Context-Sensitive and Duration-Aware Qubit Mapping for Various NISQ Devices

YU ZHANG ✉, HAOWEI DENG, and QUANXI LI, University of Science and Technology of China

Quantum computing (QC) technologies have reached a second renaissance in the last decade. Some fully programmable QC devices have been built based on superconducting or ion trap technologies. Although different quantum technologies have their own parameter indicators, QC devices in the NISQ era share common features and challenges such as limited qubits and connectivity, short coherence time and high gate error rates. Quantum programs written by programmers could hardly run on real hardware directly since two-qubit gates are usually allowed on few pairs of qubits. Therefore, quantum computing compilers must resolve the mapping problem and transform original programs to fit the hardware limitation.

To address the issues mentioned above, we summarize different quantum technologies and abstractly define Quantum Abstract Machine (QAM); then propose a **CO**ntext-sensitive and **D**uration-**A**ware **R**emapping algorithm (Codar) based on the QAM. By introducing lock for each qubit, Codar is aware of gate duration difference and program context, which bring it abilities to extract more program's parallelism and reduce program execution time. Compared to the best-known algorithm, Codar halves the total execution time of several quantum algorithms and cut down 17.5% ~ 19.4% total execution time on average in different architectures.

1 INTRODUCTION

Quantum Computing (QC) has attracted huge attention in recent a decade due to its ability to exponentially accelerate several important algorithms [12, 15, 27, 33]. Both QC algorithm designers and programmers work at a very high level, and need to know little about (future) Noisy Intermediate-Scale Quantum (NISQ) devices that (will) execute quantum programs. There exists a gap, however, between NISQ devices and the hardware requirements (*e.g.,* size and reliability) of QC algorithms. To bridge this gap, QC requires abstraction layers and toolchains to translate and optimize applications [9]. QC compilers typically translate high-level QC code into (optimized) circuit-level assembly code in multiple stages. In order to use NISQ hardware, quantum circuit programs have to be compiled to the target device, which includes mapping logical qubits to physical ones of the device. The mapping step, which we focus on in this abstract, faces a tough challenge because further physical constraints have to be considered. In fact, 2-qubit gates can only be applied to certain physical qubit pairs. Therefore, additional SWAP operations have to be inserted in order to "move" the logical qubits to positions where they can interact with each other. This qubit mapping problem has been proved to be a NP-Complete problem [35].

Previous solutions to this problem can be classified into two types. One type is to formulate the problem into an equivalent mathematical problem and apply a solver [6–8, 22, 25, 28, 30, 31, 38, 39, 41, 43], and another type is to use heuristic search to obtain approximate results [5, 17–20, 29, 34, 42, 45]. The former suffers from very long runtime and can only be applied to small size

This work was partly supported by the grants of the National Natural Science Foundation of China (No. 61772487), Anhui Initiative in Quantum Information Technologies (No. AHY150100) and Anhui Provincial Major Teaching and Research Project (No. 2017jyxm0005).

Corresponding author: Yu Zhang, email: yuzhang@ustc.edu.cn.

This is the extended abstract of the talk accepted by the First International Workshop on Programming Languages for Quantum Computing (PLanQC 2020), Jan 19, 2020, New Orleans, Louisiana, USA .

cases. The latter is better in runtime especially when the circuit is in a large scale. All of them assume different gates have the same execution duration.

Table 1. Parameter information of several quantum computing devices.

		Ion Trap		Superconducting			Neutral Atom[32]
		Ion Q5[21]	Ion Q11[44]	IBM Q5[21]	IBM Q16[26]	IBM Q20[19]	
Available 1-qubit gate		R_α^θ		X, Y, Z, H, S, T			R_α^θ
Available 2-qubit gate		XX		CNOT			CNOT[36]
Fidelity	1-qubit gate	99.1(5)%	99.5%	99.7%	~99.8%	~99.56%	99.995% [32]
	2-qubit gate	97(1)%	97.5%[95.1%,98.9%]	96.5%	~96%	~97%	82%[23]
	1-qubit readout	$\|0\rangle$:99.7(1)%, $\|1\rangle$:99.1(1)%	99.3%	~ 96%	~93%	~91.2%	98.6% [13]
	average readout	95.7(1)%	–	~ 80%	–	–	>97% [24]
Time	1-qubit gate	20μ s		130 ns	80 ns	–	1μ ~20μ s
	2-qubit gate	250μ s	–	250-450 ns	170-391 ns	–	~10μ s
Depolarization (T_1)		~ ∞	–	~ 60μ s	~ 70μ s	87.29μ s	>10s
Spin dephasing (T_2)		~ 0.5s	–	~ 60μ s	~ 70μ s	54.43μ s	~ 1s

