Mitigating Crosstalk in Quantum Computers through Commutativity-Based Instruction Reordering

Lei Xie, Jidong Zhai, Weimin Zheng
Tsinghua University
Email: xie-l18@mails.tsinghua.edu.cn, {zhaijidong, zwm-dcs}@tsinghua.edu.cn

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586145

***Abstract*—Crosstalk is one of the major types of noise in quantum computers. To design high-fidelity quantum gates and large-scale quantum computers, effectively suppressing crosstalk is becoming increasingly important. Previous approaches to mitigate crosstalk rely on either hardware strategies, which are only applicable on limited platforms, or software techniques, which, however, cannot fully explore instruction parallelism. To address the above challenges, we propose a commutativity-based compile-time instruction reordering approach. We first propose a complete set of generalized commutativity rules for main types of quantum gates, and then we design a two-level bidirectional reordering algorithm to reorder gates according to these rules for effective crosstalk mitigation. Our approach can not only capture both forward and backward instruction correlations but also reduce the effect of decoherence. We evaluate our approach with 117 quantum programs. Compared with the state-of-the-art, our approach can improve program fidelity by up to 120% (18% on average).**

***Index Terms*—Quantum Computing, Error Mitigation, Crosstalk, Commutativity, Compiler Optimization**

I. Introduction

Quantum computing is expected to solve many classically intractable problems [1]–[3]. However, due to imperfect manufacture and imprecise quantum control, current Noisy Intermediate-Scale Quantum (NISQ) computers suffer from severe noise [4], [5], which limits their ability to execute valuable applications [6]. Therefore, suppressing noise effectively is vital for practical quantum computing.

Crosstalk has been identified as a major type of noise in many leading architectures including both superconducting and ion trap [7]–[9] where crosstalk arises from unwanted qubit interactions especially *when executing multiple quantum gates (instructions) simultaneously*. Crosstalk has two major types of hazards. One is degrading gate fidelity severely. For instance, we observe a 20× error rate increase for CNOT gates due to crosstalk on IBM quantum computers (Section II). Such degradation can lead to low program fidelity and even a wrong output [4], [5], [10]. The other hazard is spreading the effect of local quantum gates to the global. Even though a gate is perfect on a subset of qubits, it can still affect other qubits through unwanted interactions, causing a non-local noise [11]. However, quantum error correction, a key technique towards fault-tolerant quantum computing, is usually based on local noise assumption [12], the non-local noise can invalidate this assumption and the technique.

Previous approaches to mitigate crosstalk can be divided into two categories, hardware strategies and software techniques. Typical hardware strategies include tunable qubit coupling [13] and smart frequency allocation [14]. The main drawback of hardware strategies is their limited applicability. For example, tunable qubit coupling requires coupling to turn on and off dynamically, which cannot apply to IBM fixed-coupling computers [15]. And the frequency allocation strategy assigns qubit frequency statically, which is inadequate for Google tunable-qubit computers where qubit frequency is dynamically adjusted to maintain system reliability [3].

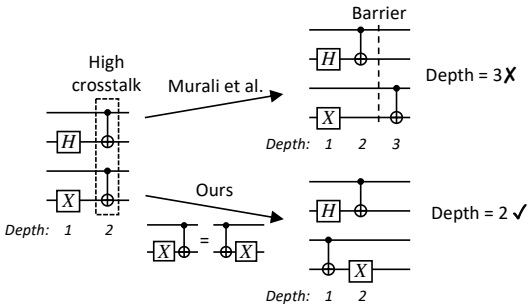


Fig. 1: To mitigate the high crosstalk between marked CNOTs, the scheduling approach of Murali et al. [8] inserts a barrier, which incurs more decoherence errors, while ours can avoid the unnecessary decoherence. (All examples in this paper assume that gates have the same duration, while the real case is considered in our evaluation.)

Recently, Murali et al. [8] propose an instruction scheduling approach at the software level. For crosstalk mitigation, they insert barriers among simultaneous instructions to serialize their execution. However, as serialization reduces instruction parallelism, it increases execution time and causes more decoherence errors. Fig. 1 shows that their approach mitigates crosstalk at the cost of a longer execution.

To address the above challenges, we propose a novel commutativity-based compile-time instruction reordering approach for crosstalk mitigation. We design our approach based on a typical quantum circuit model [12] to apply to a wide range of hardware. Besides, our reordering approach does not serialize any gates, so we keep the maximum instruction parallelism.

Specifically, we first propose a *complete* set of *generalized* commutativity rules for CNOTs and SWAPs. Then we design a two-level bidirectional reordering algorithm to utilize these rules to move gates with high crosstalk so as to separate their execution and finally to mitigate crosstalk (Fig. 1 below gives an example of our approach.). Through bidirectional reordering, our approach can capture both forward and backward instruction correlations (defined in Section III-C1). We further analyze the trade-off between crosstalk and decoherence and consider both of them in a cost function to guide reordering decisions.

