QRCC: Evaluating Large Quantum Circuits on Small Quantum Computers through Integrated Qubit Reuse and Circuit Cutting

Aditya Pawar
adp110@pitt.edu
University of Pittsburgh
USA

Yingheng Li
yil392@pitt.edu
University of Pittsburgh
USA

Zewei Mo
zewei.mo@pitt.edu
University of Pittsburgh
USA

Yanan Guo
yag45@pitt.edu
University of Pittsburgh
USA

Xulong Tang
tax6@pitt.edu
University of Pittsburgh
USA

Youtao Zhang
zhangyt@cs.pitt.edu
University of Pittsburgh
USA

Jun Yang
juy9@pitt.edu
University of Pittsburgh
USA

Abstract

Quantum computing has recently emerged as a promising computing paradigm for many application domains. However, the size of quantum circuits that can be run with high fidelity is constrained by the limited quantity and quality of physical qubits. Recently proposed schemes, such as wire cutting and qubit reuse, mitigate the problem but produce sub-optimal results as they address the problem individually. In addition, gate cutting, an alternative circuit-cutting strategy that is suitable for circuits computing expectation values, has not been fully explored in the field.

In this paper, we propose QRCC, an integrated approach that exploits qubit reuse and circuit-cutting (including wire cutting and gate cutting) to run large circuits on small quantum computers. Circuit-cutting techniques introduce non-negligible post-processing overhead, which increases exponentially with the number of cuts. QRCC exploits qubit reuse to find better cutting solutions to minimize the cut numbers and thus the post-processing overhead. Our evaluation results show that on average we reduce the number of cuts by 29% and additional reduction when considering gate cuts.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0391-1/24/04.
https://doi.org/10.1145/3622781.3674179

ACM Reference Format:
Aditya Pawar, Yingheng Li, Zewei Mo, Yanan Guo, Xulong Tang, Youtao Zhang, and Jun Yang. 2024. QRCC: Evaluating Large Quantum Circuits on Small Quantum Computers through Integrated Qubit Reuse and Circuit Cutting. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3622781.3674179

1 Introduction

Quantum computing has recently emerged as a promising computing paradigm for many application domains, such as machine learning [7, 28, 47], chemistry simulation [3, 25, 40], and optimization [31, 34]. The problems from these domains scale quickly such that they require increasingly larger fault-tolerant quantum computers. Unfortunately, we are currently in the NISQ (noisy intermediate-scale quantum) era [15] where quantum devices suffer from various noises, e.g., short coherence time and crosstalk among qubits, and small device sizes, e.g., current quantum devices only have up to 100s of qubits.

It has become one of the major challenges to run large quantum circuits in the NISQ era. Large quantum computers, e.g., IBM 433 osprey, often have limited availability to the general public. In addition, not all qubits of large quantum computers exhibit high computational fidelity — some noisy qubits have to be *frozen*, i.e., not used, for some computation tasks [4]. Error mitigation schemes [12, 14, 39, 51] help to improve computation fidelity [15, 45, 47] but have limited effectiveness due to the limited availability of physical qubits on devices. As an alternative to physical quantum execution, software simulation offers a noise-free execution environment for quantum circuits. However, the simulation cost

increases exponentially with the number of qubits [50] and thus faces intrinsic drawbacks for scalability.

Recently proposed schemes, i.e., *wire cutting*, *qubit reuse*, and *gate cutting*, help to mitigate the challenge. The wire-cutting schemes [37, 41, 44] partition a large circuit into several smaller subcircuits, run subcircuits on quantum devices, and then reconstruct the output of the original circuit through classical post-processing. That is, they adopt a hybrid approach that combines physical quantum execution and classical software post-processing. Since the classical post-processing overhead increases exponentially with the number of wire cuts, minimizing the cut number is the major design goal. The qubit-reuse technique [16, 21, 35] exploits the hardware support for Mid-Circuit Measurement and Reset (MR) [23] such that the physical qubits that have finished all their operations can be redeployed as other logical qubits during circuit execution. Another circuit-cutting approach, gate cutting [30], was recently proposed for circuits that compute expectation values, e.g., Hamiltonian simulation algorithms, where minimization of expectation value is the main design goal. Gate cutting cuts a two-qubit gate into a linear sum of single-qubit gates and exploits classical post-processing to reconstruct the original result. Gate cutting has not been well-studied at the circuit level, i.e., deciding the best cut locations for a given large circuit.

Unfortunately, we observe that these schemes are currently applied individually and tend to produce sub-optimal results. Qubit reuse can reuse physical qubits only after their initially assigned operations finish. Its effectiveness diminishes as the circuits grow larger — only a few qubits can start their operations after some other qubits have finished. Wire cutting introduces one extra qubit (*initialization qubit* in [44]) after each cut, which may artificially increase the total number of physical qubits required for partitioning circuits. Gate cutting has not been well-studied at the circuit level. In addition, gate cutting, since its post-processing cannot reconstruct the distribution result, can only be applied to the quantum circuits that compute expectation values.

In this work, we present QRCC, a framework for evaluating large quantum circuits on small quantum devices through integrated Qubit-Reuse and circuit-cutting. QRCC is an end-to-end approach that models a large quantum circuit using ILP (integer linear programming), finds a good cutting solution using an ILP solver, maps the decision to subcircuits, runs the subcircuits on quantum devices, and reconstructs the original result through classical post-processing.

Compared with prior schemes, our key observation is that wire cuts in the circuit enlarge qubit-reuse opportunities, which in turn helps to eliminate unnecessary cuts in the circuit. By integrating qubit reuse and circuit-cutting, QRCC strives to find better cutting solutions with fewer numbers of cuts, reduced post-processing overhead, and improved per-circuit computation fidelity. When only the quantum circuit's expectation value is required, gate cutting can be applied to enlarge the cutting possibilities and thus further decrease the number of required cuts.

We further study in detail, in Section 6.6, the post-processing overhead with regards to i) the number of cuts, ii) the size of the quantum circuit, and iii) the reconstruction strategy for original output construction. We show that the number of cuts required to partition a circuit is the dominant factor contributing to the overhead. We further study the impact of i) circuit and device size, and ii) density of two-qubit gates when partitioning a quantum circuit on the number of cuts required. Our data shows the scalability of our circuit-cutting framework in the NISQ era.

We summarize our contributions as follows.

- We propose QRCC, a framework for evaluating large circuits on small quantum computers. To the best of our knowledge, QRCC is the first framework that (i) integrates wire and gate cuttings; and (ii) exploits qubit reuse to take advantage of the opportunities from circuit-cutting.
- We formulate the problem as a searching framework using ILP, which enables the searching for solutions under different optimization goals. The ILP formulation helps to achieve efficiency in an enlarged search space, and better scalability over the state-of-the-art [44].
- We evaluate QRCC using different benchmarks. Our results show that, on average, we reduce the number of cuts by 29% when considering wire cutting only and gain additional reduction when considering wire and gate cutting. We verify our approach using real device execution and post-processing.
- We provide a detailed analysis of the post-processing overhead of our framework. We highlight the key factors contributing to the complexity of circuit cutting with respect to the circuit and device size, reconstruction strategy, and cutting strategy.

2 Background

2.1 Quantum Circuits and their Outputs

A quantum program is represented as a quantum circuit consisting of qubits and quantum gates. Current quantum hardware supports single-qubit and two-qubit gates, which are also the gates considered in this paper. A quantum circuit, represented as a unitary matrix $\mathcal{U}$, takes an initial qubit state $|\psi\rangle$, usually $|0\rangle^{\otimes n}$, and evolves it to an output state $|\phi\rangle$.

$$\mathcal{U}|\psi\rangle = |\phi\rangle \tag{1}$$

Many quantum algorithms, e.g., Grover's algorithm for quantum search [19], compute probability vector $|\phi\rangle$, which indicates the probability distribution of measuring each of the possible 2^n states. Alternatively, other algorithms, such as Variational Quantum Algorithm (VQA) [32, 39], compute the expectation value of a Hamiltonian in the computational

measurement basis M.

$$\mathbf{E} = \langle\phi|M|\phi\rangle \tag{2}$$

2.2 Quantum Circuit Simulation

Quantum circuits can be simulated on classical computers using state-vector simulation. The classical simulation provides an ideal noise-free run of quantum circuits and accurately reproduces the output. However, there is an exponential cost of simulation, which restricts the simulation of larger quantum circuits. Wu *et al.* showed that simulating the 61-qubit Grover search algorithm needed Argonne's Theta supercomputer with 4,096 nodes and 768TB memory [50].

An alternative approach to simulation is to run the circuits on real quantum computers, using the shots-based model. That is, the quantum circuit is executed thousands of times, with each execution referred to as a shot, on the quantum hardware and the measurement of each qubit from each shot is summarized as the output probability-vector of the circuit. The drawbacks of this approach are: (i) many shots are required, even in an ideal noise-free setting, to reproduce the output probability vector of the original circuit accurately; (ii) given that today's quantum computers are noisy, the computation fidelity is often low for large circuit execution; (iii) the available quantum computers have limited numbers of qubits. The largest quantum computer from IBM has 433 physical qubits [18].

2.3 Circuit-Cutting

Circuit-cutting is a technique for obtaining the result of a large quantum circuit on small quantum devices. After cutting the large circuit into two or more smaller subcircuits using either *wire cutting* or *gate cutting*, we can execute subcircuits on small quantum devices, and generate the result of the original circuit from the classical post-processing of the results of subcircuits [37].