On NISQ hardware, however, different gates have different durations (see Table 1). Ignoring the gate duration difference may cause these algorithms to find the shortest depth but not the shortest execution time. The real execution time of the circuit is associated with the weighted depth, in which different gates have different duration weights. Considering gate duration difference will help the compiler make better use of the parallelism of quantum circuit and generate the circuit with shorter execution time. In this abstract, we focus on solving the *qubit mapping problem* by heuristic search with the consideration of gate duration difference and program context to explore more program's parallelism. To address the challenges of qubit mapping problem and adapt to different quantum technologies, we first give several examples to explain our motivation, then propose a quantum abstract machine (QAM) for studying the qubit mapping problem. The QAM is modelled as a 2D coupling graph with limited connectivity and configurable durations of different kinds of quantum gates. Based on the QAM, we further propose two mechanisms that enable **CO**ntext-sensitive and **D**uration-**A**ware **R**emapping algorithm (CODAR) to solve the qubit mapping problem with the awareness of gate duration difference and program context. Experimental results show that compared to the best known remapping algorithm, CODAR can cut down 17.5% ~ 19.4% weighted depth at average.

2 PROBLEM ANALYSIS

2.1 Recent Work on Qubit Mapping

There are a lot of research on the qubit mapping problem. Here we focus on analyzing some valuable solutions in recent two years [4, 19, 26, 35, 37, 41, 45]. All of them are proposed for some IBM QX architectures, and none of them consider the gate duration difference.

Solutions only considering qubit coupling. [35, 41] provide solutions for 5-qubit IBM QX architectures with directed coupling. Siraichi *et al.* [35] propose an optimal algorithm based on dynamic programming, which only fits for small circuits; then they propose a heuristic one which is fast but oversimplified with results worse than IBM's solution. Wille *et al.* [41] present a solution with a minimal number of additional `SWAP` and `H` operations, in which qubit mapping problem is formulated as a symbolic optimization problem with high complexity. They utilize powerful reasoning engines to solve the computationally task.

[19, 45] use heuristic search to provide good solutions in acceptable time for large scale circuits. Zulehner *et al.* divide the two-qubit gates into independent layers, then use A^* search plus heuristic cost function to determine compliant mappings for each layer [45]. Li *et al.* propose a SWAP-based

bidirectional heuristic search algorithm, named SABRE [19], which can produce comparable results with exponential speedup against previous solutions such as [45].

Solutions further considering error rates. [4, 26, 37] provide another type of perspective for solving the qubit mapping problem. They consider the variation in the error rates of different qubits and connections to generate directly executable circuits that improve reliability rather than minimize circuit depth and number of gates. Based on the error rate data from real IBM Q16 and Q20 respectively, [26, 37] use a SMT solver to schedule gate operations to qubits with lower error probabilities. Ash-Saki *et al.* propose two approaches, Sub-graph Search and Greedy approach, to optimize gate-errors [4]. Circuits generated by them may suffer from long execution time due to no consideration of the minimal circuit depth.

What we consider in the qubit mapping. We want to produce solutions for the qubit mapping problem with speedup against previous works and maintain the fidelity meanwhile. Besides the coupling map, what we further concern includes the program context and the gate duration difference, which affect the design of qubit mapping. Considering these factors will help to find remapping solution with approximate optimal execution close to reality.

2.2 Motivating Examples

We use several examples written in OpenQASM [11] to explain our motivation for considering program context and gate duration difference in qubit remapping process. The two examples base on the coupling map of four physical qubits $Q_0 \sim Q_3$ and the assumed gate durations defined in Fig. 1 (a) and (b). We directly map the logical qubits `q[0]`~`q[3]` initially to physical qubits $Q_0 \sim Q_3$ for easier explanation.

