AND MULTI-QUBIT QUANTUM OPERATORS

Operator	Symbol	Transfer Matrix
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
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)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
CX		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
CZ		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

must be used. This set includes **RZ** rotations, **RX** rotations that are integer-multiples of $\pi/2$, and **CZ** [9].

Not every qubit pair can couple on a QPU, and this severely limits the total number of executable multi-qubit operations. The Rigetti devices demonstrate this operational constraint because the only available two-qubit gate, **CZ**, may only be implemented on adjacent qubits that are connected. Therefore, the device topology prevents the execution of arbitrary **CZ** operations in an algorithm. The device topology for the Rigetti QPUs can be represented as dictionaries where **device** = $\{a_0 : [b_0], a_1 : [b_1], \dots, a_{n-1} : [b_{n-1}]\}$. In these dictionaries, the keywords, a_i , are qubits that can act as **CZ** controls and the paired list, b_i , indicates which qubit(s) that the **CZ** control can target [11]:

- **Agave (8 qubits)** = $\{ 0:[1,7], 1:[0,2], 2:[1,3], 3:[2,4], 4:[3,5], 5:[4,6], 6:[5,7], 7:[0,6] \}$
- **Aspen (16 qubits)** = $\{ 0:[1,7], 1:[0,2,16], 2:[1,3,15], 3:[2,4], 4:[3,5], 5:[4,6], 6:[5,7], 7:[0,6], 10:[11,17], 11:[10,12], 12:[11,13], 13:[12,14], 14:[13,15], 15:[2,14,16], 16:[1,15,17], 17:[10,16] \}$
- **Acorn (19 qubits)** = $\{ 0:[5,6], 1:[6,7], 2:[7,8], 4:[9], 5:[0,10], 6:[0,1,11], 7:[1,2,12], 8:[2,13], 9:[4,14], 10:[5,15,16], 11:[6,16,17], 12:[7,17,18], 13:[8,18,19], 14:[9,19], 15:[10], 16:[10,11], 17:[11,12], 18:[12,13], 19:[13,14] \}$

An interesting observation is that the **CZ** gate is bidirectional in the sense that the transformation matrix is equivalent if the control and target qubits are interchanged.

C. Quantum Cost

Since the likelihood of decoherence increases as a set of qubits are continually transformed, when considering quantum cost reduction, the most common method is usually centered around reducing the total number of operations performed on a set of qubits, especially when operations are multi-qubit. Performance parameters for one- and two-qubit operators for the Rigetti machines can be found in [11]. Based upon this information we formulate the cost function used by our synthesis tool for the Rigetti QPUs as

$$q_{cost} = 5c + 3h + 2y + x + z + s + t. \quad (2)$$

In Eqn. (2), c is the count of **CZ** gates, h is the count of **H** gates, y is the count of **Y** gates, x is the count of **X** gates, s is the count of **S** and **S**[†] gates, and t is the count of **T** and **T**[†] gates for the technology-dependent circuit mapping generated by the tool. **X**, **Z**, **S**, and **T** are given the lowest weights in the cost function because these gates can be executed with single **RX** or **RZ** gates native to the Rigetti library. **H** and **Y** gates are given weights of 3 and 2, respectively because $H = RZ(\pi/2)RX(\pi/2)RZ(\pi/2)$ and $Y = RZ(\pi)RX(\pi)$ are the transformations used to decompose the single qubit gates into the Rigetti gate set. Finally, **CZ** is given a weight five times that of the gates that are purely **X** and **Z** rotations function because as a two-qubit gate, it has on average longer execution times with a lower fidelity.

D. Quantum Multiple-valued Decision Diagrams

The transfer matrix describing an quantum circuit grows exponentially as the number of qubits in the function increases. The Quantum Multiple-valued Decision Diagram (QMDD) allows these matrices to be represented in a compact form. The QMDD represents quantum functions efficiently as a directed acyclic graph. This structure was first introduced in [12].