2.3.1 Wire Cutting (W-Cut) . Wire cutting (W-Cut) [37, 44] cuts the wire that connects two quantum gates, as shown in Figure 1(a). Here, U1 and U2 are two generic two-qubit gates. W-Cut cuts the original circuit into two independent subcircuits: $subcircuit_0$ and $subcircuit_1$, as shown in the figure. To obtain the output state $|\rho\rangle$ of the original circuit, CutQC [44] runs $subcircuit_0$ with measurements in four bases, and $subcircuit_1$ with four initializations; and then reconstructs the output of the original circuit using Equation (3).

$$\rho = \frac{A_1 + A_2 + A_3 + A_4}{2} \tag{3}$$

where

$$A_1 = Tr(\rho I)[|0\rangle\langle 0| + |1\rangle\langle 1|]$$
$$A_2 = Tr(\rho Z)[|0\rangle\langle 0| - |1\rangle\langle 1|]$$
$$A_3 = Tr(\rho X)[2|+\rangle\langle +| - |0\rangle\langle 0| - |1\rangle\langle 1|]$$
$$A_4 = Tr(\rho Y)[2|i\rangle\langle i| - |0\rangle\langle 0| - |1\rangle\langle 1|]$$

Here, Tr() is the trace operator indicating running $subcircuit_0$ physically on quantum devices and measuring the output in one of the Pauli basis bases (i.e., M ∈{I, X, Y, Z}). Measuring a qubit in either the I or Z basis gives the same circuit. $|x\rangle\langle x|$ is the density matrix indicating initializing $subcircuit_1$ in one of the eigen states (i.e., I ∈{$|0\rangle$, $|1\rangle$, $|+\rangle$, $|i\rangle$}). From Equation 3, W-Cut needs four pairs of Kronecker products between the subcircuit results to reconstruct the result of the original circuit. If it takes k (k>0) cuts to partition a large circuit into multiple independent subcircuits, the classical post-processing overhead of result reconstruction is $O(4^k)$.

Applying W-Cut at the circuit level is an optimization problem that finds the wires to be cut in a given large circuit such that the cutting has the smallest k and ensures the execution of each subcircuit on small quantum devices. CutQC formulates the problem as an MIP (mixed integer programming) model and exploits an MIP solver to search for the best solution.

2.3.2 Gate Cutting (G-Cut) . Gate cutting (G-Cut) cuts a two-qubit quantum gate, e.g., $U3$ in Figure 1(b), into a linear sum of single-qubit gates $U_{3.T}$ and $U_{3.B}$. According to the theory of gate cutting [30], if G-Cut cuts a two-qubit gate of the form $e^{i\theta A_1 \otimes A_2}$ (e.g., CNOT, CZ, and ZZ gates) where A_1^2 = A_2^2 = I, the expectation value **E** of the original circuit can be reproduced based on the output state $|\phi_i\rangle$ of subcircuits, using Equation (4). G-Cut differs from W-Cut in that G-Cut cannot reproduce the original circuit state-vector, but rather only the expectation value.

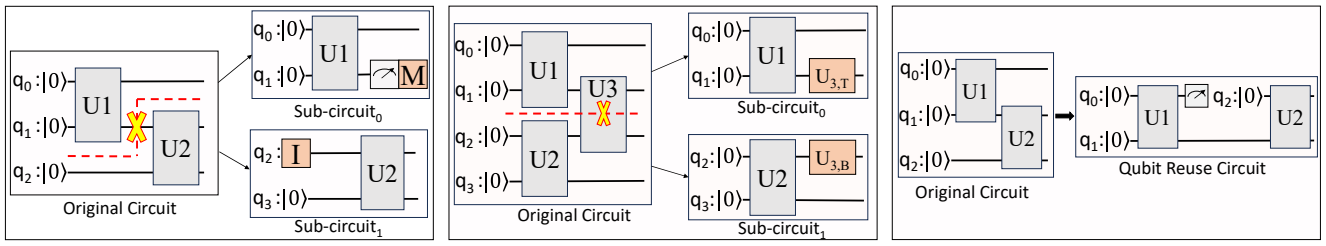
$$\mathbf{E}[\phi] = \sum_{i=1}^{6} c_i E[\phi_i] \tag{4}$$

where,

$$\phi_1 = \mathbf{S}(I \otimes I) \qquad c_1 = cos^2(\theta)$$
$$\phi_2 = \mathbf{S}(A_1 \otimes A_2) \qquad c_2 = sin^2(\theta)$$
$$\phi_3 = \beta\mathbf{M}_{A_1,\beta} \otimes \mathbf{S}(e^{i\pi A_2/4}) \qquad c_3 = cos(\theta)sin(\theta)$$
$$\phi_4 = \beta\mathbf{M}_{A_1,\beta} \otimes \mathbf{S}(e^{-i\pi A_2/4}) \qquad c_4 = -cos(\theta)sin(\theta)$$
$$\phi_5 = \mathbf{S}(e^{i\pi A_1/4}) \otimes \beta\mathbf{M}_{A_2,\beta} \qquad c_5 = cos(\theta)sin(\theta)$$
$$\phi_6 = \mathbf{S}(e^{-i\pi A_1/4}) \otimes \beta\mathbf{M}_{A_2,\beta} \qquad c_6 = -cos(\theta)sin(\theta)$$

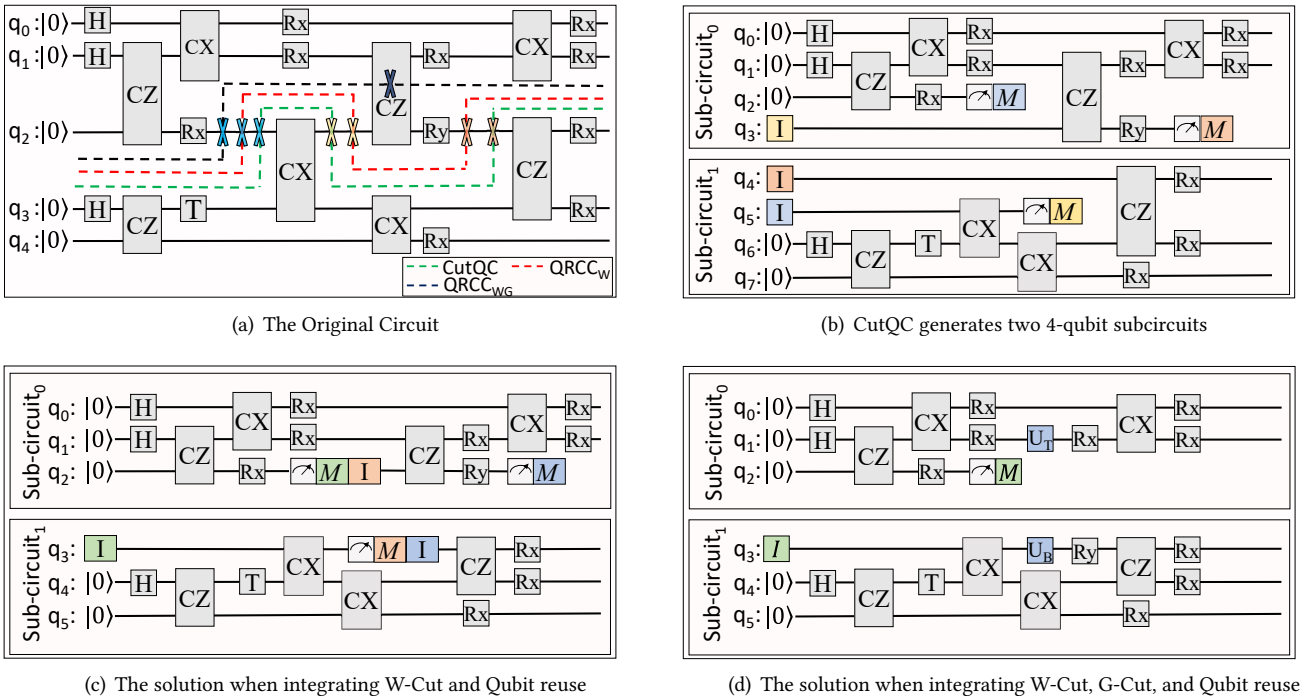
G-Cut produces six subcircuit instances, i.e., ϕ_1 to ϕ_6. Each ϕ_i is an independent instance, from which during its execution, we remove the two-qubit gate that has been cut from the original circuit, and replace it with single-qubit gates of the respective instance. The M_{A_i} term is single qubit measurement operations, with β representing the outcome of the measurement, $\beta \in \{1,-1\}$. More details can be found in [30].

G-Cut has not been well-studied at the circuit level. Given a large circuit, it remains an open problem to determine the subset of two-qubit gates to be cut to achieve our design goal, in particular, together with W-Cut and qubit reuse.



(a) Wire Cutting (b) Gate Cutting (c) Qubit Reuse

Figure 1. Circuit-cutting and qubit reuse. (a) An example of wire cutting is where qubit q_1 has been cut. It leaves measurement (M) and Initialization (I) operations in two subcircuits, respectively. (b) An example of Gate cut is where gate U3, acting on qubits q_1 and q_2 has been cut. It leaves two single-qubit gates in two subcircuits, respectively. (c) An example of qubit reuse. Once the U1 gate has finished executing, qubit q_0 can be measured and reused for logical qubit q_2.



(a) The Original Circuit

(b) CutQC generates two 4-qubit subcircuits

(c) The solution when integrating W-Cut and Qubit reuse

(d) The solution when integrating W-Cut, G-Cut, and Qubit reuse

Figure 2. The integration of W-Cut, G-Cut, and qubit reuse helps to find better cutting solutions. (a) Original Circuit, showing three different cutting solutions. (b) The solution generated by CutQC. (c) The solution when integrating W-Cut and qubit reuse. (d) The solution when integrating W-Cut, G-Cut, and qubit reuse. (**M**/**I** indicate measurement and initialization, respectively, due to W-Cut; and $\mathbf{U}_T$/$\mathbf{U}_B$ indicate the top/bottom single-qubit gates after applying G-Cut to a two-qubit gate **U**).