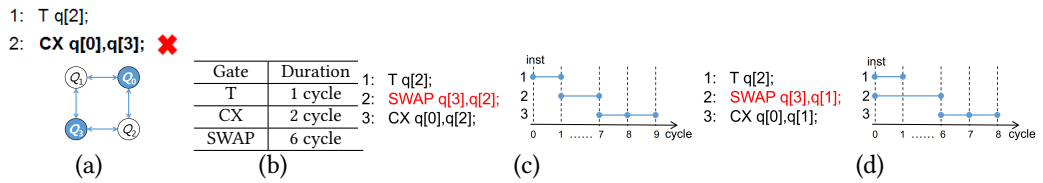

Fig. 1. An example reflecting the impact of **program context** on SWAP-based transformations: `SWAP q[3], q[1]` is selected in (d) to avoid using `q[2]` operated by the previous `T` gate, accordingly increasing parallelism.

Impact of program context. Consider the OpenQASM code fragment shown in Fig.1 (a), where `CX` means `CNOT` in OpenQASM. Since qubits Q_0 and Q_3 are non-adjacent, the instruction "`CX q[0],q[3]`" at line 2 cannot be applied. To solve the problem, `SWAP` operation is required before performing the `CX` operation. In this case, there are four candidate `SWAP` pairs, *i.e.,* (Q_0, Q_1), (Q_0, Q_2), (Q_3, Q_1) and (Q_3, Q_2). If the program context, *i.e.,* the predecessor instruction "`T q[2];`", is not considered, there are no differences among four candidates when selecting. However, `SWAP` operation on pair (Q_3, Q_2) or (Q_0, Q_2) conflicts with the context instruction "`T q[2];`" due to operating the same Q_2, and has to be executed serially after `T` operation as shown in Fig.1 (c). `SWAP` on pair (Q_3, Q_1) or (Q_3, Q_1) does not conflict with "`T q[2];`" and can be executed in parallel as shown in Fig.1 (d). With the awareness of the context information, `SWAP` operations which improve parallelism can be sifted out.

Impact of gate duration difference. We use a 4-qubit QFT (quantum fourier transform) circuit to explain the limitation of ignoring the duration of quantum gates. Fig. 2 (b) lists the fragment of

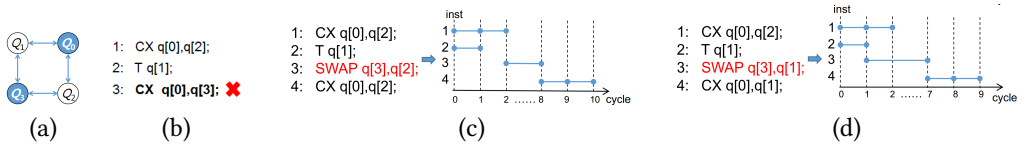

Fig. 2. A 4-qubit QFT example reflecting the impact of **gate duration difference**: "`SWAP q[3],q[1]`" is the best candidate since it can start immediately after "`T q[1]`" while "`CX q[0],q[2]`" has not finished yet, increasing the parallelism of the circuit.

a 4-qubit QFT OpenQASM program, which is generated by ScaffCC compiler [2]. Similar to the first example, `SWAP` operation is required before performing the `CX` operation and there are also the same four candidate `SWAP` pairs. Instructions "`T q[2]`" and "`CX q[0],q[2]`" can be executed in parallel and we assume both of them start at cycle 0. If the difference of gate durations is ignored, the two gates "`T q[2]`" and "`CX q[0],q[2]`" are assumed to finish at the same time t and the four candidate `SWAP` operations have to start after t. But if the duration of `CX` is twice as much as that of `T`, we find that "`T q[2]`" will finish at cycle 1 while "`CX q[0],q[2]`" at cycle 2. As a result, `SWAP` between `q[3]` and `q[1]` can start at cycle 1 as shown in Fig.2 (d), while other three candidate SWAP operations have to start at cycle 2 since one of operands Q_0 or Q_2 is occupied as shown in Fig.2 (c). Fig. 2 (d) has better parallelism, which can be deduced by the awareness of different quantum gate durations.

