Compiling Permutations for Superconducting QPUs

Mathias Soeken Fereshte Mozafari Bruno Schmitt Giovanni De Micheli
Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

***Abstract*—In this paper we consider the compilation of quantum state permutations into quantum gates for physical quantum computers. A sequence of generic single-target gates, which realize the input permutation, are extracted using a decomposition based reversible logic synthesis algorithm. We present a compilation algorithm that translates single-target gates into a quantum circuit composed of the elementary quantum gate sets that are supported by IBM's 5-qubit and 16-qubit, and Rigetti's 8-qubit and 19-qubit superconducting transmon QPUs. Compared to generic state-of-the-art compilation techniques, our technique improves gate volume and gate depth by up to 59% and 53%, respectively.**

I. Introduction

Quantum computers are physical machines that consist of an array of qubits (quantum memory), which state can be altered by means of a set of quantum operations, typically referred to as quantum gates. The state of an array of n qubits is described in terms of 2^n complex-valued amplitudes. Each amplitude corresponds to the probability of the quantum state being in one of the 2^n possible Boolean states after measuring all qubits. A quantum state applied to a subset of qubits in the array of qubits may change all amplitudes in the quantum state. This is one of the main reasons for the power of quantum computers, since the application of an operation to a linear number of resources (qubits) can simultaneously change an exponential number of values (amplitudes).

In the last few years several physical implementations of quantum computers have been demonstrated by, e.g., IBM [1], Rigetti [2], Google [3], Intel [4], IonQ [5], and Alibaba [6]. The numbers of qubits appears to be the prime distinguishing property of quantum computers, however, several other factors significantly affect the quality of a quantum computer. These include qubit coherence times, coupling restrictions on which qubits can interact with each other in quantum operations, as well as the supported quantum gate set. Since the typical use of a quantum computer is as a compute kernel inside a classical computation, they are also referred to as quantum processing units (QPUs).

The interaction of quantum operations with qubits in a quantum computer is described in terms of quantum algorithms, also called quantum circuits. High-level quantum algorithms are technology-independent, allow arbitrary quantum operations, and do not take architectural constraints into account. Quantum compilation is the task of translating a high-level quantum algorithm into a low-level quantum circuit, which is technology-dependent, i.e., it is described in terms of supported quantum operations and respects all architectural constraints. It is not uncommon that several phases of quantum compilation take place when translating a high-level quantum algorithm into a low-level quantum circuit [7].

In this paper, we propose an automatic technology-dependent compilation technique to translate quantum operations that permute the amplitudes in quantum states. Many quantum algorithms make use of such permutations, in particular as a way to implement combinatorial operations [8]. Our compilation algorithm targets quantum architectures which gate set supports rotation gates with arbitrary angles, such as the 8-qubit and 19-qubit superconducting transmon computers from Rigetti.

The proposed approach utilizes Young-subgroup based reversible logic synthesis [9], [10], [11], which for a given permutation for a qubit state over n qubits, finds a sequence of $2n-1$ so-called single-target gates, which describe quantum operations to alter the quantum state w.r.t. a Boolean function. We describe a general algorithm to translate a single-target gate into a quantum circuit composed of Clifford+R_z gates (details on these operations are provided in the next section). Finally, we employ an explicit rewiring technique in order to reduce the number of quantum gates.

In an experimental evaluation, we show that our proposed approach leads to quantum circuits with lower quantum gates and lower depth compared to state-of-the-art generic compilation techniques. For Rigetti QPUs we can reduce gate count and gate depth up to 59% and 53%, respectively, and for IBM QPUs we can reduce gate count and gate depth up to 56% and 53%, respectively.

II. Preliminaries

A. Boolean functions and spectral techniques

A Boolean function is a mapping $f : \mathbb{B}^n \to \mathbb{B}$ where $\mathbb{B} = \{0, 1\}$. One way to represent a Boolean function $f(x_1, \ldots, x_n)$ is in terms of its *truth table*, a column vector $f = (f_0, \ldots, f_{2^n-1})^T$ where each entry $f_{(b_n \ldots b_1)_2} = f(b_1, \ldots, b_n)$ corresponds to an assignment of inputs to f. Note that we use f both to refer to the function and to its truth table.

Example 1: The majority-of-three function $f = \langle x_1 x_2 x_3 \rangle = x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$ has the truth table $f = (0, 0, 0, 1, 0, 1, 1, 1)^T$.

