FMCC: Flexible Measurement-based Quantum Computation over Cluster State

Yingheng Li
yil392@pitt.edu
University of Pittsburgh
Pittsburgh, USA

Aditya Pawar
adp110@pitt.edu
University of Pittsburgh
Pittsburgh, USA

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

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

Jun Yang
juy9@pitt.edu
University of Pittsburgh
Pittsburgh, USA

Xulong Tang
tax6@pitt.edu
University of Pittsburgh
Pittsburgh, USA

Abstract

Measurement-based quantum computing (MBQC) is a promising quantum computing paradigm that performs computation through "one-way" measurements on entangled quantum qubits. It is widely used in photonic quantum computing (PQC), where the computation is carried out on photonic cluster states (i.e., a 2-D mesh of entangled photons). In MBQC-based PQC, the cluster state depth (i.e., the length of one-way measurements) plays an important role in the overall execution time and circuit error. In this paper, we propose FMCC, a compilation framework that employs dynamic programming with heuristics to efficiently minimize the cluster state depth. Experimental results on six quantum applications show that FMCC achieves 51.7%, 57.4%, and 56.8% average depth reductions in small, medium, and large qubit counts compared to the state-of-the-art MBQC compilations.

1 Introduction

Quantum computing has experienced rapid development in the past decades, demonstrating its supremacy over classical computers in various application fields [8, 20, 43]. Most quantum computing primarily focuses on matter-based quantum computers that utilize superconducting [2] and trapped-ion [9] qubits, following the gate-based quantum computation paradigm. Recently, there has been a growing interest in photonic quantum computing (PQC) that employs photon qubits [7, 36, 44]. PQC generally leverages measurement-based quantum computation (MBQC) that carries out computation through "one-way" measurements on a 2-D mesh of entangled photon qubits (known as cluster states) [7, 40]. Specifically, computation is conducted by measuring the photons and propagating the measurement results through a cluster state [7, 40, 41]. MBQC-based PQC offers several advantages compared to gate-based quantum computation, such as longer coherence times and better scalability [37].

In MBQC, a quantum algorithm is executed by mapping the measurement patterns of each quantum gate onto a cluster state (detailed discussion in Section 2.1). A cluster state is generated using cluster state generator hardware that produces entangled photons with fixed cluster state width, where the width is the number of rows of entangled photons in the 2-D mesh [7, 10, 15, 18, 27, 38, 40]. In MBQC, the computation is realized by measurements to process and transmit information between neighboring entangled photons. As such, the depth (i.e., the number of photon columns to execute a quantum algorithm) is a metric that critically affects the execution time and error of quantum algorithms.

In this paper, we observe that modern MBQC execution on cluster states is not optimized for the cluster state depth, resulting in a long depth that induces a large photon resource cost [15, 18, 38] and increases quantum error rate [21, 44, 46]. To this end, we propose FMCC (Flexible MBQC Compilation on Cluster State), an MBQC compilation framework that achieves the minimized depth to execute quantum algorithms on a cluster state. Specifically, the motivation stems from our observations that i) a significant amount of photons are wasted in cluster states during execution, and ii) the MBQC execution allows different valid mapping "variants". To generate different MBQC mapping variants that

utilize the wasted photons, we categorize the measurements in MBQC circuits into "components" — each being a group of neighboring measurements. We identify two types of mapping variants, intra-component and inter-component, and employ dynamic programming with heuristics to exploit the variants to generate the MBQC circuit with minimized depth on a cluster state.

To the best of our knowledge, the only existing work that optimizes the depth is proposed by Li et al. [29]. While their work primarily focuses on reducing the number of redundant photon measurements, they do not explore MBQC mapping variants, yielding sub-optimal depth. In this paper, we use their work as our baseline and provide a quantitative comparison in Section 6.2 and Section 6.3. The main contributions of the paper are the following:

- We identify cluster state depth as a key metric for reducing the execution time of quantum algorithms on photonic cluster states. We find that an alternative MBQC mapping effectively reduces cluster state depth.

- We categorize the photon measurements on cluster state into different components. We define the component constraints in circuit mapping and use components to generate mapping variants. Then, we formulate the problem into a dynamic programming problem and exploit both intra-component and inter-component mapping variants. This significantly reduces the mapping search space and avoids simple heuristic pitfalls.

- We conduct experiments using six representative MBQC quantum applications with a small, medium, and large qubit count. Experimental results indicate that, compared to the state-of-the-art MBQC compilation, FMCC achieves 51.7%, 57.4%, and 56.8% average depth reductions in small, medium, and large qubit counts, respectively. We also report the error impact of FMCC and show it improves the overall circuit fidelity compared to state-of-the-art MBQC compilers.