2.3 Quantum Architecture Abstraction

Table 2. Definition of Quantum Abstract Machine.

	Notation	Definition
Static Structure	Q_H	The set of physical qubits, $\|Q_H\| = N$; $\forall Q \in Q_H$, $Q.t_{end}$ is the qubit lock described in Section 3.1
	G	The set of elementary quantum operations and SWAP, $\|G\| = M$
	$\mathbb{M} = (Q_H, E_H)$	The coupling graph of a quantum device
	τ: $G \rightarrow \mathbb{N}$	Mapping from quantum operations to their durations, $\mathbb{N}$ represents the set of natural numbers
	$\mathbb{D}$: $Q_H \times Q_H \rightarrow \mathbb{N}$	Mapping from physical qubit pairs to their shortest path lengths on the $\mathbb{M}$, if there is no path between Q_i and Q_j, then $\mathbb{D}(Q_i, Q_j)$ = INT_MAX
Dynamic Structure	π:$Q_P \rightarrow Q_H$	Mapping from logical qubits to physical qubits
	CF(I)	Commutative Front gate set of a gate sequence I, defined in Definition 3.1
Auxiliary Functions	`gate`(g)	the name of a given gate g.
	`qseq`(g)	the logical qubit sequence applied by a given gate g.
Variables	$Q_{\{1,2,\ldots,N\}}$	Physical qubits, $Q_i \in Q_H$, $1 \le i \le N$
	$q_{\{1,2,\ldots,n\}}$	Logical qubits, $q_i \in Q_P$, $1 \le i \le n$
	$g_{\{1,2,\ldots,M\}}$	Physical quantum operations, $g_i \in G$, $1 \le i \le M$
	$g_{\{1,2,\ldots,m\}}$	Quantum operations in the circuit program
	I	A sequence of quantum operations, $I = [g_1, g_2, \ldots, g_k]$ if $k = \|I\|$, and the length of I is written as $I.len$

Since the qubit mapping problem is affected by the constraints of underlying QC devices, which base on various and evolving quantum technologies, it is essential to design quantum mapping algorithms that are compatible with different quantum technologies.

In view of the above, we consider the qubit connectivity of various NISQ devices, and take each gate duration as a multiple of quantum clock cycle τ_u, which can be analogized to the classic clock cycle. We then introduce a **M**ulti-architecture **A**daptive **Q**uantum **A**bstract **M**achine (maQAM) which consists of static and dynamic structures, denoted as $\mathbb{A} = (A_s, A_d)$. Table 2 shows the definitions for maQAM, where $A_s = (Q_H, G, \mathbb{M}, \tau, \mathbb{D})$, and $A_d = (\pi, CF)$. We assume the device can provide enough physical qubits (denote the number as N) for the program's execution (denote the number of logical qubits in the program as n), *i.e.*, $N \ge n$.

For a QC device, we abstract its qubit layout as a graph $\mathbb{M}$ where qubits are vertices and there are edges between qubit pairs where a two-qubit gate is allowed to apply on them. We introduce the Gate Duration Map τ into A_s which maps each kind of quantum gate to its duration, depending on the information from quantum architecture. We assume the same kind of quantum gates have the same duration and fidelity. We also introduce the shortest distance matrix map $\mathbb{D}$ between each pair of physical qubits for quick selection of exchangeable qubits in our CODAR scheduling algorithm.

3 DESIGN

In this section, we discuss our **CO**ntext-sensitive and **D**uration-**A**ware **R**emapping algorithm (CODAR). We first overview the idea of CODAR, then introduce the two key mechanisms that enable CODAR context-sensitivity and duration awareness.

The main idea of CODAR is to generate an executable gate sequence for a given input OpenQASM program by adjusting the gate sequence and inserting the swap operation with the program semantics unchanged. The generated gate sequence fits quantum hardware limitation on one hand, and has better parallelism on the other hand to reduce the circuit's weighted depth , *i.e.,* simulated execution time. We propose qubit lock mechanism in Section 3.1 for quickly finding available qubits. And we adjust the gate order based on the quantum gate commutativity described in Section 3.2.