The $\{-1, 1\}$ encoding of a truth table is a column vector $\hat{f} = (\hat{f}_{2^n-1}, \ldots, \hat{f}_0)^T$ where $\hat{f}_i = 1 - 2f_i$ for $0 \leq i < 2^n$. In other words, $\hat{f}_i = -1$, if $f_i = 1$, and $\hat{f}_i = 1$, if $f_i = 0$.

Example 2: For $f = \langle x_1 x_2 x_3 \rangle$ from the previous example, we have $\hat{f} = (1, 1, 1, -1, 1, -1, -1, -1)^T$.

The recursive Hadamard matrix is

$$H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}, \quad \text{and } H_0 = 1. \tag{1}$$

Then for an n-variable Boolean function f, the vector $s = H_n \hat{f}$ is called the Rademacher-Walsh *spectrum* of f. We will refer to s as spectrum in the remainder of the paper.

Example 3: For $f = \langle x_1 x_2 x_3 \rangle$, the spectrum is $s = (0, 4, 4, 0, 4, 0, 0, -4)^T$.

B. Qubits and quantum gates

A quantum computer consists of an array of qubits, which in contrast to classical bits, can be in a superposition state and can be entangled [12]. Formally, a qubit is in a quantum state that is a column vector $|\varphi\rangle = \binom{\alpha}{\beta}$ of two complex numbers α and β, called amplitudes, such that $|\alpha|^2 + |\beta|^2 = 1$. The squared amplitudes $|\alpha|^2$ and $|\beta|^2$ indicate the probability that the quantum state will collapse to the classical state $|0\rangle = \binom{1}{0}$ or $|1\rangle = \binom{0}{1}$ after the qubit is measured. A quantum state can be transformed into another quantum state by applying quantum gates, which are represented by 2×2 unitary matrices.

Example 4: The Hadamard gate $H = \frac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ transforms the classical quantum state $|0\rangle$ into the state $\frac{1}{\sqrt{2}}\binom{1}{1}$, which is in the perfect superposition between 0 and 1.

Since single qubit states correspond to points on the Bloch sphere [12], quantum operations on a single qubit correspond to rotations. Besides the H operation, we will consider x-rotations $R_x(\theta) = \left(\begin{smallmatrix} \cos\frac{\theta}{2} & -\mathrm{i}\sin\frac{\theta}{2} \\ -\mathrm{i}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{smallmatrix}\right)$ and z-rotations $R_z(\theta) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\theta} \end{smallmatrix}\right)$ with continuous angles $\theta \in \mathbb{R}$. Commonly used rotations are $I = R_x(0) = R_z(0)$, $X = R_x(\pi)$, $Z = R_z(\pi)$, $V = R_x(\frac{\pi}{2})$, $S = R_z(\frac{\pi}{2})$, and $T = R_z(\frac{\pi}{4})$, as well as their complex conjugates $V^\dagger$, $S^\dagger$, and $T^\dagger$ obtained by negating the angle.

Quantum states over n qubits are represented by a column vector of 2^n complex values α_x with $x \in \mathbb{B}^n$ such that $\sum_{x \in \mathbb{B}^n} |\alpha_x|^2 = 1$. Each squared amplitude $|\alpha_x|^2$ indicate the probability that after measurement the n qubits are in classical states x. Quantum states can be combined by applying the Kronecker product to produce larger ones, e.g., $\binom{1}{0} \otimes \frac{1}{\sqrt{2}}\binom{1}{1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$, which represents a 2-qubit state that is in the perfect superposition between the classical states 00 and 01. On the contrary, larger states cannot always be represented in terms of smaller ones. For example, there are no two independent qubit states $|\varphi_1\rangle$ and $|\varphi_2\rangle$ such that $|\varphi_1\rangle \otimes |\varphi_2\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, the state that is in the perfect superposition between the classical states 00 and 11. This phenomena is called entanglement.

Quantum gates that act on n qubits are represented in terms of $2^n \times 2^n$ unitary matrices. We are considering 3 two-qubit gates in this paper, controlled-X (CNOT), controlled-Z (CZ), and SWAP, which are defined as

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \ \mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$
$$\text{and } \mathrm{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (2)$$

C. Quantum gate libraries and quantum architectures

Rigetti quantum computers: The 8 qubit QPU (called Agave) and 19 qubit QPU (called Acorn) from Rigetti natively supports four X-rotations $\{R_x(\frac{k\pi}{2}) \mid k \in \mathbb{Z}\} = \{I, V, X, V^\dagger\}$, arbitrary