2.4 Measure and Reset Functionality

IBM recently introduced mid-circuit measurement operation and mid-circuit reset operation [23] to support dynamic circuits for quantum error correction[13, 24, 48] and runtime program verification[22, 27]. As shown in Figure 1(c), once qubit q_0 finishes its operation with gate $U1$, we measure this physical qubit and re-initialize another logical qubit q_2 in the $|0\rangle$ state, and assign qubit q_2 to the same physical qubit on the quantum device. This is referred to as *qubit-reuse* in [21]. In the figure, qubit-reuse enables the execution of the original three-qubit circuit on a two-qubit quantum device.

CaQR [21] proposes a compiler-assisted tool that automatically identifies qubit-reuse opportunities in a given circuit, reduces the total number of required physical qubits, and achieves better performance and computation fidelity. The effectiveness of qubit reuse diminishes as the circuit becomes bigger — only a few qubits can delay their operations enough to start after some other qubits have finished.

3 Motivation

In this section, we use an example (in Figure 2) to illustrate the effectiveness when integrating qubit reuse, W-Cut, and G-Cut. Our problem is to run a 5-qubit circuit on small quantum devices, e.g., 4-qubit or 3-qubit quantum devices.

When adopting CutQC [44], the original circuit is split into two subcircuits using three cuts, as shown in Figure 2(b). Each subcircuit has four qubits. Three extra qubits (i.e., the initialization qubits) are introduced, e.g., the first wire cut on $q2$ generates an extra qubit $q5$, which is now the second qubit in $subcircuit_1$. It leaves a measurement in $subcircuit_0$. We use three color pairs to indicate the introduced measurement operation and its matching initialization bit.

For this circuit, CutQC can't find a solution that splits the original circuit into two 3-qubit subcircuits. It is also impossible to apply qubit reuse [21] directly on the original circuit to reduce the number of required qubits.

3.1 W-Cut and qubit-reuse

Figure 2(c) shows the cutting solution when we integrate W-Cut and qubit reuse. The integrated scheme, even though choosing the same cutting positions as those in CutQC, generates two 3-qubit subcircuits.

The improvement comes from the reuse opportunities exposed from wire cutting. For example, for $subcircuit_0$, qubit $q2$ becomes idle after the first cut. It can be reused by the initialization qubit. By exploiting the qubit reuse opportunities, each subcircuit requires one fewer qubit so that both can run on three-qubit quantum devices.

W-cut partitions the operations on the cut qubit, such that it introduces new qubit reuse opportunities into the circuit, which previously did not exist.

3.2 W-Cut and G-Cut

Figure 2(d) shows the cutting result when we integrate W-Cut and G-Cut. The integrated scheme can cut the original circuit into two subcircuits in two cuts — one wire cut and one gate cut. The two-qubit gate CZ is cut into two single-qubit gate instances U_T and U_B in different subcircuits.

In this example, G-Cut is enabled only if the circuit computes the expectation value. The classical post-processing overhead also impacts the solution selection, in particular, the cost from G-Cut is slightly higher than that from W-Cut, i.e., 6^k vs 4^k, where k is the number of cuts. Therefore, we need to consider this difference when choosing a cutting solution. Given a solution S($k1$, $k2$) where $k1$ and $k2$ are the numbers of gate cuts and wire cuts, respectively, its classical post-processing overhead is O($4^{k1}6^{k2}$). It is better to choose S(1,1) over S(2,1) for the example in the figure. In another situation, it would be worse to choose S(0,4) over S(5,0).

4 The QRCC Framework

In this section, we elaborate QRCC, an end-to-end framework, for running large quantum circuits on small quantum devices. Given a large circuit, QRCC converts it to a QR (qubit-reuse)-aware DAG, formulates and solves an ILP model, maps the cutting solutions to subcircuits, runs the subcircuits, and reconstructs the original result.

4.1 The QR-aware DAG Representation

Given a N-qubit input quantum circuit that is to be cut for an D-bit quantum device (N>D>0), we first convert the circuit to a QR-aware DAG by adding dummy *Identity* gates such that, each qubit goes through the same number of quantum operations. After adding the identity gates, all qubits are *aligned* so that we define a **quantum layer** m as the set of m-th gate for each qubit.

In Figure 3, V1, V6, and V7 are two-qubits gates, S2 and S4 are single-qubit gates, and F3, F5, F8, and F9 are inserted *Identity* gates. Gates S4, F5, and V6 belong to layer M. We place a yellow × on each wire before a gate to indicate a potential cutting location.

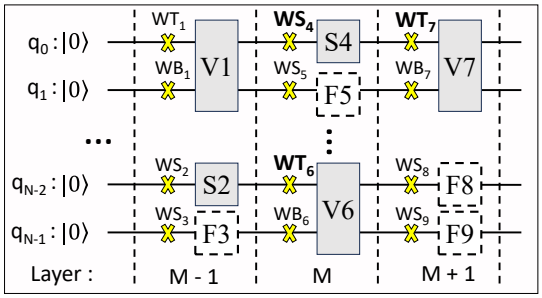


Figure 3. A QR-aware DAG representation of the quantum circuit. (Dashed boxes indicated identity gates. Each wire segment may potentially be W-Cut, and each two-qubit gate may be potentially G-Cut.)

Compared to the DAG for traditional wire cutting [44] that lists two-qubit gates only, our QR-aware DAG explicitly lists all single-qubit gates and differentiates the cuts on the wires connecting different single qubits. For example, for the two cuts: (i) WS_4 is the one on the wire connecting the S4 gate and (ii) WT_7 is the one on the top wire connecting the V7 gate, the traditional DAG treats WS_4 and WT_7 as the same cut, as cutting at either location does not affect the number of required qubits in each subcircuit. However, when we consider qubit reuse, if there is another cut WT_6 (the top wire connecting V6), we may prefer to choose WS_4 if gate V1 and V6 are in the same subcircuit. This is because qubit q_0 can be reused for qubit q_{N-2} while cutting at WS_4 disables this reuse. For discussion purposes, we assume the measurement and initialization operations take no depth and qubit-reuse does not increase circuit depth. Section 4.2.6 discusses how to handle the depths of these operations.

4.2 The ILP Model

4.2.1 The Meta Parameters. We formulate the problem as an ILP model. We first list the meta parameters, i.e., the constants that we define for the problem and/or we collect from preprocessing the input circuit.

- N and D: They are the number of qubits in the input quantum circuit and the number of available physical qubits of the quantum device, respectively. We have $N>D>0$.
- G_{max} and W_{max}: The maximum number of allowed gate cuts and wire cuts, respectively.
- C_{min} and C_{max}: The minimum and maximum numbers of subcircuits to be cut, respectively. Note, that our ILP solver often reports a cutting solution that has fewer than C_{max} subcircuits. This is because our model focuses on reducing the classical post-processing overhead, which does not relate to the number of subcircuits. As we elaborate next, the cost relates to a combination of wire cuts and gate cuts, as the two types of cuts have slightly different classical post-processing overhead.
  If C_{min}=C_{max}, the solution that we find has the specified number of subcircuits.
- δ: The relative weight for adjusting the optimization goal between classical post-processing overhead and computation fidelity. We will elaborate in Section 4.2.5.
- *G-Cut-enabled*: This is a binary parameter indicating if gate cutting should be enabled. As we discussed, after G-Cut, we can only reconstruct the expected value of the circuit. If the original circuit is to compute the probability vector, we disable the gate cutting in the model.

4.2.2 ILP Variables. When preprocessing the original circuit, we differentiate three types of gates, i.e., two-qubit gates, single-qubit gates in the original circuit, and identity gates that we inserted. We number all gates and define a binary variable for each of the gates as follows.

$$\begin{aligned} V_{x,c} &= \begin{cases} 1 & \text{if two-qubit gate } x \text{ is in subcircuit } c \\ 0 & \text{Otherwise} \end{cases} \\ S_{x,c} &= \begin{cases} 1 & \text{if single-qubit gate } x \text{ is in subcircuit } c \\ 0 & \text{Otherwise} \end{cases} \\ F_{x,c} &= \begin{cases} 1 & \text{if identity gate } x \text{ is in subcircuit } c \\ 0 & \text{Otherwise} \end{cases} \end{aligned} \tag{5}$$

For single-qubit and identity gates, we can only perform W-Cut. We set the cutting point on the wire before each gate. We do not Cut any gate on the first layer.

$$WS_x = \begin{cases} 1 & \text{if single-qubit/identity gate } x \text{ is W-Cut,} \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

For two-qubit gates, we can perform both W-Cut and G-Cut. For W-Cut, we can cut either of its input wires. We define the following variables.

$$\begin{aligned} U_x &= \begin{cases} 1 & \text{if two-qubit gate x is neither W-Cut} \\ & \text{nor G-Cut} \\ 0 & \text{Otherwise} \end{cases} \\ WT_x &= \begin{cases} 1 & \text{if top wire to two-qubit gate } x \\ & \text{is W-cut} \\ 0 & \text{Otherwise} \end{cases} \\ WB_x &= \begin{cases} 1 & \text{if bottom wire to two-qubit gate } x \\ & \text{is W-cut} \\ 0 & \text{Otherwise} \end{cases} \\ G_x &= \begin{cases} 1 & \text{if two-qubit gate x is G-Cut} \\ 0 & \text{Otherwise} \end{cases} \end{aligned} \tag{7}$$

When G-Cutting a two-qubit gate x, we get two single-qubit gates, referred to as $x.top$ and $x.bottom$. These two gates appear only if G_x=1. Similar to those in definition (6), we define variables to determine if they are in some subcircuits.

$$GT_{x,c} = \begin{cases} 1 & \text{if for two-qubit gate } x, \text{ we have} \\ & G_x\text{=1 and } x.top \text{ is in subcircuit c} \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

$$GB_{x,c} = \begin{cases} 1 & \text{if for two-qubit gate } x, \text{ we have} \\ & G_x\text{=1 and } x.bottom \text{ is in subcircuit c} \\ 0 & \text{Otherwise} \end{cases}$$

4.2.3 The General ILP Constraints. We next list the general constraints in our model. These constraints are the same regardless of the input circuit and user parameters.