2 Background

2.1 Photonic Cluster State

A cluster state consists of photon qubits initially prepared in the $|+\rangle$ state where each photon is entangled with its neighboring photons. A cluster state is generated by a photonic hardware generator (i.e., cluster state generator) that continuously produces entangled photons [10, 15, 18, 27, 38, 46, 48]. Fig. 1(c) shows such a generation process in which the generator has a fixed width of 3 photons and the entangled photons are generated from left to right. In the 2-D mesh cluster state, the *width* and *depth* are important for a quantum circuit execution on a cluster state [24, 29, 40].

Cluster state width. The width refers to the number of rows in a cluster state. A fixed width simplifies photonic cluster state generator hardware, which is a typical design [10, 15, 18, 27, 38, 48]. However, the minimum width required by each quantum algorithm varies according to the
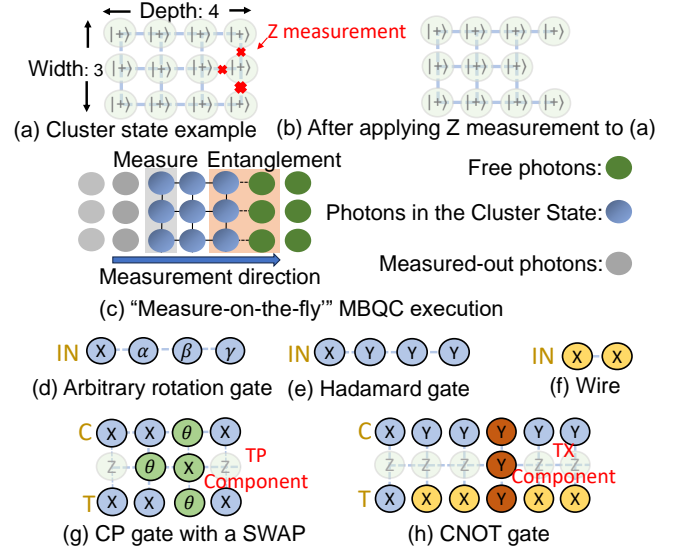

Figure 1. Cluster state and Measurement pattern examples.

number of logical qubits it has [7, 29, 40]. Specifically, each logical qubit is mapped to a row of photons in the cluster state. To prevent undesired interactions between two logical qubits, the corresponding photon rows cannot be adjacent since the adjacent photons are entangled in a cluster state. Therefore, MBQC cuts out one row of photons between the two rows where logical qubits are mapped. Hence, a quantum algorithm requires a minimum of $2N_q - 1$ rows to be executed in a cluster state, where N_q represents the number of qubits in the quantum algorithm [29, 40, 41].

Cluster state depth. The depth refers to the number of columns in a cluster state. A hardware generator produces entangled photons from left to right, where there are no hardware generator constraints in the cluster state depth. Thus, the depth of the cluster state is only determined by a quantum algorithm and its mapping on the cluster state. The depth directly affects execution time and error rate: the shorter the cluster state depth, the faster the execution and the lower the error rate [10, 15, 18, 38, 44, 46, 48]. Therefore, our goal is to reduce the depth in a fixed-width cluster state.

There are two approaches to generating a photonic cluster state: i) emitter-based and ii) fusion-based. In the emitter approach, a 2D cluster state is generated deterministically by using entangled quantum emitters [15, 18, 48] or one emitter with feedback loop [17, 38]. While the emitter approach guarantees successful cluster state generation, it is limited by the emitter decoherence [15, 18, 30, 46, 48]. In contrast, the fusion approach generates cluster states by merging multiple resource states (i.e., each resource state consists of three entangled photons) [4, 19]. While it shows advantages in cluster state scalability and generation speed [4], the fusion operation (shown in Fig. 2(a)) has only a 75% success rate. That is, the more fusion being used, the less possibility to

generate a cluster state successfully. In this paper, we primarily focus on emitter-based cluster state generation. However, our approach is also applicable in fusion-based cluster state generation and mitigates the error probability in both approaches (Section 6.5).

Fig. 1(c) also illustrates the "measure-on-the-fly" execution paradigm employed in cluster states [36, 41, 44] for both cluster state generation methods. That is, the measurements are applied to the photons while the cluster state is generated. This avoids the need to store a large-scale photonic cluster state in optical quantum memory, increasing MBQC's scalability [36, 41, 44]. Note that the measurement direction must be the same as the cluster state generating direction in this paradigm (i.e., from left to right). As shown in Fig. 1(c), the cluster state generator entangles free photons (green circles) into a cluster state (blue circles). Meanwhile, a column of photon detectors measures the leftmost column of the cluster state to perform computations. Once measured, the photons (gray circles) are no longer entangled.