In this work, we make three main contributions.

- To better understand crosstalk, we perform a quantitative analysis of crosstalk on a real NISQ computer and uncover the causes of crosstalk from the perspective of quantum programs.
- We propose a set of generalized commutativity rules for both CNOTs and SWAPs, and we further prove the completeness of these rules. To the best of our knowledge, we are the first to propose these generalized rules.
- We design a two-level bidirectional reordering algorithm to mitigate crosstalk based on our proposed commutativity rules.

We also reveal a backward instruction correlation in quantum programs.

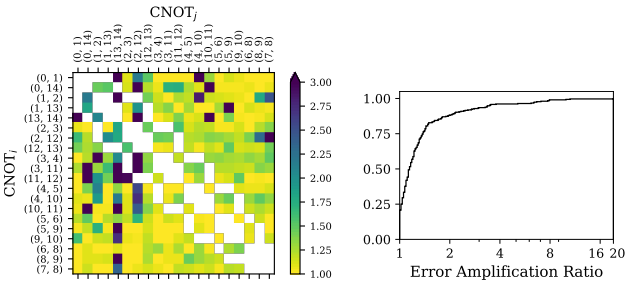
We evaluate our approach using 117 quantum programs on two different hardware topologies. The results show that our approach can significantly improve program fidelity by up to 120% (18% on average) compared with the state-of-the-art. Moreover, an in-depth analysis shows that our approach also achieves a *win-win* result that both crosstalk and decoherence are mitigated simultaneously.

II. Crosstalk Characterization and Cause Analysis

In this section, we first study the crosstalk effect on gate error rates on a real NISQ computer. Next, we analyze the main causes of crosstalk.

A. Crosstalk Characterization

Randomized Benchmarking (RB) [16], [17] is a standard protocol for evaluating gate error rates. To characterize the crosstalk effect between two gates i and j, we first perform RB on each of them separately to obtain independent error rates e_i, e_j, then we perform RB on both of them simultaneously for simultaneous error rates $e_{i|j}, e_{j|i}$. We use an error amplification ratio $r_{i|j} = e_{i|j}/e_i$ as the indicator for the effect of crosstalk between two gates i and j on the gate i. Previous studies have demonstrated that simultaneous CNOT execution is a main cause of crosstalk [8], [11], thus we mainly focus on crosstalk between CNOT pairs in this work.



(a) Crosstalk matrix with item i, j representing $r_{i|j}$.

(b) Cumulative distribution of error amplification ratios.

Fig. 2: Crosstalk characterization of IBMQ Melbourne. (Darker colors mean higher crosstalk. CNOT pairs with white color cannot be executed simultaneously because they share the same qubit(s).)

Fig. 2 presents the error amplification ratios on IBMQ Melbourne, a real 15-qubit NISQ computer [18]. It is shown that crosstalk is prevalent between almost all CNOT pairs and it can be very severe. For instance, more than 12% of the CNOT pairs have an error amplification ratio larger than 2, which means that error is amplified by 100%. And in the worst case, crosstalk degrades a gate by 20×.

Since CNOTs provide entanglement, which is a vital resource for quantum computing, such degradation can lead to low program fidelity and even a wrong output [4], [5], [10]. Moreover, other architectures, such as ion trap [9], have also reported severe gate fidelity degradation due to crosstalk. Therefore, **mitigating crosstalk is necessary and crucial for practical quantum computing**.

B. Cause Analysis

Since simultaneous CNOT execution is a main cause of crosstalk [8], [11], the count of Simultaneous CNOTs (S-CNOTs) is an important metric for crosstalk analysis from the perspective of quantum programs. Through intensive analysis of 117 programs (Section IV) on IBMQ Melbourne, we summarize two key causes resulting in S-CNOTs: program parallelism and SWAPs due to qubit mapping and routing.

1) Program Parallelism: Parallelism in quantum programs refers to simultaneous instruction streams on independent qubits. Since different streams proceed in parallel, they can introduce S-CNOTs. For example, a program for a 4-qubit hidden shift problem [19] shown in Fig. 3 has two parallel instruction streams, and each CNOT is executed along with another, resulting in high crosstalk.