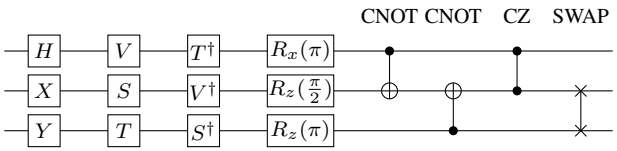
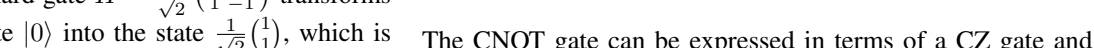
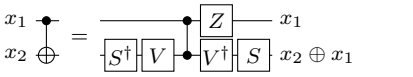


Fig. 1. Quantum gates in a quantum circuit. Single-qubit gates are drawn as box with their name inside, two-qubit gates have a special symbolic notation.

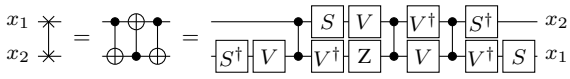
Z-rotations $R_z(\theta)$ for any $\theta \in \mathbb{R}$ and CZ gates. Other gates that were mentioned in the previous section can be represented in terms of this library. The Hadamard gate H can be expressed in terms of three rotations:



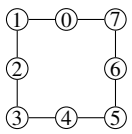
The CNOT gate can be expressed in terms of a CZ gate and rotations, using 6 gates and a depth of 5:



Finally, the SWAP gate can be expressed in terms of three CZ gates and rotations, using 14 gates and a depth of 10:

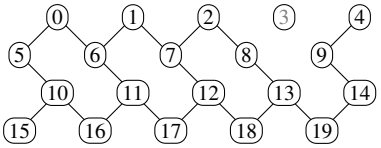


The topology of the 8 qubit quantum computer is the undirected 8-cycle



, (3)

while the topology of the 19 qubit computer is according to the following undirected graph:

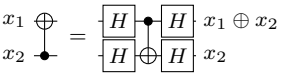


(4)

IBM quantum computers: The IBM quantum computers natively support the U gate, $U(\theta, \phi, \lambda) = R_z(\phi) R_y(\theta) R_z(\lambda)$, which is parameterized over 3 continuous variables, and the CNOT gate. As the algorithm described in this paper maps permutations to an intermediate representation containing of CNOT, Hadamard, and arbitrary Z rotations, it is helpful to point out the following identities:

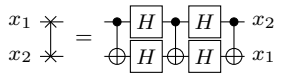
$$H = U(\tfrac{\pi}{2}, 0, \pi) \quad \text{and} \quad R_z(\theta) = U(\theta, 0, 0) \quad (5)$$

The coupling constraints in an IBM architecture also imply a direction of a CNOT gate, i.e., for two adjacent qubits the target of a CNOT gate can only be applied to one of the qubits. The following circuit identity can be used to turn the direction of a CNOT gate:



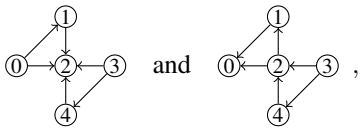
(6)

The same identity can be used to implement a SWAP operation:



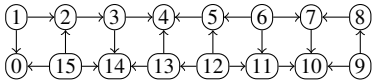
(7)

In our experiments we evaluate our algorithm on the three available cloud-based IBM quantum computers ibmqx2 (also called IBM Q 5 Yorktown), ibmqx4 (also called IBM Q 5 Tenerife), each of which have 5 qubits, and ibmqx5 (also called IBM Q 16 Rueschlikon), which has 16 qubits. The coupling constraints for ibmqx2 and ibmqx4 are defined be the two directed graphs



(8)

respectively. For ibmqx5, the coupling constraints are as follows:



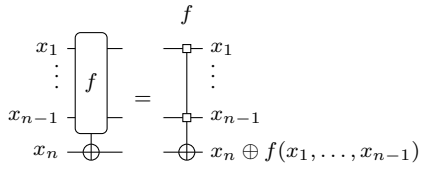
(9)

D. Single-target gates

A single-target gate with control function $f(x_1, \ldots, x_{n-1})$ and target qubit x_n is an abstract quantum operation acting on n qubits that maps

$$U_f : |x_1 \ldots x_{n-1}\rangle|x_n\rangle \mapsto |x_1 \ldots x_{n-1}\rangle|x_n \oplus f(x_1, \ldots, x_{n-1})\rangle. \tag{10}$$

In other words, it inverts the value of the target qubits x_n, if and only if the control function evaluates to true for the values of the control qubits $x_1, \ldots, x_{n-1}$. Well-known instances of single-target gates are the X gate for which $n = 1$ and $f = 1$, the CNOT gate for which $n = 2$ and $f = x_1$, or the Toffoli gate for which $n = 3$ and $f = x_1 x_2$. Several pictorial representations for single-target are used in the literature. In the remainder of the paper we use one of the following two:



(11)

Single-target gates describe complex operations and cannot generally be implemented natively on quantum computer. However, they provide a convenient intermediate representations when mapping complex functionality, such as permutations, into quantum gates.