Whether a single-qubit or identity gate x is cut is determined by its WS_x variable. However, a two-qubit gate may be W-Cut, G-Cut, or not cut. That is, we cannot W-Cut and G-Cut the gate at the same time. Therefore, we have the following constraints for each two-qubit gate x.

$$\begin{aligned} U_x + WT_x + WB_x + G_x &\geq 1 \\ U_x + WT_x &\leq 1 \\ U_x + WB_x &\leq 1 \\ U_x + G_x &\leq 1 \end{aligned} \tag{9}$$

Each gate x must belong to one and only one subcircuit, unless it is a two-qubit gate and has been G-Cut. If a two-qubit gate x is G-Cut, its two-qubit gate form *conceptually* disappears such that the two single-qubit gates, i.e., $x.top$ and $x.bottom$, emerge in the circuit. In this case, the newly generated single-qubit gates, i.e., $x.top$ and $x.bottom$, must belong to one and only one subcircuit. These two single-qubit gates cannot belong to the same subcircuit.

We use this technique to linearize the gate cut constraints. This technique is also used in Section 4.2.6.

$$\begin{aligned} \text{for single-qubit gate } x,\ \sum_{c\in C} S_{x,c} &= 1 \\ \text{for two-qubit gate } x,\ \sum_{c\in C} V_{x,c} + G_x &= 1 \\ \sum_{c\in C} GT_{x,c} &= G_x \\ \sum_{c\in C} GB_{x,c} &= G_x \\ \text{for } \forall \text{ subcircuit } c \in C,\ GT_{x,c} + GB_{x,c} &\le 1 \end{aligned} \tag{10}$$

After cutting, each subcircuit should not have more than D qubits, i.e., the device size constraint. Therefore, for all identity gates x, single-qubit gates s, and two-qubit gates t, respectively at each layer l,

$$Q_{c,l} = \sum_x F_{x,c} + \sum_s S_{s,c} + \sum_t (2V_{t,c} + GT_{t,c} + GB_{t,c}) \le D \tag{11}$$

where $Q_{c,l}$ is the number of qubits used in subcircuit c at layer l. By adopting the layer-based cutting approach, our model allows us to find better qubit reuse opportunities such that a wire cut at an early layer can be reused by a different qubit at a later layer.

We also restrict the number of gate cuts and wire cuts.

$$\begin{aligned} \sum_x G_x &\le G_{max} \\ \sum_x (WS_x + WT_x + WB_x) &\le W_{max} \end{aligned} \tag{12}$$

4.2.4 The Circuit-dependent Constraints.

In addition to the general constraints, we have circuit-dependent constraints. These constraints specify the relationship between two neighboring gates.

If two neighboring gates are two two-qubit gates, we may have two cases: (a) the bottom output of the upstream gate connects to the top input of the downstream gate, e.g, the U1-U3 connection in Figure 1(b); or (b) the top output of the upstream gate connects to the bottom input of the downstream gate, e.g, the U2-U3 connection. We specify their constraints as follows.

$$\begin{aligned} 2 \times WT_{U3} &= \textstyle\sum_{c\in C} (|V_{U1,c} - V_{U3,c} + GB_{U1,c} - GT_{U3,c}|) \\ 2 \times WB_{U3} &= \textstyle\sum_{c\in C} (|V_{U2,c} - V_{U3,c} + GT_{U2,c} - GB_{U3,c}|) \end{aligned} \tag{13}$$

If two neighboring gates are one upstream single-qubit gate and one downstream two-qubit gate, for example, if we replace U1 and U3 with single-qubit gates, the constraints are

$$\begin{aligned} 2 \times WT_{U3} &= \textstyle\sum_{c\in C} (|S_{U1,c} - V_{U3,c} - GT_{U3,c}|) \\ 2 \times WB_{U3} &= \textstyle\sum_{c\in C} (|S_{U2,c} - V_{U3,c} - GB_{U3,c}|) \end{aligned} \tag{14}$$

Similar constraints are specified for other circuit connections. They follow the same rules of gate and wire cuts.

4.2.5 The Objective Function.

Our ILP model consists of two optimization goals.

- Our main optimization goal is to reduce the number of cuts to minimize the classical post-processing overhead. Since the overhead of a cutting solution (k,m), i.e., with k wire cuts and m gate cuts, is O($4^k 6^m$), a naive integration of this overhead in the objective function would lead to a non-linear component, which can greatly slow down the solver. Instead, we linearize the cost as $\alpha k + \beta m$ such that if the exponential cost of (k_1,m_1) is smaller than that of (k_2,m_2), our linear cost has the same relative relationship. In this work, we choose α=3.25 and β=4.2 as they satisfy the requirement for the number of cuts smaller than 240 (120 W-cut and 120 G-cut).
  Therefore, the classical post-processing overhead is
  $$PPCost = \alpha \times \sum_x (WS_x + WT_x + WB_x) + \beta \times \sum_x G_x \tag{15}$$
- The other optimization goal of our model is to improve the computational fidelity. Studies have shown that the computation error of a quantum circuit depends on the number of operations, in particular, two-qubit quantum operations [33, 36, 52]. This is because the error rate of two-qubit gates is orders of magnitude higher than that of single-qubit gates. To improve the computational fidelity after circuit cutting, we strive to balance the number of two-qubit gates across different subcircuits.
  We define a new variable TE to track the maximal number of two-qubit gates in a subcircuit. Minimizing TE would help to improve the overall computation fidelity. We add one linear constraint for each subcircuit c as follows. This helps to find the subcircuit that has the maximal number of two-qubit gates.
  $$TE \ge \sum_x V_{x,c} \tag{16}$$
  We further define the two-qubit gate-related error as
  $$CError = f(TE) \tag{17}$$
  Next, we illustrate how to choose a linear function so that $PPCost$ and $CError$ have similar value ranges. We use an example to explain how to choose the function. We first run the model considering $PPCost$ only such that we may find a cutting solution (4, 6) with the PPCost value being 3.25×4+4.2×6=38, the number of subcircuits being 5, and the current TE being 40. Assuming we can achieve a perfect balancing of the subcircuits with a maximal increase of 4 additional cuts and get a solution (6,7) with $PPcost$ being 53. The PPCost range is [38,53]. The TE range is now [20,40]. We choose linear function f(TE)= TE×0.75+23. Note, that a further refined linear function can be derived for a given circuit.

Oftentimes, balancing the number of two-qubit gates across subcircuits may result in a cutting solution with more cuts

and thus higher classical post-processing overhead, or even no solution. Therefore, we introduce another meta parameter δ to adjust the optimization goal between *PPCost* and *CError*. The δ value can be integrated into deciding the linear function in *CError*.

To summarize, our objective function is as follows.

$$\textbf{Min}[\delta \times PPCost + (1-\delta) \times CError] \tag{18}$$

4.2.6 Discussion. We make two simplifications for clarity purposes in the preceding discussion of the model. (1) We assume adding *Identity* gates to ensure all layers have N gates. This may introduce a large number of identity gates and their corresponding constraints, which slows down the solver. In our implementation, for a long wire that connects two gates far away from each other, we selectively add two or three identity gates at the beginning, middle, and end of the wire. (2) We assume the measurement and initialization operations take no depth. However, when considering their depths, we introduce a trailing measurement gate after the cut point, and a leading initialization gate before the cut point. These two gates emerge in the circuits only if the corresponding wire is cut. We use the same technique as that for gate cutting, i.e., only if a two-qubit gate is cut, its corresponding single-qubit gates emerge in the circuit.

4.3 Output Reconstruction

Reconstruction after W-Cut. If the original quantum circuit computes the probability distribution vector, we can only adopt wire-cut. The probability vector results from the subcircuit runs can be recombined using Equation (3). The classical post-processing process follows the techniques as elaborated in CutQC [44].

Reconstruction after W-Cut and G-Cut. The original quantum circuit, if computing the expectation value, can be cut by both W-Cut and G-Cut. The reconstruction overhead of expectation values is lower than that of probability vectors as the expectation value is a floating point value, while a probability vector consists of multiple floating point values.

To reconstruct the expectation value of the original circuit, we sort the subcircuits according to the numbers of their qubits and then start from the smallest subcircuit. For each subcircuit, we first reconstruct at the wire-cutting positions and then at the gate-cutting positions. When handling the W-Cut wires, we reconstruct the expectation values directly, instead of the probability vectors. The Equation (19) in Section 2 is applicable for both probability vectors and expectation values [37]. For the latter, it can be adapted as follows.

$$\mathbf{E}[\rho] = \frac{A_1 + A_2 + A_3 + A_4}{2} \tag{19}$$

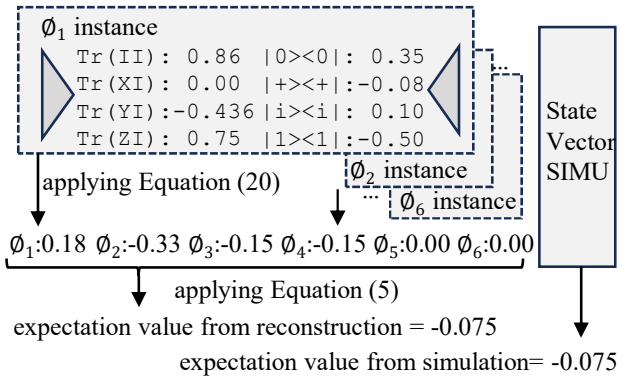


Figure 4. The reconstruction of the expectation value after W-Cut and G-Cut.

where

$$A_1 = \mathbf{E}[Tr(\rho I)](\mathbf{E}[|0\rangle\langle 0|] + \mathbf{E}[|1\rangle\langle 1|])$$
$$A_2 = \mathbf{E}[Tr(\rho Z)](\mathbf{E}[|0\rangle\langle 0|] - \mathbf{E}[|1\rangle\langle 1|])$$
$$A_3 = \mathbf{E}[Tr(\rho X)](2\mathbf{E}[|+\rangle\langle +|] - \mathbf{E}[|0\rangle\langle 0|] - \mathbf{E}[|1\rangle\langle 1|])$$
$$A_4 = \mathbf{E}[Tr(\rho Y)](2\mathbf{E}[|i\rangle\langle i|] - \mathbf{E}[|0\rangle\langle 0|] - \mathbf{E}[|1\rangle\langle 1|])$$

After the reconstruction from W-Cut, we adopt Equation (4) to handle G-Cut for reconstructing the expectation value of the original circuit.