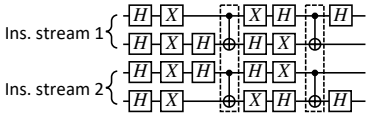
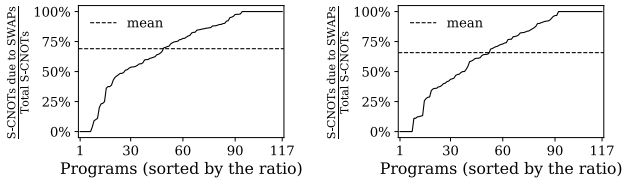


Fig. 3: Parallelism and S-CNOTs in a 4-qubit hidden shift program.

2) SWAPs due to Qubit Mapping and Routing: Qubit mapping and routing is a necessary step to compile quantum programs to satisfy the connectivity constraints of real quantum computers. Common techniques [4], [5], [20], [21] are to insert SWAPs in programs, and a SWAP is further decomposed into 3 CNOTs. To reduce decoherence errors, they favor inserting simultaneous SWAPs to minimize program execution time, which can result in S-CNOTs after decomposition and cause severe crosstalk.

We find that, in the above two causes, SWAPs due to qubit mapping and routing is the main one. To show its severity, we perform an analysis using two typical mapping and routing algorithms, the state-of-the-art SABRE algorithm [22] and IBM Qiskit stochastic algorithm [23]. Fig. 4 shows that, for the evaluated programs, 70% and 65% of S-CNOTs result from SWAPs after mapping and routing with the two algorithms respectively. Therefore, optimizing SWAPs is much more important for crosstalk mitigation.



(a) SABRE algorithm

(b) Qiskit stochastic algorithm

Fig. 4: Ratio of S-CNOTs due to SWAPs to total S-CNOTs.

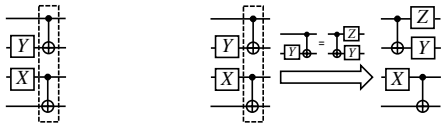
III. Commutativity-Based Instruction Reordering

In this section, we elaborate on our commutativity-based instruction reordering approach. We first present our key techniques and then our reordering algorithm.

A. Generalized Commutativity Rules for CNOTs and SWAPs

In our approach, we leverage commutativity to break data dependence among instructions so that those with high crosstalk can be reordered and separated, and finally to mitigate crosstalk. We propose *generalized* commutativity rules for CNOTs and SWAPs in this work.

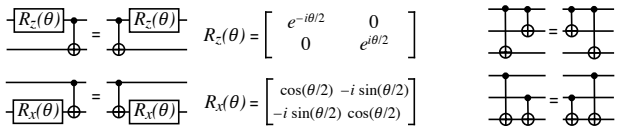
1) Rules for CNOTs: Previous work considering commutativity for CNOTs (e.g., [20], [24]–[26]) focuses on basic rules with the form $U_1C(x,y) = C(x,y)U_1$ and inter-CNOT rules (Fig. 6 (a),(b)), where U_1 is a single qubit gate and $C(x,y)$ is a CNOT gate from qubit x to y. We find it insufficient for these rules to fully explore the potential of instruction reordering. Fig. 5 shows a typical crosstalk case that cannot be mitigated with these rules, but, by allowing inserting an additional single qubit gate after swapping two gates, it can be mitigated.



(a) Since X is not of R_z type and Y is not of R_x type, no basic rules can apply to this case.

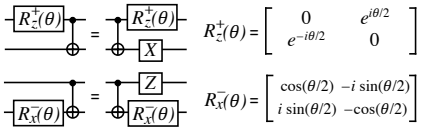
(b) This crosstalk is mitigated by allowing inserting an additional gate after swapping two gates.

Fig. 5: A crosstalk case that cannot be mitigated by basic rules.



(a) Basic rules

(b) Inter-CNOT



(c) Generalized rules

Fig. 6: Commutativity rules for CNOTs.

Therefore, we study *generalized* rules that allow inserting additional single qubit gates after swapping gates, i.e., $U_1C(x,y) = C(x,y)U_2\boldsymbol{U_3}$ where U_3 is the additional inserted gate. Note that one additional gate is sufficient because consecutive single qubit gates can be merged into one. The rules are listed in Fig. 6(c). We give the following theorem stating that these generalized rules are complete [27].

Theorem 1: All the cases for generalized commutativity rules with the form $U_1(x)C(x,y) = C(x,y)U_2(x)U_3(y)$ or $U_1(y)C(x,y) = C(x,y)U_2(x)U_3(y)$, where $U_1(x), U_2(x), U_3(x)$ are single qubit gates on qubit x and $C(x,y)$ is a CNOT gate from qubit x to y, are listed in Fig. 6.