III. Proposed Compilation Algorithm

Fig. 2 provides a birds-eye overview of our proposed algorithm. The algorithm takes as input a permutation over 2^n elements, the gate library, and coupling constraints of the target quantum computer. It returns a quantum circuit composed of gates from the gate library, which respects the coupling constraints. The algorithm essentially contains the following 3 steps:

1) Map the input permutation into a reversible circuit compose of $2n - 1$ single-target gates.
2) Map each single-target gate into a circuit over the gate set $\{\text{CNOT}, R_z(\theta), H\}$.
3) Map the resulting quantum circuit into a circuit composed of gate library gates, which respects the coupling constraints.

For the first step, we employ the decomposition-based reversible synthesis algorithm using Young-subgroups presented in [10]. Fig. 2 illustrates this step for the permutation $[0, 2, 3, 5, 7, 1, 4, 6]$, which can be realized using a reversible circuit consisting of five single-target gates with control functions $f_1, \ldots, f_5$.

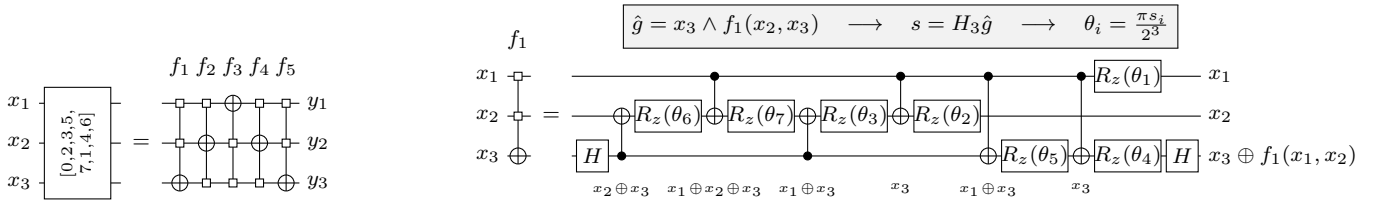
The second step encompasses the main contribution of this paper. Each single-target gate is decomposed into a circuit structure composed of H gates, CNOT gates, and R_z gates. The step is illustrated for the first single-target gate with control function f_1 in Fig. 2(b). Hadamard gates are inserted at the beginning and at the end of the circuit on the target qubit of the single-target gate. CNOT gates are used to create all linear combinations of the inputs. In Fig. 2(b) these linear combinations are written under each CNOT gate. To each of these linear combinations one R_z gate is applied, whose rotation angle corresponds to the spectral coefficients of the function $x_3 \wedge f_1(x_1, x_2)$. For example, the first CNOT gate creates the linear combination $x_2 \oplus x_3$, which corresponds to the spectral coefficient s_6. The rotation angle is $\theta_6 = \frac{\pi s_6}{2^3}$. Note that $R_z(0)$ gates may be omitted which in turn can cause further reduction of CNOT gates. We make use of the GRAYSYNTH algorithm [13] to find a network with possibly few CNOT gates for generating required linear combinations.

By performing the second step for each single-target gate, one obtains a quantum circuit that can be passed as input to a quantum compiler. However, the compiler of the quantum computer still needs to apply changes to accommodate for the supported gate library and coupling constraints. For example, in case of the Rigetti 8-qubit computer, the H and CNOT gates need to be translated into R_x, R_z and CZ gates; also, there are no three qubits that permit pairwise interaction and therefore some SWAP gates need to be inserted. We extended the existing GRAYSYNTH algorithm by heuristics to further minimize the number of CNOTs, thereby reducing the possibility of violating coupling constraints. For example, special strategies can be used when the spectrum has no zero coefficients and therefore CNOTs for all linear combinations need to be generated.

For the third step, we make use of the quantum gate compilers shipped with the software development kits from Rigetti and IBM.