An example. We next illustrate the reconstruction process for the example shown in Figure 2(d), which consists of both W-Cut and G-Cut. For Figure 2(c) that contains only W-Cut, the reconstruction process is the same as that in CutQC[44].

In Figure 2(d), the original quantum circuit was cut into two subcircuits with one wire cut and one gate cut on a CZ gate. The CZ gate has the form

$$CZ = e^{\frac{i\pi I\otimes Z}{4}} e^{\frac{i\pi Z\otimes I}{4}} e^{\frac{i\pi Z\otimes Z}{4}} \tag{20}$$

Each exponential term in this form can be decomposed as shown in Equation (4). We can then combine and simplify all three terms to the following six instances [30]. These instances are independent of each other.

$$\phi_1 = \mathbf{S}(rz(\frac{-\pi}{2}) \otimes rz(\frac{-\pi}{2})) \qquad c_1 = \frac{1}{2}$$
$$\phi_2 = \mathbf{S}(rz(\frac{\pi}{2}) \otimes rz(\frac{\pi}{2})) \qquad c_2 = \frac{1}{2}$$
$$\phi_3 = \beta\mathbf{M}_{Z,\beta} \otimes \mathbf{S}(e^{i\pi Z/2}) \qquad c_3 = \frac{-1}{2}$$
$$\phi_4 = \beta\mathbf{M}_{Z,\beta} \otimes \mathbf{S}(I) \qquad c_4 = \frac{1}{2}$$
$$\phi_5 = \mathbf{S}(e^{i\pi Z/2}) \otimes \beta\mathbf{M}_{Z,\beta} \qquad c_5 = \frac{1}{2}$$
$$\phi_6 = \mathbf{S}(I) \otimes \beta\mathbf{M}_{Z,\beta} \qquad c_6 = \frac{-1}{2}$$

For example, during ϕ_1 instance's execution, we replace the two-qubit CZ gate with two single-qubit $rz(\frac{-\pi}{2})$ gates.

We then reconstruct the expectation value of this instance, for the wire cut (whose reconstruction follows Equation (20)). Figure 4 shows the reconstructed expectation value of each ϕ_i ($1 \leq i \leq 6$) using Equation (20) and then the expectation value of the original circuit using Equation (4). For verification purposes, the expectation value of the original circuit is also computed through state vector simulation, which shows the same result.

5 Experimental Methodology

We evaluate the effectiveness of QRCC using different benchmarks and compare the results with those from CutQC [44], the state-of-the-art wire-cutting scheme. While both were implemented using the Gurobi optimizer[20], QRCC builds an ILP (integer linear programming) model while CutQC builds an MIP (mixed integer programming) model. In the experiments, we set the maximal number of cuts to 100. The imbalance threshold is set as 500 gates for CutQC. In the experiments, we choose two δ values for case study purposes and present a comprehensive study of the meta-parameter in Section 6.4.

- **QRCC-C:** we choose δ=1, i.e., we focus on post-processing overhead only. The subcircuits, due to their smaller sizes, generally have better computational fidelity than that of the original circuit. However, some subcircuits may have significantly better computational fidelity than others if they contain fewer two-qubit gates.
- **QRCC-B:** we choose δ=3/4 such that in addition to the main design goal of reducing the post-processing overhead, we also balance the two-qubit gates for computational fidelity improvement.

We also run experiments on the IBM Lagos quantum computer through the IBM cloud service to verify our approach.

5.1 Benchmarks

We test our scheme using two groups of benchmarks: one computes the probability distribution while the other computes the expectation value. We generate multiple quantum circuits for each benchmark. We use a three-letter abbreviation to indicate each benchmark, the abbreviation is in the parameters as we describe each benchmark next.

The following four benchmarks compute the probability distribution and thus can only be cut using W-Cut.

- **QFT (QFT)**: Quantum Fourier Transform [10] is an important building block in many quantum algorithms, including Shor's factoring algorithm.
- **AQFT (AQFT)**: Approximate Quantum Fourier Transform is an approximation [6] of the QFT sub-routine, which tends to produce better results on NISQ devices.
- **Supremacy (SPM)**: This is a type of random circuit that was used by Google to demonstrate quantum supremacy [8].
- **Adder (ADD)**: This is a linear Ripple Carry Adder [11], which reduces the number of required ancilla qubits to 1.

To evaluate the effectiveness of G-Cut together with W-Cut, we choose the following five variational quantum algorithms that compute expectation values.

- **m-Regular (REG):** The graph in REG is a regular graph in which each node has m edges [42]. By default, m=5.
- **Erdos-Renyi (ERD):** The graph in ERD is a random graph in which we exploit a probability p in creating edges across different nodes in the graph [17]. By default, p=0.1.
- **Barabasi-Albert (BAR):** The graph for this benchmark is also a random graph. Each node in the graph has m edges that connect preferentially to nodes with high degrees [5]. by default, m=3.
- **Hamiltonian Simulation**: For the 2D square lattice Hamiltonian simulation [26], we choose three variations: 2D Traverse Field Isling (IS), XY(XY), and Heisenberg(HS) Hamiltonian. For each variation `M`, we use `M` and `M-n` to indicate the interactions for the nearest neighbor and the next nearest neighbor, respectively.
- **Variational Quantum Eigensolver (VQE)**: We simulate the Hydrogen chain VQE algorithm, using a linear two-local ansatz [46].

6 Experimental Results

6.1 Wire Cutting Evaluation

Table 1 compares the W-Cut only results when adopting three different cutting schemes, i.e., CutQC, QRCC-C, and QRCC-B, on benchmarks that compute probability vectors. We report the number of subcircuits (#SC), the required number of W-Cuts (#cuts), and the number of two-qubit gates in the largest subcircuit (#MS). When CutQC cannot find a solution, we report *no-solution* in the table.

Given two cutting solutions `C1` and `C2`, if `C1` cuts the original circuit to fewer subcircuits than `C2` does, `C1`'s subcircuits tend to have more two-qubit gates such that `C1` tends to have larger #MS values and thus worse computational fidelity. For a fair comparison, if QRCC cuts the original circuit to fewer subcircuits with fewer numbers of cuts than CutQC does, we also report the solutions from QRCC that, with slightly more cuts, cuts the original circuits into the same numbers of subcircuits as CutQC does. For example, for QFT(N=30, D=27), CutQC cuts the original circuits to four subcircuits while, by default, QRCC-B cuts into two subcircuits, resulting in larger #MS and worse computational fidelity than that of CutQC, i.e., 351 in CutQC vs 426 in QRCC-B. However, if allowing four subcircuits, QRCC-B achieves better-balanced subcircuits, i.e., 351 in CutQC vs 146 in QRCC-B.

From the table, our scheme significantly reduces the number of cuts — on average, QRCC-C and QRCC-B achieve 29% and 24% reductions over CutQC, respectively. The test cases from QFT have the most complicated circuits, i.e., more two-qubit gates that exhibit all-to-all qubit connections. For these test cases, CutQC may not find a solution if the device size D is small; and QRCC achieves the largest improvements. For

Table 1. Comparing W-Cut results using QRCC and CutQC. (D and N are meta parameters in Section 4.2.1; #SC: the number of subcircuits after cutting; #Cuts: the number of wire cuts; #MS: the maximal number of two-qubit gates in the subcircuits)

Benchmark			CutQC			QRCC-C			QRCC-B		
	N	D	#SC	#cuts	#MS	#SC	#cuts	#MS	#SC	#cuts	#MS
QFT	15	7	No Solution			3	20	69	3	20	68
	15	9	9	44	27	2	12	81	2	12	75
	30	16	No Solution			2	28	330	2	28	318
	30	20	No Solution			2	20	380	2	20	335
	30	24	4	52	276	2	12	414	2	12	399
						4	14	413	4	32	145
	30	27	3	32	351	2	6	429	2	6	426
						3	7	428	3	30	146
SPM	15	7	3	6	8	3	5	9	3	6	8
	20	7	5	11	8	4	9	13	4	9	9
	30	16	3	8	22	2	6	25	2	6	25
	42	16	4	13	21	3	12	26	3	12	24
ADD	16	7	4	6	35	3	4	51	3	4	51
	22	7	5	8	34	4	6	51	4	6	51
	30	16	2	2	120	2	2	120	2	2	120
	40	16	3	4	119	3	4	120	3	4	119
AQFT	15	7	4	10	18	4	10	18	4	10	16
	20	7	7	22	20	6	22	31	6	22	19
	30	16	3	8	65	3	8	65	3	8	65
	40	16	4	16	74	4	16	75	5	16	71

example, QRCC reduces the #cuts from 32 to 6 when N=30 and D=27, exhibiting 81% improvement. As a comparison, for the AQFT benchmark that approximates QFT with all-to-all connections removed, it is easier to find a cutting solution, resulting in negligible improvements over CutQC.

6.2 Wire- and Gate- Cutting Evaluation

Table 2 compares the cutting solutions when applying cutting on the benchmarks that compute expectation values. We compare two choices for our scheme: one is to choose W-Cut only while the other allows both W-Cut and G-Cut.