Proof: Two equivalent gates must map product states to product states. Therefore, we can choose specific product states as the input, compare their outputs, and finally infer $U_{1,2,3}$. For $U_1(x)C(x,y) = C(x,y)U_2(x)U_3(y)$, we choose $|00\rangle$ as the input, the outputs can be described as $(s|0\rangle + t|1\rangle)|0\rangle$ (LHS) and $(ac|00\rangle + ad|01\rangle + bc|11\rangle + bd|10\rangle)$ (RHS). Then we have $ad = bc = 0$, from which we can infer that $s = 0$ or $t = 0$, i.e., U_1 maps $|0\rangle$ to $|0\rangle$ or $|1\rangle$. Thus, U_1 must be one of I, R_z, R_z^+. By substituting each of them for U_1, we can obtain U_2 and U_3. For the other rule, we can prove it similarly with $\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)$ as the input. ■

2) Rules for SWAPs: To tackle crosstalk caused by SWAPs, we can decompose SWAPs into CNOTs and reorder them based on the commutativity rules for CNOTs. However, Fig. 7 shows a crosstalk case that cannot be mitigated with this approach. But if we regard the SWAP as a whole, we can mitigate the crosstalk with commutativity for SWAPs. We present the commutativity rules for SWAPs in Fig. 8.

In principle, SWAPs commute with all gates, while in practice, SWAPs are inserted to satisfy the connectivity of underlying hardware. Thus, there is an additional connectivity constraint. We summarize our rules and the constraint in the following theorem. The theorem also states the completeness of our rules.

Theorem 2: A gate commutes with a SWAP if and only if it still satisfies the connectivity of hardware after applying the rules in Fig. 8.

Proof: The sufficient condition can be proved by definition. The necessary condition is actually a completeness statement. To prove it,

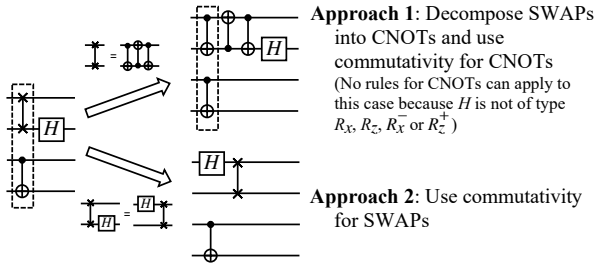


Fig. 7: A crosstalk case that cannot be mitigated by decomposition.

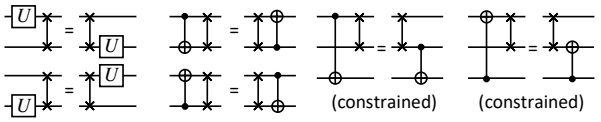
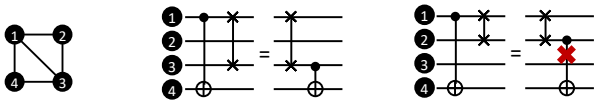


Fig. 8: Commutativity rules for SWAPs. (U is an arbitrary single qubit gate.)

we consider a complete gate set of single qubit gates and CNOT [12]. Since the rules in Fig. 8 include all the cases for this complete gate set, the rules are also complete. ■

Fig. 9 illustrates the connectivity constraint with examples.



(a) Connectivity (b) Constraint satisfied (c) Constraint violated

Fig. 9: Examples to illustrate the connectivity constraint. CONTs can be performed only between connected qubits. Example (b) satisfies the constraint because ③ and ④ are connected, while (c) violates the constraint because ② and ④ are disconnected.

B. A Cost Function Considering Both Crosstalk and Decoherence

Although we avoid serialization in contrast to previous scheduling approaches [8], there are still some cases that we need to balance crosstalk and decoherence. We present a trade-off example in Fig. 10 (where the depth of a program is the count of gates on its critical path, and it is a decisive factor for decoherence).

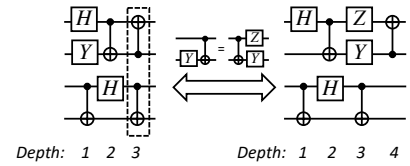


Fig. 10: A trade-off example. *Left:* Higher crosstalk with a shorter depth. *Right:* Lower crosstalk with a longer depth.

Since *both* crosstalk and decoherence can affect quantum program fidelity, we should achieve a good balance between them. Specifically, we need a cost function to evaluate the fidelity of programs generated by different reordering decisions, and the cost function should consider comprehensively both the two factors.

Our core idea is to *equivalently transform the effect* of crosstalk to decoherence by answering the following question: How many depths can be sacrificed to mitigate crosstalk such that program fidelity remains unchanged before and after mitigation?