QMDDs are a collection of vertices and directed edges. Non-terminal vertices represent qubits and have four outgoing edges that represent one of the four quadrants in a quantum transformation matrix. From left to right, the four edges leaving a non-terminal node represent the sub matrices of U_{00} , U_{01} , U_{10} , and U_{11} for the quantum transformation matrix U . Redundancy in the QMDD is forbidden, and each qubit variable may only appear once in a path. These rules cause the QMDD to become very compact. An example of the **CX** operation represented as a QMDD is shown in Fig. 1 (x_0 represents the control and x_1 represents the target). The variable order is $x_0 \rightarrow x_1$, so x_0 acts as the initial vertex and x_1 vertices appear afterwards along the path for submatrices that are not equal to the constant zero.

QMDD representations are canonical with respect to a fixed variable order [12]. Even if two circuits with the same transformation matrix are described using different operators, their QMDDs will be equivalent as long as the variable order is maintained. This concept is used to perform formal verification within the quantum logic synthesis tool. An equivalence check between the original technology-independent quantum circuit

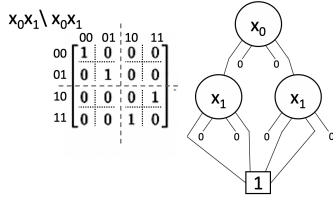


Fig. 1. Representation of CX Operation as a QMDD.

and the resulting technology-dependent mapped circuit is ensured by requiring the QMDDs to match. If the realizations are equivalent, the reduction rules for the QMDDs allow the specifications to share the same graph in memory. This is important in the current era of NISQ devices since errors can be attributed to decoherence since the executable has been formally verified.

II. QUANTUM LOGIC SYNTHESIS METHODOLOGY

The prototype of this work transforms technology independent quantum circuits into technology-dependent Quil specifications. Initially, the original circuit is parsed in as source code, and it contains a variety of operators that may or may not be supported by the target architecture. The tool that is the subject of this paper is considered a back-end portion of a quantum compiler, and it performs transformations and optimizations needed for technology mapping to a specific QPU. Multi-qubit gate decomposition algorithms, such as those given by Barenco et al. in [13], are representative of some of the transformations implemented in this tool. Additional optimizations and algorithms were also developed to accommodate for QPU topological differences that limit qubit connections for two-qubit operations.

A significant drawback of solid-state qubit QPUs is that the stationary qubits are limited to certain multi-qubit operations due to the layout and physical properties of the device. For this reason, a generic reroute algorithm capable of implementing multi-qubit operations over uncoupled qubits for any topology is essential for technology-dependent quantum logic synthesis. Tree data structures assist in finding the shortest SWAP route for two-qubit gate execution while preserving the original placement of the algorithm’s qubits on the device.

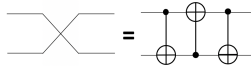


Fig. 2. SWAP implementation using CX.

The CX gate is commonly used in technology-independent circuits, and as shown in Fig. 2, three of the operators in series can form a SWAP operation among physically connected qubits. Due to the simplicity of the SWAP implementation with CX, the connectivity tree reroute algorithm (CTR) in the quantum synthesis tool uses CX to automatically reroute two-qubit operations that are not supported by a coupling map. Of course, as CX is not a supported gate in the Rigetti QPU library, these gates are eventually converted into CZ gates

using the transformation in Fig. 3 after reroute operations have completed. In this method, a tree structure based on the coupling map for the selected QC determines the shortest SWAP path that the control qubit travels to reposition for a two-qubit operation. SWAP operations move the qubit to the desired location and after the required two-qubit operation executes, the control qubit traverses the SWAP path in reverse to return to its original position.

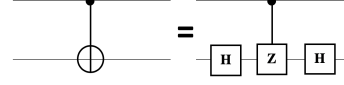


Fig. 3. CX to CZ transformation.

To find the SWAP path, CTR builds a tree data structure where the root node is the control qubit, and edges leading to other qubit nodes are generated according to the available coupling map configurations. The tree describes all possible paths that the qubit can take, until the shortest path to a position coupled with the target qubit is found. If a node is reached that is already represented in the tree, the branch is discarded. Pseudocode describing the CTR algorithm has been included in Fig. 4.

```

CTR(control,target,map):
  tree = control
  2q_op_exe = False
  while 2q_op_exe != True:
    tree = build_branches(tree,map)
    if target in tree:
      path = tree[control:target]
      2q_op_exe = True
  swap_and_exe(control,target,path)
  swap_back(control,target,path)

```

Fig. 4. Pseudocode CTR Algorithm.