IV. Compiling Single-target Gates

In this section, we describe the second step of the algorithm that was presented in the previous section. Before delving into the details on how to compile single-target gates, we briefly review a well-known identity for Toffoli gates, i.e., a single target gate acting on three qubits with control function $f = x_1 x_2$. For compiling Toffoli gates into quantum gates, it

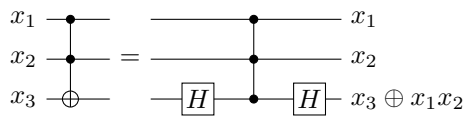


(a) Step 1: map permutation into sequence of single-target gates using Young-subgroup based synthesis [10].

(b) Step 2: each single-target gate is translated into a regular circuit structure composed of H gates, CNOT gates, and R_z gates. The angles for the R_z gates can be obtained from the control function of the single-target gate.

Fig. 2. Birds-eye overview of the proposed compilation algorithm.

is often first mapped into a doubly-controlled Z gate using Hadamard gates:



(12)

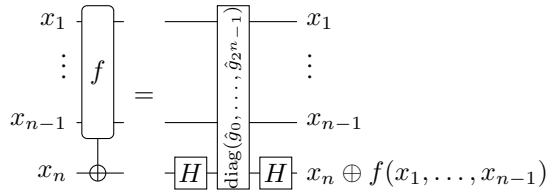
The unitary operation of the doubly-controlled Z gate is $\mathrm{diag}(1,1,1,1,1,1,1,-1)$.

The following theorem generalizes this decomposition for single-target gates of arbitrary size.

Theorem 1: Let U_f be the unitary matrix realized by a single-target gate acting on x_n with control function $f(x_1, \ldots, x_{n-1})$. Then

$$U_f = (I_{2^{n-1}} \otimes H) \cdot \mathrm{diag}(\hat{g}_0, \ldots, \hat{g}_{2^n-1}) \cdot (I_{2^{n-1}} \otimes H), \quad (13)$$

where $\hat{g}_i$ are the truth table entries of the function $g = x_n \wedge f$ in $\{-1, 1\}$ coding. The following circuit illustrates the decomposition:



(14)

Proof: We evaluate the effect of applying the right-hand side of (13) to the quantum state $|x\rangle|x_n\rangle$, where $|x\rangle = |x_1 \ldots x_{n-1}\rangle$. Applying $(I_{2^{n-1}} \otimes H)$ to $|x\rangle|x_n\rangle$ yields

$$|\varphi_1\rangle = \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^{x_n}|1\rangle).$$

Now note that $\mathrm{diag}(g_0, \ldots, g_{2^n-1})$ maps a state $|x\rangle|x_n\rangle$ to $(-1)^{x_n \wedge f(x)}|x\rangle|x_n\rangle$. When making a case distinction on x_n, one can see that the operation maps $|x\rangle|0\rangle$ to $|x\rangle|0\rangle$, and $|x\rangle|1\rangle$ to $(-1)^{f(x)}|x\rangle|1\rangle$. Therefore, applying the diagonal matrix to $|\varphi_1\rangle$ yields

$$\begin{aligned}|\varphi_2\rangle &= \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^{f(x)}(-1)^{x_n}|1\rangle) \\ &= \frac{1}{\sqrt{2}}|x\rangle(|0\rangle + (-1)^{f(x)\oplus x_n}|1\rangle).\end{aligned}$$

Finally, $(I_{2^{n-1}} \otimes H)|\varphi_2\rangle = |x\rangle|x_n \oplus f(x)\rangle = U_f|x\rangle|x_n\rangle$. ■

Welch et al. have shown in [14] that the matrix $\mathrm{diag}(\hat{g}_0, \ldots, \hat{g}_{2^n-1})$ is equivalent to the unitary operation that maps $|x\rangle$ to

$$e^{\mathrm{i}\pi s(x)/2^n}|x\rangle, \quad (15)$$

where $s(x) = \sum_{y \in \mathbb{B}^n} s_y |y\rangle\langle x|$.

Schuch and Siewert have shown in [15] that a unitary mapping as in (15) can be implemented by a quantum circuit on n qubits that uses only CNOT and R_z gates. The CNOT gates are used to generate the linear combinations $|y\rangle\langle x|$ on some qubit to which then the phase gate $R_z\left(\frac{\pi s_y}{2^n}\right)$ is applied. Note that only those linear combinations need to be generated for which $s_y \neq 0$. In [13], Amy et al. presented an algorithm called GRAYSYNTH that finds a circuit which minimizes the number of CNOT gates.

V. REWIRING OPTIMIZATIONS

The algorithm to compile single-target gates as described in the previous section does not take into account the coupling constraints of the quantum computer. In this section, we describe three modifications to the algorithm that take the coupling constraints into account.