For comparison purposes, for a solution (k_1, k_2), where k_1 and k_2 are the W-Cut and G-Cut numbers, respectively, its overhead $4^{k_1}6^{k_2}$ is converted to 4^{k_3} with k_3 being the effective W-Cut number reported in the table. On average, QRCC (W-Cut only) and QRCC (both) achieve 41% and 44% reductions in the number of cuts. Exploiting G-Cut further reduces post-processing overhead. For example, for ERD-50, QRCC (both) has an #EffCuts of 22.46. While being a small reduction over 24 from QRCC(W-Cut only), it corresponds to an 8.45× reduction in post-processing overhead.

6.3 Real Machine Evaluation

The cutting solutions that adopt either W-cut or G-cut have already been independently verified in [44] and [30], respectively. We next verify the solution by adopting both and highlight our strategy for efficient post-processing.

We verify our approach by testing benchmark REG(m=2) on an IBM 7-qubit LAGOS quantum computer. The computer has 1.7 physical connections per qubit. When we ran the experiments, it had median error rates of $8.25e^{-3}$ for CNOT gates and $2.6e^{-4}$ for single-qubit, $\sqrt{x}$ gates, respectively.

We choose N=7 and D=4, i.e., the original quantum circuit has seven qubits, and QRCC partitions it into smaller subcircuits so that each subcircuit can run on a 4-qubit quantum computer. Table 3 compares four execution modes.

- *State Vector Simulation*: The result from the state vector simulation computes the ground truth for the comparison of the results from different schemes.
- *Shot-based Simulation:* In this mode, we run a shot-based state-vector simulation, which uses the state-vector probability to introduce a random bit-string output every shot for simulating an ideal device. We report the average from 10 circuit runs with each run having 16,384 shots.
- *Device Execution (7-qubit)*: We ran the 7-qubit original circuit on the real quantum computer with 16,384 shots for each run. We run 10 times and report the average.
- *QRCC*: QRCC partitions the original circuit into two subcircuits with one gate cut and one wire cut. The subcircuits are run with different measurement and initialization instances, resulting in a total of 42 instances. Each instance runs just one time (with 16,384 shots) using four physical qubits. The results from subcircuits are then combined to compute the result for the original circuit.

From the table, QRCC achieves better accuracy than that of 7-qubit device execution. This is because (1) The original circuit has 16 two-qubit CNOT gates (and 9 of them were introduced from SWAP operation) while each of our subcircuits contains 3 CNOT gates. This results in better computation fidelity for the subcircuit execution. (2) the subcircuits have fewer qubits and short execution depths. Due

Table 2. Comparison of W-Cut and W-Cut+G-Cut schemes (#EffCuts is the effective wire-cuts for comparison) for benchmarks computing expectation values.

Benchmark			CutQC			QRCC-C (W-Cut Only)			QRCC-C (W-Cut and G-Cut)				
	N	D	#SC	#Cuts	#MS	#SC	#Cuts	#MS	#SC	#W-cuts	#G-cuts	#EffCuts	#MS
REG	40	27	3	21	49	2	17	51	2	15	1	16.29	55
	50	27	4	38	43	2	24	63	2	22	1	23.29	62
ERD	40	27	3	31	67	2	23	109	2	21	1	22.29	109
	50	27	5	39	41	2	24	69	2	17	5	23.46	65
BAR	40	27	3	17	71	2	15	55	2	13	1	14.29	56
	50	27	3	28	62	2	24	71	2	20	2	22.46	71
IS	36	27	3	12	38	2	10	54	2	10	0	10	55
	49	27	3	19	42	2	14	54	2	14	0	14	54
XY	36	27	5	61	42	2	23	112	2	17	3	20.88	111
	50	27	No Solution			2	30	111	2	26	2	28.58	108
HS	36	27	4	62	60	2	23	165	2	19	2	21.58	167
	49	27	No Solution			2	30	160	2	26	3	29.88	161
IS-n	36	27	2	18	81	3	16	127	2	16	0	16	127
	50	27	No Solution			2	24	71	2	20	2	22.46	71
XY-n	40	27	No Solution			2	38	259	2	34	2	36.58	255
	50	27	No Solution			2	84	264	2	73	4	78.17	264
HS-n	40	27	No Solution			2	15	55	2	13	1	14.29	56
	50	27	No Solution			2	24	71	2	20	2	22.46	71
VQE	42	27	2	1	26	2	1	26	2	1	0	1	26
	50	27	2	1	26	2	1	26	2	1	0	1	25

Table 3. Comparison between 7-qubit device execution and QRCC (4-qubit device execution + post-processing).

Execution Mode	Results	Accuracy
State Vector simulation	-0.0349	100%
Shot-based Simulation	-0.0323	92%
Device Execution (7-qubit)	-0.0078	22.3%
QRCC-B	-0.0355	98.3%

to noisy qubits, the expectation value result from quantum device execution shows low accuracy; similar results were also observed in recent studies [45].

In addition, QRCC achieves better accuracy than that of shot-based simulation. The state vector of the shot-based simulation is 8× that of the 4-qubit device execution, which tends to introduce more randomness in output probability distribution than that of the real execution.

6.4 Studying δ Parameter

In Equation (18), assigning different δ values changes the priority on post-processing overhead and computation fidelity. We next study the impact on the effective cut numbers (i.e., #cuts) and the largest #MS in subcircuit with varying δ values. The post-processing overhead increases with larger #cuts values, and the computation fidelity increases with smaller #MS values. Figure 5 reports the average for the benchmarks with both W-Cut and G-Cut. The δ value varies from 0.1 to 1.0 as the post-processing overhead is the main design goal and thus cannot be completely ignored.

From the figure, #cuts decrease and #MS increases when the δ value increases. This is because giving higher priority

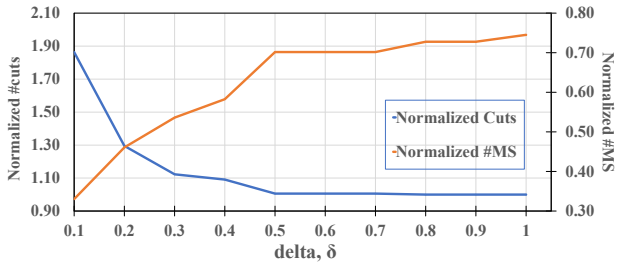


Figure 5. Correlating varying δ values with #cuts and #MS. The left y-axis represents the #cuts, normalized to that when δ=1. The right y-axis represents the #MS, normalized to the size of the original circuits.

to post-processing overhead, i.e., assigning a large δ value, minimizes #cuts. Given higher priority to computation fidelity, i.e., assigning a smaller δ value, minimizes #MS but hurts #cuts significantly. The figure also reveals that #cuts stabilize when δ>0.5, while the impact on #MS is significant. In the paper, we choose δ=0.7 for QRCC-B which exhibits negligible impact on #cuts but a large improvement on #MS. For a strategy that chooses a smaller δ value, e.g., δ=0.2, it increases #cuts by 30% on average. We observed that, for some benchmarks, it may find a solution that has a higher post-processing overhead than that from CutQC. Meanwhile, this choice leads to significantly improved computation fidelity, on average 52% improvement of #MS.

6.5 Time Comparison

We next compare the time required to find the cutting solutions using QRCC and CutQC. For a fair comparison, we

assume we know k, the number of subcircuits of the solution for each setting. This is because QRCC and CutQC work slightly differently. For QRCC, the user specifies a range $[C_{min}, C_{max}]$ and the subcircuit number of the found solution k is guaranteed to be within the range. For CutQC, the user needs to specify the exact k such that CutQC searches for the best solution with k subcircuits. For the latter, the user needs to manually increment k if a smaller k value results in *no-solution*. In the experiment, assuming we know k, we set C_{min}=k=C_{max} for QRCC and start with k for CutQC.

Table 4. The searching time comparison of the ILP model in QRCC and the MIP model in CutQC.

Benchmark			CutQC time	QRCC time	Improv.
Name	Circuit	Device			
SPM	15	7	0.80	0.63	21%
	20	7	11.7	6.21	47%
	30	16	7.05	0.54	92%
	42	16	65.16	18.1	72%
QFT	15	9	1800	1.42	100%
	30	24	1800	19.28	100%
	30	27	1800	1.92	100%
ADD	16	7	31.1	4.16	87%
	22	7	148.5	19.70	87%
	30	16	4.72	0.75	84%
	40	16	14.0	13.1	6%
AQFT	15	7	18.0	18.0	0%
	20	7	1800	1800	0%
	30	16	33.9	26.3	22%
	40	27	1565	1297	17%

Table 4 summarizes the wall clock time to find the solutions in Table 1. From the table, QRCC runs much faster than CutQC for most cases. On average, QRCC is 58% faster than CutQC, for cases where CutQC can find a solution. The main reason is that QRCC builds a linear model while CutQC adopts a non-linear model. The quadratic constraints in CutQC significantly slow down the search performance. In addition, without qubit reuse, CutQC introduces one extra qubit (i.e.,*initialization qubit*) after each cut, which increases the number of qubits in the subcircuit and makes it difficult to find a valid cutting solution.

6.6 Scalability

For cutting-based approaches such as QRCC and CutQC, the classical post-processing overhead dominates the overall overhead for handling large and complicated quantum circuits. In this section, we study the scalability of such approaches with scaled problem sizes.