Given a program, where $\mathcal{S}$ denotes the set of S-CNOTs (a SWAP is regarded as 3 CNOTs), $\mathcal{O}$ denotes the set of other gates, and the

error rate of gate g is e_g, the program fidelity can be estimated using the Estimated Success Probability (ESP) metric [5]:

$$\text{ESP} = \prod_{s \in \mathcal{S}} (1 - e_s) \prod_{o \in \mathcal{O}} (1 - e_o) \tag{1}$$

Suppose that, after reordering, the count of S-CNOTs is reduced by 1 and the depth remains unchanged. Thus, the ESP now can be written as

$$\text{ESP}' = \prod_{s' \in \mathcal{S}'} (1 - e_{s'}) \prod_{o' \in \mathcal{O}'} (1 - e_{o'}) \tag{2}$$

where $|\mathcal{S}'| = |\mathcal{S}| - 1$ and $|\mathcal{O}'| = |\mathcal{O}| + 1$. To counteract the benefit of reducing S-CNOTs, we append D identity gates to the critical path so that decoherence errors increase due to the increased depth. Now the ESP is

$$\text{ESP}'' = \text{ESP}' \cdot \prod_{i=1}^{D} (1 - e_i) \tag{3}$$

If program fidelity is unchanged after the above reordering and appending (i.e., $\text{ESP}'' = \text{ESP}$), then D can serve as the quantity indicating how many depths one S-CNOT can be transformed to. In other words, the effect of crosstalk is equivalently transformed to decoherence. Using the average independent gate error rate $\bar{e}_{ind}$ for e_o ($o \in \mathcal{O}$) and the average simultaneous gate error rate $\bar{e}_{sim}$ for e_s ($s \in \mathcal{S}$), we can derive D as:

$$D = \log_{1-e_i} \left[\frac{1 - \bar{e}_{sim}}{1 - \bar{e}_{ind}} \right] \tag{4}$$

where e_i is the identity gate error rate representing decoherence effect. Our cost function is designed by transforming all the effect of crosstalk to decoherence:

$$\text{Cost} = D \cdot |\mathcal{S}| + \text{Depth} \tag{5}$$

According to the characterization results in Section II, for IBMQ Melbourne, $\bar{e}_{ind} = 0.033$, $\bar{e}_{sim} = 0.052$, $e_i = 0.002$; thus $D \approx 10$.

C. Two-Level Bidirectional Reordering Algorithm

In this section, we present our reordering algorithm. Bidirectional reordering aims to capture both forward and backward instruction correlations (described later), and two-level reordering aims to better utilize both high-level commutativity for SWAPs and fine-grained commutativity for CNOTs.

The basic reordering procedure is shown in Algorithm 1 REORDER. A quantum program is represented as a directed acyclic graph (DAG) where nodes denote instructions (gates) and edges denote qubits. We follow the topological order of the DAG to ensure data dependency but utilize commutativity rules to break the dependency and then reorder instructions. The reordering procedure is based on beam search where our proposed cost function is used to evaluate each reordering decision. Finally, our algorithm ends with a reordered program of the minimum cost for best crosstalk mitigation.

1) Bidirectional Reordering: Reordering according to topological order (forward reordering) can capture *forward correlation* well, i.e., reordering decisions on preceding instructions can affect the succeeding such that they together improve program fidelity further. However, it cannot deal with *backward correlation*. Fig. 11 gives an example for such case.

The observation from Fig. 11 motivates us to consider backward reordering according to *reversed* topological order. Specifically, we perform several iterations composed of both forward and backward reordering until we reach an optimal reordered program (BIREORDER). In practice, we find it sufficient to find a good solution after only one iteration for most programs (Section IV).

Algorithm 1: Commutativity-based Instruction Reordering

```
Procedure REORDER(Program P, Gate Type T)
    gates ← topological_gate_list(P, T)
    candidate_list ← [P]
    for gate in gates do
        new_candidates ← ∅
        for candidate in candidate_list do
            update new_candidates by reordering gate in the
              candidate program according to the
              commutativity rules for T
        calculate cost for each new candidate
        candidate_list.append(new_candidates)
        sort candidate_list by their costs
        trim candidate_list to the first N candidates
    return the candidate with the minimum cost
Procedure BIREORDER(Program P, Gate Type T)
    Do
        P ← REORDER(P, T)
        P ← reversed(REORDER(reversed(P), T))
    until cost is unchanged
    return the reordered program P
Procedure TWOLEVELREORDER(Program P)
    P ← BIREORDER(P, SWAP & CNOT)
    Decompose SWAPs of P into CNOTs
    P ← BIREORDER(P, CNOT)
    return the reordered program P
```

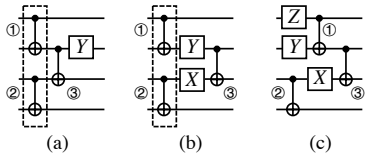


Fig. 11: To mitigate the crosstalk between CNOT ① and ② in (a), if we perform forward reordering (①→②→③), we can **only** swap ③ and Y as shown in (b), which fails to mitigate the crosstalk. But for backward reordering (③→②→①), we can swap ③ and Y (b), and then **swap** ① **and Y** (c), thus mitigate successfully. (Note since $Y = R_x^-(\pi) = R_z^+(-\pi)$, it commutes with ③ and ①.)