First, in order to find a good CNOT network to create linear combinations for the non-zero spectral coefficients, GRAYSYNTH uses a heuristic to minimize the number of CNOT gates. Often the algorithm has multiple choices and we use the coupling constraints as a tie breaker in such cases.

Second, in the case of spectra with no zero coefficients, all linear combinations of qubits are required. In this case, we employ dedicated algorithms to generate CNOT sequences for all linear combinations instead of using GRAYSYNTH. Since all spectral coefficients are not zero, the sequence of CNOT gates depends only on the number of variables in the function. As a consequence, we can precompute circuit structures that minimize the number of CNOTs as well as violations of coupling constraints of the quantum computer.

Third, we allow rewirings after each single-target gate. Formally, instead of implementing U_f as in (10), we find a circuit for $U_\pi U_f$, where $U_\pi : |x_1 \ldots x_n\rangle \mapsto |x_{\pi(1)} \ldots x_{\pi(n)}\rangle$ for some permutation $\pi \in S_n$ (i.e., over n elements). The unitary transformation U_π can be realized without any gates,

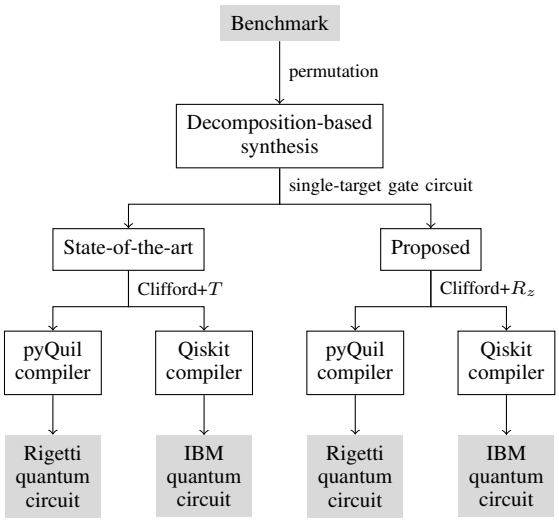
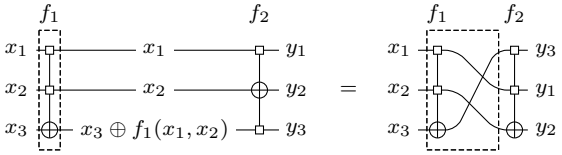


Fig. 3. Compilation flows for experimental results

simply by rewiring the qubits. The following example illustrates this:



The highlighted gate on the left-hand side implements U_{f_1} while the highlighted gate on the right-hand side implements $U_{[3,1,2]}U_{f_1}$. The implementation of the latter may require fewer gates.

VI. Experimental Results

We implemented the proposed algorithm and the state-of-the-art algorithm in C++ into RevKit [16][1] using the quantum compilation framework *tweedledum*.[2] The scripts to regenerate the experimental results are available on Github.[3]

Benchmarks: We use the benchmark families `TOF(`n`)`, `PRIME(`n`)`, and `HWB(`n`)` to evaluate our proposed approach, where n indicates that the benchmark describes a permutation over 2^n elements. The benchmark `TOF(`n`)` describes a multiple-controlled Toffoli gate with $n-1$ control lines acting on the least-significant qubit; it represents the single transposition $(2^{n-2}, 2^{n-1})$. Such gates play, e.g., an important role in the Grover search quantum algorithm [17]. The benchmark `PRIME(`n`)` represents the permutation that maps 0 to 0, and then maps the successive numbers first to all primes and then to all non-primes smaller than 2^n in increasing order. For example, `PRIME(3)` $= [0, 2, 3, 5, 7, 1, 4, 6]$, the permutation used in the example in Fig. 2. Finally, the benchmark `HWB(`n`)` maps x to $x \ll_r \nu x$, i.e., x is left-shift-rotated by the number of 1s in x. For example, `HWB(3)` $= [0, 2, 4, 5, 1, 6, 3, 7]$.

[1]see github.com/msoeken/cirkit

[2]see github.com/boschmitt/tweedledum

[3]see github.com/lsils/benchmarks-date2019-permutations

Methodology: Fig. 3 shows the flow for obtaining the results. Input is a benchmark generated from the benchmark families described in the previous section. This permutation is decomposed into a sequence of single-target gates using the algorithm described in [10]. Then each single-target gate is mapped into a Clifford+T circuit using a state-of-the-art compilation flow or into a Clifford+R_z circuit using our proposed flow. In the state-of-the-art synthesis flow, we first use ESOP-based (exclusive sum-of-products) synthesis to map each single-target gate into a sequence of multiple-controlled mixed-polarity Toffoli gates, which are single-target gates in which the control function is a product term (conjunction of literals). Multiple-controlled Toffoli gates with more than four controls are decomposed into Toffoli gates with at most four controls using the decomposition described in [18]; this may introduce one additional helper qubit called *ancilla*. Finally, each remaining multiple-controlled Toffoli gate is decomposed into a Clifford+T circuit as described in [19].