3.1 Qubit Lock

CODAR is based on a reasonable assumption: *a qubit cannot be applied by two or more gates at the same time.* If a qubit is occupied by a gate, it is called *busy* (not free) qubit and cannot be applied by other gates. As the example shown in Fig. 1, when inserting `SWAP` for a specific two-qubit gate `CX q[0],q[3]`, the neighbour qubit `q[2]` of the target qubits may be occupied by the contextual gate which has started in earlier time. Using the occupied qubits to route the two-qubit gate will reduce the parallelism of the program because the routing process has to wait until occupied qubits become free.

To make CODAR aware of the qubit occupation by the past contextual gate, we introduce a **qubit lock** t_{end} for each physical qubit in Q. When start applying a quantum gate g $\in$ G at time t on a physical qubit in Q and the gate's duration is τ_g, CODAR will update this qubit's t_{end} as $t+\tau_g$ which means that it is occupied before $t+\tau_g$. A qubit is *free* only when its lock $t_{end} \leq$ current time. When try to find routing path for a specific two-qubit gate, by comparing t_{end} of each qubit with the current time, CODAR can be aware of which qubit is occupied by the past contextual gate. Fig. 3 shows an example. Gates can only be applied to the physical qubits in free state. The gates whose associate physical qubits are all free, are called *lock free* gates.

Qubit lock can also help CODAR aware of the gate duration difference. Different gate kinds have different durations and CODAR updates the operated qubit's lock t_{end} with different value. As a result, qubits applied by gates with shorter duration will be set smaller t_{end} and become free earlier. Thus CODAR can use those earlier free qubits to route two-qubit gates and improve the parallelism of the program. As the example shown in Fig. 2 (d), suppose the program starts at time 0 and τ_T=1, τ_{CNOT}=2. Then t_{end} of Q_1 is set to 1 while t_{end} of Q_0 and Q_2 are set to 2. Q_1 becomes free at time 1 while Q_2 is still busy. CODAR can use Q_1 to route for the third gate and need not wait for the freedom of Q_2.

3.2 Commutativity Detection

Qubit lock brings CODAR awareness of the past contextual gate. Considering gate commutation relation can expose more future contextual gate for CODAR to decide routing path.

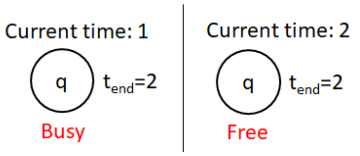

Fig. 3. Qubit lock. A 2-cycle gate is applied on qubit q at time 0, then q is busy until time 2.

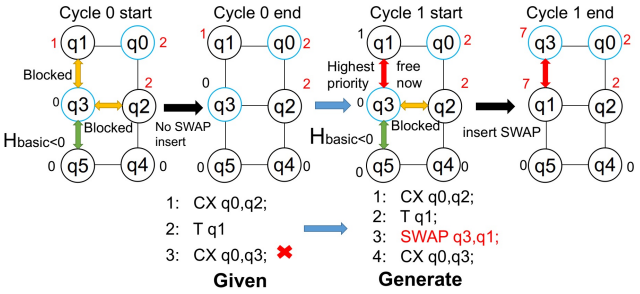

Fig. 4. Example of heuristic search for the high parallelism SWAP. The number near the qubits denote the qubit lock t_{end}.

Definition 3.1 (Commutative Forward Gate, CF gate). Given a gate sequence I=[g_1, g_2, ..., g_k, ...], $\forall$ $g_k \in I$, g_k is a commutative forward gate iff $\forall j, 0 < j < k$, g_j and g_k are commutative.

The commutation relation between two-qubit gates g_A, g_B that share qubits with each other can be resolved by checking the relevant unitary operators $\hat{A}\hat{B} = \hat{B}\hat{A}$. Gates applied to disjoint qubits are obviously commutative with each other. If a commutative forward gate is commutative with all the gates before it in sequence I, it can exchange with the head of I.

All CF gates in sequence I are denoted as $CF(I)$, which can be *executed instantly* from software perspective. Compared to the method that fetches gates with no predecessor as instantly-executable gates, choosing CF gates as instantly-executable gates can expose more contextual gates for heuristic search to determine better remapping solutions.