6.6.1 Scalability vs #cuts. As discussed in Section 2, the classical post-processing overhead increases exponentially with increasing numbers of effective circuit cuts. We next compare the computation overhead using different schemes to reconstruct the results of the original circuit. We evaluate

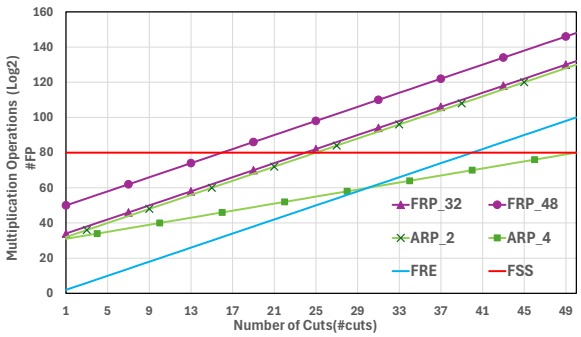


Figure 6. Comparison of the computation overhead with different reconstruction schemes: i) FRP (purple curves) – the hybrid full state reconstruction; ii) ARP (green curves) – the hybrid approximate reconstruction; iii) FSS (red curve) – the full-state simulation threshold; iv) FRE (blue curve) – the expectation value of the original circuit. The Y-axis indicates the log scale of the post-processing overhead in terms of #FP operations.

the computation overhead using the number of floating-point number operations (#FP) and ignore the memory requirement. We summarize the result in Figure 6. The X-axis indicates the number of circuit cuts (#cuts) and the y-axis indicates the log scale of required #FP for post-processing.

- **FSS**: the full-state simulation of a dense 34-qubit 1000-gate quantum circuit. It requires about 1e24 #FP, shown as the Horizontal red curve in the figure. Simulating such a circuit sequentially at the gate level may take several hours on CPU [1, 2, 9]. This is set as a threshold such that a reconstruction process is considered *too expensive* if its post-processing overhead exceeds this threshold. This threshold is set for illustration purposes and thus can be adjusted according to different settings.
- **FRP**: the hybrid full-state reconstruction for the probability vector of the original circuit. FRP assumes that we reconstruct two subcircuits cut from an N-qubit original circuit with *#cuts* cuts (X-axis). For simplicity, we assume that the original qubits are evenly distributed among two subcircuits.
- **FRE**: The reconstruction of FRE is for the expectation value of the original circuit. Shown as the blue curve in figure 6, this scheme has a similar assumption as above.
- **ARP-2**: the hybrid approximate reconstruction for the probability vector of the original circuit. ARP-2 assumes the reconstruction from two subcircuits and the original qubits are evenly distributed among them.
  ARP-2 differs from FRP in that it adopts an approximation strategy as follows. A full-state reconstruction strategy such as FRP faces a big challenge for circuits with large numbers of qubits. For example, saving the full probability vector of a 50-qubit quantum circuit demands O(2^{50}) or PB scale memory space, which is prohibitive for most small-

to medium-scale servers. An alternative approximate reconstruction [43] is to shrink the original vector space 2^{50} to a small vector space, e.g., 2^{30}. ARP-2 exploits the approximate reconstruction for all N>30.

- **ARP-4**: This is similar as ARP-2 but the reconstruction is conducted on four subcircuits. For simplicity, we horizontally partition the original circuit into four subcircuits (i.e., S1, S2, S3, and S4) such that the original qubits are evenly partitioned to the four subcircuits; and the cuts are evenly partitioned for cutting S1/S2, S2/S3, and S3/S4. A wire cut for S1/S2 indicates its measurement is in S1 and its initialization is in S2.

From the figure, the FRP_48 curve has the highest reconstruction cost, as a function of the number of $\#cuts$, because of the 2^{48} state space of the output vector. FRP reproduces the full state vector and thus demands $O(2^{N+2*\#cuts})$ #FP. FRE has much lower overheads, $O(2^{2*\#cuts})$, due to computing one expectation value instead of long probability vectors. This can be observed by the vertical distances between the purple lines (FRP_32 and FRP_48) to the blue curve (FRE). FRE independently computes the expectation value of each subcircuit instance, and multiples these expectation values together based on equation 3 and 4. As such, only scalar multiplication is required in FRE, and the number of scalar multiplication is independent of the number of qubits and is only affected by the number of $\#cuts$.

As a comparison, when N=48, FRE can tolerate 40 #cuts, while FRP_48 can only tolerate 16 #cuts before hitting the post-processing overhead threshold.

Furthermore, by exploiting approximate reconstruction, ARP-2 and ARP-4 can tolerate a larger number of #cuts, e.g., 25 and 50 #cuts respectively, before hitting the FSS threshold, as shown in the figure. This is because their overhead is qubit-independent when N>30. No matter the size of the circuit, only 2^{30} states of the original circuit are reproduced. One can also observe that when the original circuit is divided into more subcircuits, e.g., four subcircuits in ARP-4 instead of two in ARP-2, the reconstruction overhead decreases. This is because the overhead is dependent on the #cuts required to combine each two subcircuit pairs. Each combination of the subcircuit instances (e.g., S1/S2, S2/S3, and S3/S4) is independent of each other. For example, S1/S2 and S3/S4 can first be combined independently, as the cuts between S1/S2 do not have any effect on the combination of S3/S4 and vice-versa. This allows a divide-and-conquer strategy for the recombination of the original output, as the overhead only depends on the largest number of cuts among all the subcircuit pairs, not the total number of cuts aggregated across all subcircuit pairs. Consequently, the overhead increases at a slower pace than that of the total #cuts, validating that the use of recursive circuit-cutting improves the scalability of our proposed framework. However, we also want to mention that, while the increased number of subcircuits can have lower overhead in reconstruction, the complexity of circuit cutting increases due to the increased search space.

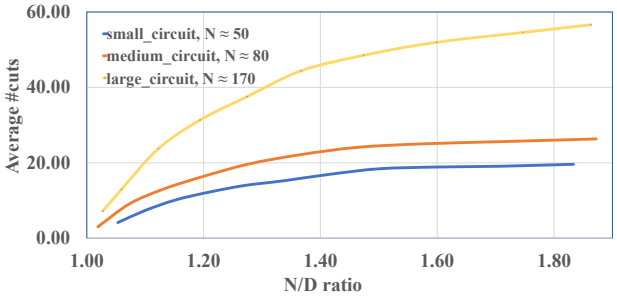


Figure 7. The #cuts values increase with larger N/D ratios. The N values are 50, 80, and 170 for small, medium, and large circuits, respectively.

From the figure, it is clear that the post-processing overhead is dominated by the number of cuts (#cuts). Therefore, the reduction of #cuts from QRCC over CutQC can effectively mitigate the corresponding post-processing computation time.

For example, for REG (N=40, D=27) in Table 2, the effective #cuts are reduced from 21 in CutQC to 16.3 in QRCC-C. This corresponds to $O(4^{21-16.29})$ reduction in computation overhead, or a 685× speedup of post-processing time, without considering the memory requirement.

6.6.2 Scalability vs N/D Ratio. We next correlate the number of circuit cuts #cuts to the circuit size N and the device size D. Intuitively, the cutting problem becomes more challenging when we have larger N values and smaller D values. Figure 7 reports the impact on #cuts with different N/D ratios with the results averaged on all benchmarks from table 2. We choose N= 50, 80, and 170 for small, medium, and large circuits, respectively.

From the figure, we observe that #cuts increase with larger N/D ratios. For small and medium circuits, the increase is moderate as there exists many qubit reuse opportunities. For large circuits, the increase is at a faster pace due to more two-qubit gates in the circuits.

6.6.3 Scalability vs Circuit Connectivity. To further study the impact of increased two-qubit gates, we fix N and D values for a subset of large circuits (in Table 2) whose circuit complexity can be adjusted with a meta-parameter, and summarize the required number of circuit cuts in Table 5. As discussed above, increasing the N/D ratio from 200/150 to 300/200 for REG (m=3) leads to more cuts for both schemes. We also observe that, by adjusting the meta-parameter m (from m=3 to m=4), the circuits contain more two-qubit gates and thus demand around double the amount of cuts. For more complex circuits, e.g., choosing N=300, D=200, and p=0.02 for ERD, our model still scales well and finds a solution. However, the solution contains large numbers of wire cuts

Table 5. The scalability correlates to circuit size and complexity.

Benchmark			QRCC		CutQC
name	N	D	#W-Cuts	#G-Cuts	#W-Cuts
REG (m=3)	200	150	19	0	21
REG (m=3)	300	200	31	3	36
REG (m=4)	200	150	36	3	49
REG (m=4)	300	200	61	6	75
BAR (m=4)	200	150	74	3	No Solution
BAR (m=2)	300	200	55	1	60
ERD (p=0.05)	200	150	96	2	No Solution
ERD (p=0.02)	300	200	52	104	No Solution

and gate cuts, indicating that the bottleneck has shifted to the post-processing overhead, i.e., O($4^{52}6^{104}$).

6.7 Qubit Reuse in Cutting

Based on the observation that QRCC exploits qubit reuse to find better cutting solutions than those of CutQC, it becomes interesting to investigate if naively combining CutQC and qubit reuse can achieve similar results.

To partition a N-qubit original circuit into smaller subcircuits that can run on D-qubit quantum devices, we have two simple approaches to combine CutQC and qubit reuse.

(i) For the first approach, we apply CutQC to partition the original circuit into small subcircuits that can each run on N-qubit devices, and then optimize each subcircuit using qubit reuse. Compared to QRCC, this is a sub-optimal approach because its first step often results in a cutting solution with more cuts than that of QRCC. Applying qubit reuse at the second step, even if it reduces the number of required qubits for each subcircuit, shall not help to reduce the post-processing overhead.

(ii) Alternatively, we may partition the original circuit into subcircuits that can run on X-qubit devices, where $N>X>D$, assuming we can reduce X to D by applying qubit reuse. Unfortunately, the assumption is not always true.

For example, we choose QFT with N=15 and D=7, QRCC finds a cutting solution that partitions the circuit into three subcircuits with 20 wire cuts. Given that CutQC cannot find a solution for D=7 or 8, we may choose other solutions and then apply qubit reuse, we try all different $N>X>D$ settings and summarize the results in Table 6.