2) Two-Level Reordering: Since SWAPs can be decomposed into a set of CNOTs, we can deal with SWAPs with either high-level commutativity rules for SWAPs or fine-grained commutativity rules for CNOTs. To better utilize the two kinds of commutativity, we propose a two-level reordering strategy (TWOLEVELREORDER). We first reorder instructions by regarding a SWAP as a whole to utilize the high-level commutativity. Then we decompose SWAPs into CNOTs and reorder CNOTs to exploit the fine-grained commutativity. In this way, we achieve both high-level and fine-grained optimization.

D. Implementation

We implement our reordering approach at the compile time of quantum programs. It works in compilers as an optimization pass after mapping and routing qubits (Fig. 12).

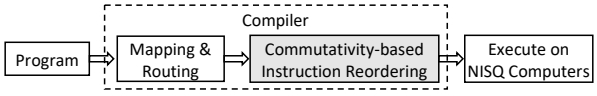


Fig. 12: Implementing instruction reordering in quantum compilers.

IV. Evaluation

In this section, we present how program fidelity can be improved with our approach and also give a breakdown analysis of our key techniques.

A. Methodology

Benchmarks We select 117 representative quantum programs from diverse applications such as reversible circuits [28], hidden group problems [19], quantum Fourier transform [12], phase estimation [1], and optimization [2]. The scale of these programs ranges from 20 instructions up to 500 instructions for matching the program scale in the NISQ era and the near future.

Hardware and Noise Model for Simulation We evaluate our approach with the topology of both IBMQ Melbourne [18] and Rigetti Aspen-8 [29]. We use deploring error for modeling gate errors and thermal relaxation error for decoherence, which is a standard way widely used in literature [30]–[32]. According to Section II, to model crosstalk, we analyze S-CNOTs in a program and increase their error rates. All the error rates are set according to the characterization results on IBMQ Melbourne but with a 10× reduction because the scale of the selected benchmarks is too large for current quantum computers [33], [34], and near-future computers are expected to have a 10× reduction in error rates [6]. And the duration is set to 1 cycle for single qubit gates and 2 cycles for CNOTs according to previous studies [21]. We perform simulation using IBM Qiskit [23].

Comparison We compare our approach with ParSched [8], which is the state-of-the-art scheduling algorithm used on IBM, Google, and Rigetti quantum computers [3], [10]. It executes maximum instructions in parallel to reduce decoherence.

B. Fidelity Improvement

We define program fidelity as the Probability of Successful Trials (PST), a widely-used metric in previous work [4], [5], [35]. We perform 4096 trials to evaluate fidelity.

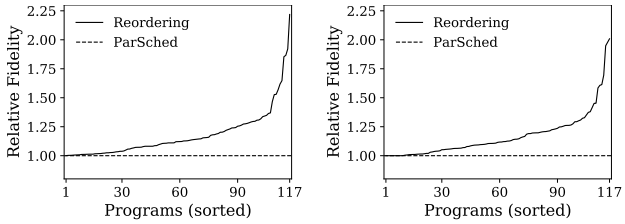
$$\text{PST} = \frac{\#\text{Successful Trials}}{\#\text{Total Trials}} \tag{6}$$

We compare the fidelity using our approach versus ParSched in Fig. 13. There are two main results. **First, our approach improves program fidelity significantly.** The maximum improvement on both topologies exceeds 100% and the average is about 18%. **Second, our approach can adapt to different topologies and utilize complicated topologies effectively.** IBMQ Melbourne has a dense 2-D mesh topology while Rigetti Aspen-8 has a sparse octagonal ring topology. Dense topologies provide more exploration possibilities but also increase difficulty. The results show that we achieve a better improvement on the dense topology, revealing the ability of our approach to utilize complicated topologies.

C. Analysis of Fidelity Improvement

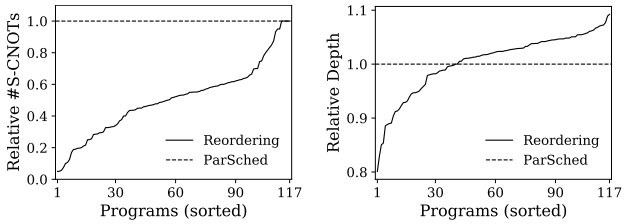
We analyze the count of S-CNOTs and the depth of programs using our approach versus ParSched to reveal improvement causes. Fig. 14 shows the statistics with the IBMQ Melbourne topology. It is shown that **our approach can eliminate up to 95% of S-CNOTs with depth increasing by no more than 10%**, which reveals that our approach manages to mitigate crosstalk with very little additional decoherence incurred.