Suppose a sequence I contains two gates: `CX q1,q3` and `CX q2,q3` in order. The second gate shares q3 with the first and might not be regarded as instantly executable due to qubit dependence. However, because the second commutes with the first and is a CF gate in I, it is instantly executable in fact. Commutativity detection will expose both CXs for heuristic search which will improve its contextual look-ahead ability.

3.3 Example

Now we use an example shown in Fig. 4 to explain our algorithm. Suppose there is a 6-qubit device and we are given a gate sequence I that contains a CX on {q0,q2}, a T on {q1} and a CX on {q0,q3}. The number near the qubit node represents the value of its t_{end}. All the three gates are CF gates. Due to the coupling limitation, CX on {q0,q3} is not directly executable and SWAP is needed. The algorithm simulates the execution timeline and starts at cycle 0. At cycle 0, the first gate "CX q0,q2" and the second gate "T q1" are directly executable so both of them will be launched and qubits {q0,q1,q2}'s t_{end} locks are updated with the gate duration (T=1 cycle, CX=2 cycle). At cycle 0, each of {q0,q1,q2} has bigger t_{end} than current time and thus they are locked. Therefore the SWAP between {q1,q3} and {q2,q3} are blocked. SWAP between {q3,q5} with $H_{basic} < 0$ (which means the SWAP wonâĂŹt shorten the total distance of CF gates according to our heuristic cost function) moves q3 away from q0 and will not be inserted. As a result, no SWAP will be inserted in cycle 0 and the mapping π stays unchanged. At cycle 1, qubit q1 becomes free while q2 stays busy. The SWAP between {q1,q3} becomes free while the SWAP between {q3,q2} is still blocked. Therefore the algorithm will know that the SWAP between {q1,q3} can start earlier than SWAP between {q3,q2} and choose SWAP q3,q1 to solve the remapping problem. After launching the SWAP between {q1,q3}, qubit locks of {q1,q3} are also updated by the sum of its start time (cycle 1) and the duration of SWAP (6 cycle) as 7.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate CODAR with benchmarks based on the latest reported hardware models.

Comparison with Previous Algorithms. Several recent algorithms proposed by IBM [16], Siraichi *et al.* [35], Zulehner *et al.* [45] and Li *et al.* [19] try to find solutions of the qubit mapping problem with small circuit depth. Among them, Li's SABRE [19] beats the other three in the performance of benchmarks, thus it is used for comparison in this paper.

Hardware Configuration. We test our algorithm on several latest reported architectures, including IBM Q20 Tokyo[19], IBM Q16 Melbourne[1], 6×6 grid model proposed by Enfield [35]'s GitHub and Google Q54 Sycamore [3]. The gate duration difference configuration is based on experimental data of symmetric superconducting technology shown in Table 1, where two-qubit gate duration is generally twice as much as that of the single-qubit gate.

Benchmarks. To evaluate our algorithm, we totally collect 71 benchmarks which are selected from the previous work, including: 1) programs from IBM Qiskit [10]'s Github and RevLib [40]; 2) several quantum algorithms compiled from ScaffCC [2] and Quipper [14]; 3) benchmarks used in the best-known algorithm SABRE [19]. The size of the benchmarks ranges from using 3 qubits up to using 36 qubits and about 30,000 gates. For the IBM Q16, Q20 and 6×6 architectures, 68 benchmarks out of the 71 benchmarks except 3 36-qubit programs are tested. While all 71 benchmarks are tested on Google Q54 Sycamore.

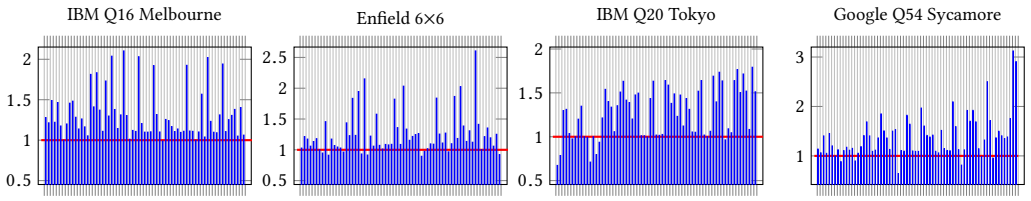

Fig. 5. Speedup ratio of all 71 benchmarks compared between CODAR and SABRE in four architectures. The benchmarks are listed from left to right in the ascending order of the number of qubits used.