From the table, sequentially applying CutQC and qubit reuse cannot find a solution as good as the one from QRCC. The closest one is the solution at X=9 when all subcircuits after the cut can run on 9-qubit devices. Applying qubit reuse enables them to run on 7-qubit devices. However, the number of cuts is more than twice the number of our solution, i.e., 44 vs 20. For all other settings, the required qubits for the subcircuits can be reduced after reuse, but the subcircuits still cannot run on 7-qubit quantum computers.

Table 6. Applying CutQC and qubit reuse sequentially produces sub-optimal results.

Device size	CutQC			+ CaQR
	#SC	#cuts	width	width
9	9	44	9	7
10	4	24	10	8
11	4	20	11	10
12	4	20	12	10
13	4	20	12	10
14	4	20	12	10

7 Related Work

The recent studies on circuit-cutting focus mainly on lowering the reconstruction overhead. Lowe *et al.*[29] proposed to reduce the overhead of wire cutting using randomized probabilistic measurements. Piveteau *et al.*[38] proposed to reduce the overhead of gate cutting, using classical two-way communication and shared bell pair between subcircuits. These works assume that the input is a pre-cut circuit and thus are orthogonal to our work.

Xie *et al.* proposed a compiler framework for distributed quantum computing [49]. Smith *et al.* exploited circuit-cutting and Clifford gate simulation to enhance the reach of classical quantum circuit simulation and the simulation time [41].

8 Conclusion

In this paper, we propose QRCC for evaluating large quantum circuits on small quantum computers. QRCC integrates wire cutting and qubit reuse in one framework to find good cutting solutions for quantum circuits that compute probability vectors, and in addition with gate cutting for circuits that compute expectation values. We formulate the problem as an ILP model to find the cutting solutions efficiently.

9 Acknowledgments

This work is supported by the National Science Foundation under award numbers #2334628, #2011146, #2154973, and #2312157, and Pittsburgh Quantum Institute under award number #007913. We extend our gratitude to the ASPLOS reviewers for their valuable insights and feedback.

References

[1] Google quantum ai. https://quantumai.google/qsim/choose_hw.

[2] State vector simulator (sv1) - amazon braket. https://docs.aws.amazon.com/braket/latest/developerguide/braket-simulator-sv1.html.

[3] Daniel S. Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Phys. Rev. Lett.*, 79:2586–2589, Sep 1997.

[4] Ramin Ayanzadeh, Narges Alavisamani, Poulami Das, and Moinuddin Qureshi. Frozenqubits: Boosting fidelity of qaoa by skipping hotspot nodes. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 311–324, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Albert-Laszlo Barabasi and Rita Albert. Albert, r.: Emergence of scaling in random networks. science 286, 509-512. *Science (New York, N.Y.)*, 286:509–12, 11 1999.

[6] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Päivi Törmä. Approximate quantum fourier transform and decoherence. *Phys. Rev. A*, 54:139–146, Jul 1996.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, September 2017.

[8] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, April 2018.

[9] A. Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, 1993.

[10] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[11] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit, 2004.

[12] Poulami Das, Eric Kessler, and Yunong Shi. The imitation game: Leveraging copycats for robust native gate selection in nisq programs. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 787–801, 2023.

[13] Poulami Das, Aditya Locharla, and Cody Jones. Lilliput: A lightweight low-latency lookup-table decoder for near-term quantum error correction. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 541–553, New York, NY, USA, 2022. Association for Computing Machinery.

[14] Poulami Das, Swamit Tannu, Siddharth Dangwal, and Moinuddin Qureshi. Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling. pages 950–962, 10 2021.

[15] Yongshan Ding and Frederic Chong. Quantum computer systems: Research for noisy intermediate-scale quantum computers. volume 15, pages 1–227, 06 2020.

[16] Yongshan Ding, Xin-Chuan Wu, Adam Holmes, Ash Wiseth, Diana Franklin, Margaret Martonosi, and Frederic T. Chong. Square: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 570–583, 2020.

[17] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[18] Jay Gambetta. Quantum-centric supercomputing: The next wave of computing, Oct 2023.

[19] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[20] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[21] Fei Hua, Yuwei Jin, Yanhao Chen, Suhas Vittal, Kevin Krsulich, Lev Bishop, John Lapeyre, Ali Javadi-Abhari, and Eddy Zhang. Caqr: A compiler-assisted approach for qubit reuse through dynamic circuit. pages 59–71, 03 2023.

[22] Yipeng Huang and Margaret Martonosi. Qdb: From quantum algorithms towards correct quantum programs. 2019.

[23] Blake Johnson. The full power of dynamic circuits to qiskit runtime, Nov 2022.

[24] Mohammad Reza Jokar, Richard Rines, Ghasem Pasandi, Haolin Cong, Adam Holmes, Yunong Shi, Massoud Pedram, and Frederic T. Chong. Digiq: A scalable digital controller for quantum computers using sfq logic. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 400–414, 2022.

[25] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White. Towards quantum chemistry on a quantum computer. *Nature Chemistry*, 2(2):106–111, January 2010.

[26] Lingling Lao and Dan E. Browne. 2qan: a quantum compiler for 2-local qubit hamiltonian simulation algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 351–365, New York, NY, USA, 2022. Association for Computing Machinery.

[27] Ji Liu, Gregory T. Byrd, and Huiyang Zhou. Quantum circuits for dynamic runtime assertions in quantum computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 1017–1030, New York, NY, USA, 2020. Association for Computing Machinery.

[28] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning, 2013.

[29] Angus Lowe, Matija Medvidović, Anthony Hayes, Lee J. O'Riordan, Thomas R. Bromley, Juan Miguel Arrazola, and Nathan Killoran. Fast quantum circuit cutting with randomized measurements. volume 7, page 934. Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, March 2023.

[30] Kosuke Mitarai and Keisuke Fujii. Constructing a virtual two-qubit gate by sampling single-qubit operations. volume 23, page 023021. IOP Publishing, feb 2021.

[31] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, jun 2018.

[32] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, jun 2018.

[33] 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 *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 1015–1029, New York, NY, USA, 2019. Association for Computing Machinery.

[34] Román Orús, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[35] Adam Paetznick and Krysta Marie Svore. Repeat-until-success: non-deterministic decomposition of single-qubit unitaries. *Quantum Inf. Comput.*, 14:1277–1301, 2013.

[36] Tirthak Patel, Ed Younis, Costin Iancu, Wibe de Jong, and Devesh Tiwari. Quest: Systematically approximating quantum circuits for higher output fidelity. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 514–528, New York, NY, USA, 2022. Association for Computing Machinery.

[37] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. Simulating large quantum circuits on a small quantum computer. volume 125, page 150504. American Physical Society, Oct 2020.

[38] Christophe Piveteau and David Sutter. Circuit knitting with classical communication. pages 1–1. Institute of Electrical and Electronics Engineers (IEEE), 2023.

[39] G. Ravi, K. N. Smith, P. Gokhale, A. Mari, N. Earnest, A. Javadi-Abhari, and F. T. Chong. Vaqem: A variational approach to quantum error mitigation. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 288–303, Los Alamitos, CA, USA, apr 2022. IEEE Computer Society.

[40] Gokul Subramanian Ravi, Kaitlin Smith, Jonathan M. Baker, Tejas Kannan, Nathan Earnest, Ali Javadi-Abhari, Henry Hoffmann, and Frederic T. Chong. Navigating the dynamic noise landscape of variational quantum algorithms with qismet. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 515–529, New York, NY, USA, 2023. Association for Computing Machinery.

[41] Kaitlin N. Smith, Michael A. Perlin, Pranav Gokhale, Paige Frederick, David Owusu-Antwi, Richard Rines, Victory Omole, and Frederic Chong. Clifford-based circuit cutting for quantum simulation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[42] A. STEGER and N. C. WORMALD. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8(4):377–396, 1999.

[43] Wei Tang and Margaret Martonosi. Scaleqc: A scalable framework for hybrid computation on quantum and classical processors, 2022.

[44] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. Cutqc: Using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 473–486, New York, NY, USA, 2021. Association for Computing Machinery.

[45] Swamit Tannu, Poulami Das, Ramin Ayanzadeh, and Moinuddin Qureshi. Hammer: Boosting fidelity of noisy quantum circuits by exploiting hamming behavior of erroneous outcomes. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 529–540, New York, NY, USA, 2022. Association for Computing Machinery.

[46] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The variational quantum eigensolver: A review of methods and best practices. *Physics Reports*, 986:1–128, November 2022.

[47] Hanrui Wang, Jiaqi Gu, Yongshan Ding, Zirui Li, Frederic T. Chong, David Z. Pan, and Song Han. Quantumnat: Quantum noise-aware training with noise injection, quantization and normalization. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 1–6, New York, NY, USA, 2022. Association for Computing Machinery.

[48] Anbang Wu, Gushu Li, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie. A synthesis framework for stitching surface code with superconducting quantum devices. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 337–350, New York, NY, USA, 2022. Association for Computing Machinery.

[49] Anbang Wu, Hezi Zhang, Gushu Li, Alireza Shabani, Yuan Xie, and Yufei Ding. Autocomm: A framework for enabling efficient communication in distributed quantum programs. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1027–1041, 2022.

[50] Xin-Chuan Wu, Sheng Di, Emma Maitreyee Dasgupta, Franck Cappello, Hal Finkel, Yuri Alexeev, and Frederic T. Chong. Full-state quantum circuit simulation by using data compression. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[51] Lei Xie, Jidong Zhai, ZhenXing Zhang, Jonathan Allcock, Shengyu Zhang, and Yi-Cong Zheng. Suppressing zz crosstalk of quantum computers through pulse and scheduling co-optimization. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 499–513, New York, NY, USA, 2022. Association for Computing Machinery.

[52] Mingkuan Xu, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma, Jens Palsberg, Alex Aiken, Umut A. Acar, and Zhihao Jia. Quartz: Superoptimization of quantum circuits. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2022, page 625–640, New York, NY, USA, 2022. Association for Computing Machinery.