More importantly, it is also shown that, for many programs (whose number is less than 40 in Fig. 14(b)), our approach achieves **a *win-win* result that both the count of S-CNOTs and the depth of programs are reduced**. Fig. 15(a) gives a win-win case found by our approach in a 5-qubit quantum Fourier transform program. The



(a) IBMQ Melbourne (b) Rigetti Aspen-8

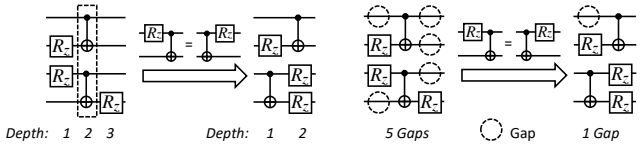
Fig. 13: Fidelity comparison using our approach versus ParSched. (Higher is better. Note that, for all the charts in this section, programs are sorted according to the values on the y-axis, and they are represented by digital numbers on the x-axis.)



(a) #S-CNOTs (b) Depth

Fig. 14: The count of S-CNOTs and the depth of programs using our approach versus ParSched. (For both metrics, lower is better.)

main reason for depth reduction is that there are many gaps between instructions due to data dependence, and our approach can utilize commutativity to break data dependence and fill in some gaps by reordering instructions. As the count of gaps decreases, the depth is also reduced. Fig. 15(b) illustrates the simultaneous reduction of both the count of gaps and the depth for the above win-win case.



(a) A win-win case that both the count of S-CNOTs and the depth are reduced.

(b) Depth reduction reason: reordering reduces the count of gaps between instructions.

Fig. 15: A win-win case (showing only the relevant qubits and gates).

D. Breakdown Analysis of Our Key Techniques

We perform a breakdown analysis of the key techniques in our reordering approach, generalized commutativity, the two-level reordering strategy, and the bidirectional reordering strategy.

To investigate the contribution of generalized commutativity and the two-level reordering strategy, we compare our approach with its two variants: one considers only basic rules and the other decomposes SWAPs directly. Fig. 16 and Fig. 17 demonstrate **the importance of the two techniques by showing that program fidelity can be improved by up to 24% and 36% respectively**.

For the bidirectional reordering strategy, we compare our approach with its variant that performs only forward reordering once. Fig. 18(a) shows that the bidirectional reordering strategy can improve program fidelity by up to 22%. We also calculate the distribution of the count of forward and backward iterations required for a program to reach its optimal reordering solution in Fig. 18(b). It is shown that more than 90% of the evaluated programs require only one iteration.

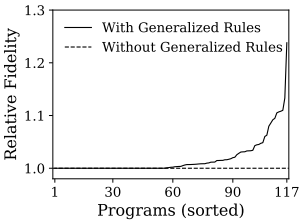


Fig. 16: Contribution of generalized commutativity. (Higher is better.)

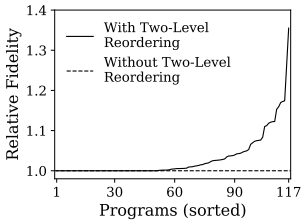
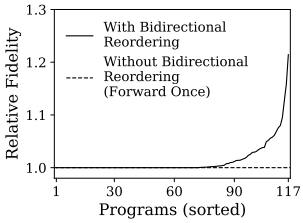
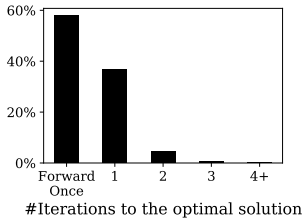


Fig. 17: Contribution of the two-level reordering strategy. (Higher is better.)



(a) Contribution of the bidirectional reordering strategy. (Higher is better.)



(b) Distribution of #iterations to the optimal reordering solution.

Fig. 18: Breakdown analysis of the bidirectional reordering strategy.

The two results imply that **it is insignificant how many iterations we perform bidirectional reordering, instead, it is necessary to perform bidirectional reordering.**

V. Conclusion

Crosstalk is a major type of noise in quantum computers. In this work, we propose a novel commutativity-based instruction reordering approach to mitigate crosstalk. Our approach features a complete set of generalized commutativity rules for both CNOTs and SWAPs. It balances crosstalk and decoherence by equivalently transforming the effect of crosstalk to decoherence in a cost function. And it also captures both forward and backward instruction correlations through bidirectional reordering. Our evaluation of 117 quantum programs demonstrates that our approach achieves high effectiveness.