Circuit Execution Speedup. We collect the weighted circuit depth of the circuits produced by CODAR and SABRE for the 71 benchmarks. Initial mapping has been proved to be significant for the qubit mapping problem, and for a fair comparison, we use the same method as SABRE to create the initial mapping for the benchmarks. We use the depth of circuits produced by SABRE compared with the one of CODAR to show the ability of our algorithm to speed up the quantum program. As shown in Fig.5, the average speedup ratio of CODAR on four architecture models, IBM Q16 Melbourne, Enfield 6×6, IBM Q20 Tokyo and Google Q54 are respectively 1.212, 1.241, 1.214 and 1.258.

5 CONCLUSION

In NISQ era, most quantum programs are not directly executable because two-qubit gates can be applied between arbitrary two logical qubits while it can only be implemented between two adjacent physical qubits due to hardware constraints. To solve this problem, in this paper we propose CODAR that can transform the origin circuit and insert necessary `SWAP` operations making the circuit comply with the hardware constraints. With the design of qubit lock and commutativity detection, CODAR is aware of program context and the gate duration difference which help CODAR

remapper find the remapping with good parallelism and reduce QC's weighted depth. Experimental results show that compared to the best known remapping algorithm, CODAR remapper can cut down 17.5% ~ 19.4% weighted depth at average.

REFERENCES

[1] IBM Q backend information. https://github.com/Qiskit/ibmq-device-information, visited in May 2019.

[2] Ali Javadi Abhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. ScaffCC: A framework for compilation and analysis of quantum computing programs. In *11th CF*, pages 1:1–1:10, New York, NY, USA, 2014. ACM.

[3] Frank Arute, Kunal Arya, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[4] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. QURE: Qubit re-allocation in noisy intermediate-scale quantum computers. In *56th DAC*, pages 141:1–6, 2019.

[5] Anirban Bhattacharjee, Chandan Bandyopadhyay, Robert Wille, Rolf Drechsler, and Hafizur Rahaman. A novel approach for nearest neighbor realization of 2D quantum circuits. In *ISVLSI*, pages 305–310. IEEE, 2018.

[6] Debjyoti Bhattacharjee and Anupam Chattopadhyay. Depth-optimal quantum circuit placement for arbitrary topologies. *arXiv preprint arXiv:1703.08540*, 2017.

[7] Kyle E. C. Booth, Minh Do, J. Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *28th ICAPS*, pages 366–374, 2018.

[8] Amlan Chakrabarti, Susmita Sur-Kolay, and Ayan Chaudhury. Linear nearest neighbor synthesis of reversible circuits by graph partitioning. *arXiv preprint arXiv:1112.0564*, 2011.

[9] Frederic T. Chong, Diana Franklin, and Margaret Martonosi. Programming languages and compiler design for realistic quantum hardware. *Nature*, 549:180âĂŞ187, 2017.

[10] Andrew Cross. The IBM Q experience and QISKit open-source quantum computing software. *Bulletin of the American Physical Society*, 2018.

[11] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open quantum assembly language. *arXiv:1707.03429*, 2017.

[12] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proc. of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[13] A Fuhrmanek, R Bourgain, Yvan RP Sortais, and Antoine Browaeys. Free-space lossless state detection of a single trapped atom. *Physical review letters*, 106(13):133003, 2011.

[14] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In *34th PLDI*, pages 333–342. ACM, 2013.

[15] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th STOC*, pages 212–219, New York, NY, USA, 1996. ACM.

[16] IBM. Qiskit. https://qiskit.org/, visited in Jan 2019.

[17] Abhoy Kole, Kamalika Datta, and Indranil Sengupta. A heuristic for linear nearest neighbor realization of quantum circuits by SWAP gate insertion using N-gate lookahead. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(1):62–72, 2016.

[18] Abhoy Kole, Kamalika Datta, and Indranil Sengupta. A new heuristic for N-dimensional nearest neighbor realization of a quantum circuit. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):182–192, 2018.

[19] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In *24th ASPLOS*, pages 1001–1014. ACM, 2019.