Our synthesis approach generates technology-dependent Quil specifications based upon two distinct objectives. The first objective is to produce a quantum algorithm that conforms to the user-specified target QPU’s architectural constraints. The second objective is to determine a mapping that minimizes quantum cost. The quantum cost function is defined in Eqn. (2), and the quantum logic synthesis tool implements the following mapping and optimization procedures:

- 1) Generalized Toffoli gates are decomposed into Toffoli cascades using [13].
- 2) Toffoli operations are decomposed into one and two-qubit operators using transforms from [4], unsupported two-qubit gate placements are rerouted, and CX to CZ transformation occurs.
- 3) Local optimizations based on removing circuit partitions that equal the identity function are implemented recursively until the cost function cannot be further reduced.
- 4) One-qubit operators are decomposed into the gates that are native to the Rigetti operator library.

After all synthesis and optimization procedures complete, the optimized technology-dependent specification is formally verified by performing an equivalence checking test using QMDDs.

TABLE II
METRICS FOR (CZ COUNT/COST) AFTER SYNTHESIS USING
BENCHMARKS FROM [14] TARGETING THE RIGETTI QPUS

Funct.	In form of (unoptimized mapping) (optimized mapping)		
	Agave	Aspen	Acorn
[#] 0001	(1026/11392) (942/9478)	(1026/11392) (942/9478)	(2070/22876) (1860/18736)
[#] 0003	(296/3296) (270/2734)	(296/3296) (270/2734)	(404/4484) (376/3798)
[#] 0007	(1141/12696) (1041/10480)	(1141/12696) (1041/10480)	(3103/34278) (2613/26272)
[#] 0017	(692/7683) (656/6627)	(692/7683) (656/6627)	(1256/13887) (1036/10447)
[#] 001f	(974/10820) (874/8814)	(974/10820) (874/8814)	(1844/20390) (1618/16296)
[#] 003f	(399/4429) (395/4001)	(399/4429) (395/4001)	(453/5023) (403/4077)
[#] 007f	(964/10703) (870/8751)	(964/10703) (870/8751)	(1936/21395) (1600/16073)
[#] 0117	(1856/20641) (1742/17449)	(1856/20641) (1742/17449)	(5426/59911) (4736/47551)
[#] 011f	(674/7486) (626/6322)	(674/7486) (626/6322)	(1670/18442) (1558/15728)
[#] 013f	(1174/13050) (1076/10814)	(1174/13050) (1076/10814)	(2878/31794) (2382/23938)
[#] 017f	(1778/19745) (1594/16023)	(1778/19745) (1594/16023)	(3704/40931) (3168/31771)
[#] 0357	(1390/15432) (1232/12350)	(1390/15432) (1232/12350)	(3088/34110) (2778/27910)
[#] 035f	(488/5423) (454/4593)	(488/5423) (454/4593)	(1670/18425) (1522/15345)

III. EXPERIMENTAL RESULTS

Quantum logic synthesis involving the mapping and optimization of algorithms for the Rigetti QPUs was completed with technology-independent benchmarks from a library titled “Optimal Single-target Gates,” from [14]. These circuits were chosen as benchmarks because they act as essential components for quantum logic synthesis based on other approaches. Complex reversible and quantum circuits decompose into these functions, and in turn, the single-target gates can be decomposed into one- and two-qubit operations. These benchmarks were input into the synthesis tool as technology-independent .qc files that contained single-qubit and CX operations. Because of arbitrary single-qubit gates and two-qubit gate placement within the benchmarks, transformation is required before the algorithms can be executed on the Rigetti QPUs. The benchmarks of [14] range from 3 to 6 qubits in size, and in the interest of testing only the more complex and non-trivial designs, the 6 qubit “Optimal Single-target Gates” were used in experimentation.