Acknowledgement

We thank anonymous reviewers for their valuable comments and suggestions. We thank Mingsheng Ying for the suggestion about the commutativity rules for CNOTs. This work is partially supported by National Key R&D Program of China (2018YFA0306702) and National Natural Science Foundation of China (U20A20226). Jidong Zhai is the corresponding author of this paper.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, p. 1484–1509, Oct. 1997.

[2] A. Peruzzo, J. McClean *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, Jul 2014.

[3] F. Arute, K. Arya *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct 2019.

[4] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers," in *ASPLOS*, 2019, p. 987–999.

[5] P. Murali, J. M. Baker *et al.*, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *ASPLOS*, 2019, p. 1015–1029.

[6] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[7] C. Neill, P. Roushan *et al.*, "A blueprint for demonstrating quantum supremacy with superconducting qubits," *Science*, vol. 360, no. 6385, pp. 195–199, 2018.

[8] P. Murali, D. C. Mckay *et al.*, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *ASPLOS*, 2020, p. 1001–1016.

[9] C. Figgatt, A. Ostrander *et al.*, "Parallel entangling operations on a universal ion-trap quantum computer," *Nature*, vol. 572, no. 7769, p. 368–372, Jul 2019.

[10] P. Murali, N. M. Linke *et al.*, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *ISCA*, 2019, p. 527–540.

[11] A. Winick, J. J. Wallman, and J. Emerson, "Simulating and mitigating crosstalk," *arXiv:2006.09596*, 2020.

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

[13] P. Mundada, G. Zhang *et al.*, "Suppression of qubit crosstalk in a tunable coupling superconducting circuit," *Phys. Rev. Applied*, vol. 12, p. 054023, Nov 2019.

[14] M. Brink, J. M. Chow *et al.*, "Device challenges for near term superconducting quantum processors: frequency collisions," in *IEDM*, 2018, pp. 6.1.1–6.1.3.

[15] M. Steffen, D. P. DiVincenzo *et al.*, "Quantum computing: An ibm perspective," *IBM Journal of Research and Development*, vol. 55, no. 5, pp. 13:1–13:11, 2011.

[16] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and robust randomized benchmarking of quantum processes," *Phys. Rev. Lett.*, vol. 106, p. 180504, May 2011.

[17] J. M. Gambetta, A. D. Córcoles *et al.*, "Characterization of addressability by simultaneous randomized benchmarking," *Phys. Rev. Lett.*, vol. 109, p. 240504, Dec 2012.

[18] "Ibm quantum experience," https://quantum-computing.ibm.com/.

[19] M. Rötteler, "Quantum algorithms for highly non-linear boolean functions," in *ACM-SIAM Symposium on Discrete Algorithms*, 2010, p. 448–457.

[20] G. G. Guerreschi and J. Park, "Two-step approach to scheduling quantum circuits," *Quantum Science and Technology*, vol. 3, no. 4, p. 045003, jul 2018.

[21] H. Deng, Y. Zhang, and Q. Li, "Codar: A contextual duration-aware qubit mapping for various nisq devices," in *DAC*, 2020.

[22] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in *ASPLOS*, 2019, p. 1001–1014.

[23] H. Abraham, AduOffei *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

[24] Y. Shi, N. Leung *et al.*, "Optimized compilation of aggregated instructions for realistic quantum computers," p. 1031–1044, 2019.

[25] T. Itoko, R. Raymond *et al.*, "Quantum circuit compilers using gate commutation rules," in *ASP-DAC*, 2019, p. 191–196.

[26] T. Itoko, R. Raymond *et al.*, "Optimization of quantum circuit mapping using gate transformation and commutation," *Integration*, vol. 70, pp. 43–50, 2020.

[27] M.-S. Ying, "Commutativity between cnot and one-qubit gates," 2009, unpublished notes.

[28] R. Wille, D. Große *et al.*, "Revlib: An online resource for reversible functions and reversible circuits," in *ISMVL*, 2008, pp. 220–225.

[29] "Rigetti computing," https://www.rigetti.com/.

[30] M. R. Geller and Z. Zhou, "Efficient error models for fault-tolerant architectures and the pauli twirling approximation," *Phys. Rev. A*, vol. 88, p. 012314, Jul 2013.

[31] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, p. 180509, Nov 2017.

[32] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, p. 031027, Jul 2018.

[33] N. M. Linke, D. Maslov *et al.*, "Experimental comparison of two quantum computing architectures," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3305–3310, 2017.

[34] A. Shukla, M. Sisodia, and A. Pathak, "Complete characterization of the directly implementable quantum gates used in the ibm quantum processors," *Physics Letters A*, vol. 384, no. 18, p. 126387, 2020.

[35] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *MICRO*, 2019, p. 253–265.