[20] Chia-Chun Lin, Susmita Sur-Kolay, and Niraj K Jha. PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(7):1221–1234, 2014.

[21] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

[22] Aaron Lye, Robert Wille, and Rolf Drechsler. Determining the minimal number of SWAP gates for multi-dimensional nearest neighbor quantum circuits. In *20th ASP-DAC*, pages 178–183. IEEE, 2015.

[23] KM Maller, MT Lichtman, T Xia, Y Sun, MJ Piotrowicz, AW Carr, L Isenhower, and M Saffman. Rydberg-blockade controlled-not gate and entanglement in a two-dimensional array of neutral-atom qubits. *Physical Review A*, 92(2):022336, 2015.

[24] Miguel Martinez-Dorantes, Wolfgang Alt, Jose Gallego, Sutapa Ghosh, Lothar Ratschbacher, Yannik Völzke, and Dieter Meschede. Fast nondestructive parallel readout of neutral atom registers in optical potentials. *Physical review letters*, 119(18):180503, 2017.

[25] Dmitri Maslov, Gerhard W Dueck, D Michael Miller, and Camille Negrevergne. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):436–444, 2008.

[26] Prakash Murali, Jonathan M Baker, Ali Javadi Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *24th ASPLOS*, pages 1015–1029, Providence, RI, USA, 2019. ACM.

[27] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, UK, 10th Anniversary edition, 2010.

[28] Angelo Oddi and Riccardo Rasconi. Greedy randomized search for scalable compilation of quantum circuits. In *CPAIOR*, pages 446–461. Springer, 2018.

[29] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 2011.

[30] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *50th DAC*, page 41. ACM, 2013.

[31] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize communication overhead in 2D quantum architectures. In *19th ASP-DAC*, pages 495–500. IEEE, 2014.

[32] Cheng Sheng, Xiaodong He, Peng Xu, Ruijun Guo, Kunpeng Wang, Zongyuan Xiong, Min Liu, Jin Wang, and Mingsheng Zhan. High-fidelity single-qubit gates on neutral atoms in a two-dimensional magic-intensity optical dipole trap array. *Phys. Rev. Lett.*, 121:240501, Dec 2018.

[33] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[34] Ritu Ranjan Shrivastwa, Kamalika Datta, and Indranil Sengupta. Fast qubit placement in 2D architecture using nearest neighbor realization. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 95–100. IEEE, 2015.

[35] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In *CGO 2018*, pages 113–125, New York, NY, USA, 2018. ACM.

[36] Robert Stockill, M.J. Stanley, Lukas Huthmacher, E. Clarke, M. Hugues, A.J. Miller, C. Matthiesen, C. Le Gall, and Mete Atatüre. Phase-tuned entangled state generation between distant spin qubits. *Physical review letters*, 119(1):010503, 2017.

[37] Swamit S Tannu and Moinuddin K. Qureshi. Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers. In *24th ASPLOS*, pages 987–999. ACM, 2019.

[38] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, 2018.

[39] Davide Venturelli, Minh Do, Eleanor G Rieffel, and Jeremy Frank. Temporal planning for compilation of quantum approximate optimization circuits. In *26th IJCAI*, pages 4440–4446, 2017.

[40] R. Wille, D. GroÃ§e, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An online resource for reversible functions and reversible circuits. In *38th ISMVL*, pages 220–225, May 2008.

[41] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In *56th DAC*, page 142. ACM, 2019.

[42] Robert Wille, Oliver Keszocze, Marcel Walter, Patrick Rohrs, Anupam Chattopadhyay, and Rolf Drechsler. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In *21st ASP-DAC*, pages 292–297. IEEE, 2016.

[43] Robert Wille, Aaron Lye, and Rolf Drechsler. Optimal SWAP gate insertion for nearest neighbor quantum circuits. In *19th ASP-DAC*, pages 489–494. IEEE, 2014.

[44] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J. S. Chen, N. C. Pisenti, et al. Benchmarking an 11-qubit quantum computer. *arXiv e-prints*, page arXiv:1903.08181, Mar 2019.

[45] A. Zulehner, A. Paler, and R. Wille. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(7):1226–1236, July 2019.