The circuits generated by the state-of-the-art and proposed compilation approach are then compiled into architecture aware circuits for both Riggeti and IBM quantum computers using the compilers provided in their software development kits. For Rigetti we use architecture configurations for Agave 8Q and Acorn 19Q, for IBM we use architecture configurations for both 5-qubit and the 16-qubit quantum computers.

Tables I and II show the experimental results after compilation for the Rigetti and IBM quantum computers, respectively. The table shows the input permutation and the number of variables. For each quantum computer, it shows the number of gates and the gate depth after compilation using the state-of-the-art and the proposed approach. Note that gate count and volume for the Rigetti computers (R gates and R depth) and the IBM computers (I gates and I depth) are not directly comparable due to differences in the underlying gate sets and device technologies. In the columns for the proposed approach also the improvement in percentage is listed. The runtime in all cases is a few seconds and negligible. As can be seen, our proposed approach is particularly powerful when the number of variables is larger than 3, because only then the proposed algorithm can exploit the smaller rotation angles in the R_z gates. Also, the proposed algorithm does not need any ancilla. As a result, benchmarks such as `TOF(5)`, `PRIME(5)`, and `HWB(5)` can be compiled for a 5-qubit quantum computer using the proposed approach, while the state-of-the-art approach cannot generate compatible circuits (see cells marked N/A). Further, for the experiments on the IBM quantum computer, it can be seen that more improvement is possible for the smaller quantum computers, indicating that our proposed algorithm better addresses the coupling constraints.

VII. Conclusions

We presented a compilation algorithm to realize permutations in terms of quantum circuits for Rigetti's and IBM's superconducting computers. Contrary to the state-of-the-art approaches, our approach better utilizes the gate set offered by the respective quantum computers and optimizes with respect to the quantum computers' architectures. In future work, we plan to fully integrate the mapping into our compilation approach.

TABLE I
EXPERIMENTAL RESULTS AFTER COMPILATION FOR RIGETTI COMPUTERS

Permutation		Rigetti Agave 8Q						Rigetti Acorn 19Q					
		SOTA		Proposed				SOTA		Proposed			
	vars	R gates	R depth	R gates	impr.	R depth	impr.	R gates	R depth	R gates	imp.	R depth	impr.
TOF(3)	3	47	32	48	-2.13%	35	-9.38%	47	32	47	0.00%	37	-15.62%
TOF(4)	4	222	99	187	15.77%	107	-8.08%	194	93	142	26.80%	90	3.23%
TOF(5)	5	374	176	484	-29.41%	256	-45.45%	237	117	308	-29.96%	160	-36.75%
TOF(6)	6	883	471	957	-8.38%	491	-4.25%	1311	692	645	50.80%	325	53.03%
PRIME(3)	3	119	79	136	-14.29%	106	-34.18%	120	79	136	-13.33%	104	-31.65%
PRIME(4)	4	1212	614	664	45.21%	396	35.50%	953	573	674	29.28%	407	28.97%
PRIME(5)	5	5163	2338	2609	49.47%	1420	39.26%	4339	2178	1867	56.97%	1083	50.28%
HWB(4)	4	1217	691	995	18.24%	545	21.13%	1290	688	957	25.81%	525	23.69%
HWB(5)	5	6825	3213	3503	48.67%	1782	44.54%	6228	3046	2545	59.14%	1528	49.84%