When the single-target gate benchmarks were mapped to QPUs, unsupported gate placements were decomposed or rerouted with swapping operations that cause the circuits to expand. After mapping finishes, the resulting circuit may be optimized using built-in local optimizers. Synthesis results of the “Optimal Single-target Gates” mapped to the Rigetti QPUs can be found in Table II. This table includes both pre- and post-optimization metrics for CZ operation count and cost calculated using Eqn. (2) for each of the generated designs.

Referring to Table II, the synthesis results for Agave match those for Aspen. This occurs since these devices have a similar style of ring topology. Acorn, characterized by a grid topology, yields different synthesis results. Information about cost improvement whenever synthesis includes optimization processes is found in Table III. On average, when optimization was performed on a technology-dependent circuit, cost

TABLE III
PERCENT DECREASE IN COST FROM UNOPTIMIZED TO OPTIMIZED
SYNTHESIS TARGETING THE RIGETTI QPUS

Funct.	Agave	Aspen	Acorn
[#] 0001	16.80	16.80	18.10
[#] 0003	17.05	17.05	15.30
[#] 0007	17.45	17.45	23.36
[#] 0017	13.74	13.74	24.77
[#] 001f	18.54	18.54	20.08
[#] 003f	9.66	9.66	18.83
[#] 007f	18.24	18.24	24.87
[#] 0117	15.46	15.46	20.63
[#] 011f	15.55	15.55	14.72
[#] 013f	17.13	17.13	24.71
[#] 017f	18.85	18.85	22.38
[#] 0357	19.97	19.97	18.18
[#] 035f	15.31	15.31	16.72

improved by approximately 17.7%. This improvement in cost is significant because a lower cost indicates that a circuit uses fewer operations and thus executes faster with a decreased probability of decoherence. All outputs of the synthesis runs were verified to be equivalent to their original technology-independent specifications using QMDDs.

IV. CONCLUSION

We have developed a quantum compiler that is capable of targeting the Rigetti family of quantum computers. Our compiler includes formal verification as an inherent feature and is shown to produce optimized results based on both architectural and quantum cost criteria.

REFERENCES

- [1] P. Niemann, R. Wille, and R. Drechsler, “Improved synthesis of Clifford+T quantum functionality,” *Design, Automation and Test in Europe*, 2018.
- [2] K. Smith and M. Thornton, “Automated Mapping Methods for the IBM Transmon Devices,” *International Workshop on Post-Binary ULSI Systems (ULSI-WS)*, 2018.
- [3] A. Cowtan et. al, “On the qubit routing problem,” arXiv:1902.08091, March 2019.
- [4] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.
- [5] J. Preskill, “Quantum Computing in the NISQ era and beyond,” *Quantum*, vol. 2, 2018.
- [6] M. Reagor et. al., “Demonstration of Universal Parametric Entangling Gates on a Multi-Qubit Lattice,” arXiv:1706.06570v3, Feb. 2018.
- [7] J. Otterbach et. al., “Unsupervised Machine Learning on a Hybrid Quantum Computer,” arXiv:1712.05771v1, Dec. 2017.
- [8] N. Dider et. al., “Analytical modeling of parametrically-modulated transmon qubits,” arXiv:1706.06566v2, Jan. 2018.
- [9] Rigetti Computing, “pyQuil Documentation Release 2.4.0,” 15 Feb. 2019. [Online]. Available: <https://media.readthedocs.org/pdf/pyquil/stable/pyquil.pdf>.
- [10] R. Smith, M. Curtis, W. Zeng, “A Practical Quantum Instruction Set Architecture,” arXiv:1608.03355v2, Feb. 2017.
- [11] Rigetti Computing, “QPU Specifications.” [Online]. Available: <https://rigetti.com/qpu>.
- [12] D. Miller and M. Thornton, “QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits, in *IEEE International Symposium on Multiple-Valued Logic*, pp. 30-30, 2006.
- [13] 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.
- [14] “Reversible Logic Synthesis and Quantum Computing Benchmarks,” 2017. [Online]. Available: <http://quantumfpl.stationq.com/>.