TABLE II
EXPERIMENTAL RESULTS AFTER COMPILATION FOR IBM COMPUTERS

Permutation		IBM Q 5 Yorktown						IBM Q 5 Tenerife						IBM Q 16 Rueschlikon					
		SOTA		Proposed				SOTA		Proposed				SOTA		Proposed			
	vars	I gates	I depth	I gates	impr.	I depth	impr.	I gates	I depth	I gates	impr.	I depth	impr.	I gates	I depth	I gates	impr.	I depth	impr.
TOF(3)	3	17	13	15	11.76%	12	7.69%	16	13	18	-12.50%	13	0.00%	48	32	41	14.58%	27	15.62%
TOF(4)	4	46	34	61	-32.61%	41	-20.59%	113	69	74	34.51%	49	28.99%	122	75	124	-1.64%	72	4.00%
TOF(5)	5	*N/A*		132		89		*N/A*		136		100		222	124	262	-18.02%	152	-22.58%
TOF(6)	6	*N/A*		*N/A*				*N/A*		*N/A*				590	304	562	4.75%	341	-12.17%
PRIME(3)	3	47	37	47		40	-8.11%	45	36	48	-6.67%	40	-11.11%	84	64	97	-15.48%	70	-9.38%
PRIME(4)	4	363	255	271	25.34%	174	31.76%	465	321	347	25.38%	229	28.66%	630	383	454	27.94%	275	28.20%
PRIME(5)	5	*N/A*		852		574		*N/A*		786		558		2827	1566	1635	42.16%	979	37.48%
PRIME(6)	6	*N/A*		*N/A*				*N/A*		*N/A*				11448	5985	4924	56.99%	2802	53.18%
HWB(4)	4	464	299	340	26.72%	223	25.42%	535	327	407	23.93%	257	21.41%	818	507	583	28.73%	336	33.73%
HWB(5)	5	*N/A*		854		582		*N/A*		1046		670		4254	2364	2096	50.73%	1211	48.77%

This allows us to drop the dependency to the quantum computer specific compiler from the compilation flow in Fig. 3. Also, we like to extend our approach to address multi-target gates.

Acknowledgments: We wish to thank Eric Peterson and Ryan Karle from Rigetti and Andrew Cross and Ali Javadi-Abhari from IBM for providing support in using their toolchains. This research was supported by the Swiss National Science Foundation (200021-169084 MAJesty) and by the European Research Council in the project H2020-ERC-2014-ADG 669354 CyberCare.

REFERENCES

[1] IBM, "IBM builds its most powerful universal quantum computing processors," 2017, press release by IBM, posted online May 17, 2017.

[2] J. S. Otterbach *et al.*, "Unsupervised machine learning on a hybrid quantum computer," *arXiv preprint arXiv:1712.05771*, 2017.

[3] Google, "A preview of Bristlecone, Google's new quantum processor," 2018, article on Google AI Blog, posted online March 5, 2018.

[4] Intel, "The future of quantum computing is counted in qubits," 2018, press release by Intel, posted online May 2, 2018.

[5] C. Figgatt, D. Maslov, K. A. Landsman, N. M. Linke, S. Debnath, and C. Monroe, "Complete 3-qubit Grover search on a programmable quantum computer," *Nature Communications*, vol. 8, no. 1918, pp. 1–9, 2017.

[6] Alibaba, "Alibaba Cloud and CAS launch one of the world's most powerful public quantum computing services," 2018, press release by Alibaba Cloud, posted online March 1, 2018.

[7] F. T. Chong, D. Franklin, and M. Martonosi, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, no. 7671, pp. 180–187, 2017.

[8] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang, "Experimental realization of an order-finding algorithm with an NMR quantum computer," *Physical Review Letters*, vol. 85, no. 25, pp. 5452–5455, 2000.

[9] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits - a survey," *ACM Computing Surveys*, vol. 45, no. 2, pp. 21:1–21:34, 2013.

[10] A. De Vos and Y. Van Rentergem, "Young subgroups for reversible computers," *Advances in Mathematics of Communications*, vol. 2, no. 2, pp. 183–200, 2008.

[11] D. Deutsch, "Quantum computational networks," *Proc. R. Soc. Lond.*, vol. A 425, pp. 73–90, 1989.

[12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[13] M. Amy, P. Azimzadeh, and M. Mosca, "On the CNOT-complexity of CNOT-phase circuits," *arXiv preprint arXiv:1712.01859*, 2017.

[14] J. Welch, D. Greenbaum, S. Mostame, and A. Aspuru-Guzik, "Efficient quantum circuits for diagonal unitaries without ancillas," *New Journal of Physics*, vol. 16, no. 033040, pp. 1–15, 2014.

[15] N. Schuch and J. Siewert, "Programmable networks for quantum algorithms," *Physical Review Letters*, vol. 91, no. 027902, 2003.

[16] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: A toolkit for reversible circuit design," *Multiple-Valued Logic and Soft Computing*, vol. 18, no. 1, pp. 55–65, 2012.

[17] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Symposium on Theory and Computing*, 1996, pp. 212–219.

[18] 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," *Physical Review A*, vol. 52, no. 5, p. 3457, 1995.

[19] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Physical Review A*, vol. 93, p. 022311, 2016.