2.2 MBQC on Cluster State

MBQC employs two types of measurements to achieve universal quantum computation [39, 40]: i) X and Y measurements (σ_x and σ_y eigenbasis) to perform Pauli gate operations [39, 40], and ii) θ measurements to perform non-Pauli gate operations. Implementation of quantum gates (also called "pattern" [40]) is achieved by a set of measurements applied to entangled photons in cluster states. Fig. 1 shows several pattern examples with annotated measurement angles. The patterns of a single-qubit arbitrary rotation gate and an H gate are shown in Fig. 1(d) and (e), where measurements are applied to a row of consecutive entangled photons and computation is carried out "one-way" from left to right. The consecutive even-number X measurements serve as identity gates or "wires", allowing logical qubit transmission through qubit rows without applying any operations [40] (Fig. 1(f)). MBQC uses Z measurements to "cut out" entangled photons from a cluster state [39, 40]. Fig. 1(a) shows an example of applying a Z measurement to a photon in the cluster state, destroying the entanglements between the photon and its neighbors. The resulting cluster state is shown in Fig. 1(b). The Z measurements eliminate unwanted interactions between logical qubits by separating qubit rows in a cluster state. It also eliminates any photons within the cluster state that are not involved in the computation.

Two notable two-qubit gate patterns are the combined CP and SWAP gate and the CNOT gate, shown in Fig. 1(g) and (h), where measurements are applied to three rows of entangled photons. The first row corresponds to the control qubit, whereas the third row corresponds to the target qubit. Among the measurements in the middle row, the X, θ, and Y measurements facilitate communication between the two logical qubits, while the Z measurements cut out unused photons and separate rows for control and target qubits. While
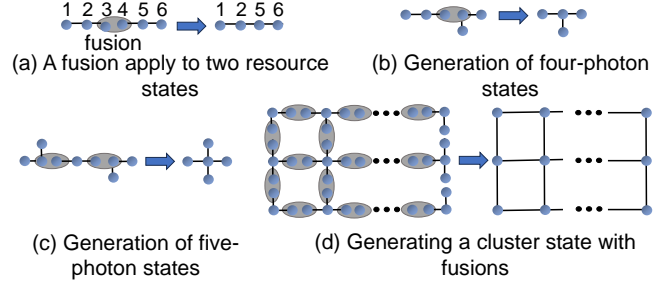

Figure 2. Examples of using fusions to generate cluster state.

CNOT and single-qubit gates together enable universal quantum computation [35, 42], MBQC provides further advantages as CNOT gates together with combined CP and SWAP gates reduce the need for quantum gate decomposition and enhance the flexibility of quantum computation [40].

3 Error Models of Photonic Cluster State

There are four major errors of MBQC on the cluster state:

- **Emitter decoherence**: If the cluster state is generated by emitters, the fidelity of photons is affected by the decoherence of the emitter [15, 18, 30, 46, 48]. The emitter decoherence error rate is $1 - e^{\frac{-t}{T_2}}$, where T_2 is the coherence time of the emitter and t is the total time to generate a cluster state. Since a column of photons is emitted at a constant rate, t is proportional to the depth of a cluster state d. In our modeling, we set $T_2 = 4.4\mu s$ and generate a column of photons in 0.2 ns based on prior work [18, 23].

- **Fusion failure**: As introduced in [4, 19], a cluster state can also be generated by merging multiple resource states using fusions. Since a fusion has a success rate of 75%, the success rate of generating a cluster state using fusions is 0.75^{N_f}, where N_f is the number of fusions. To generate a cluster state, it requires four-photon states and five-photon states, as illustrated in Fig. 2(b) and Fig. 2(c). The figures also show how to use fusion to generate four-photon states and five-photon states. Consequently, the generation of a cluster state with depth d and width w (assuming $d > 3$ and $w > 3$) requires four resource states, $2(w-2) + 2(d-2)$ four-photon states, $(w-2) \cdot (d-2)$ five-photon states, and additional fusion operations to merge them. Therefore, the total number of fusions to generate a cluster state with width w and depth d is $N_f = 2(w-2) + 2(d-2) + 2(w-2) \cdot (d-2) + (d-1) \cdot w + (w-1) \cdot d = 4w \cdot d - 3w - 3d$. As such, the smaller the width and the depth, the less number of fusions are required to generate the cluster state.

- **Photon loss**: In photonic quantum computing, photon loss (i.e., measurement failure) is the dominant error [12, 21, 36, 46]. Photon loss can be caused by: i) inefficient photon detector, or ii) imperfect photon creation [7, 21, 44, 46]. The success rate of a photon measurement is calculated as $S_m = S_d \cdot S_p$, where S_d is the success rate of a photon
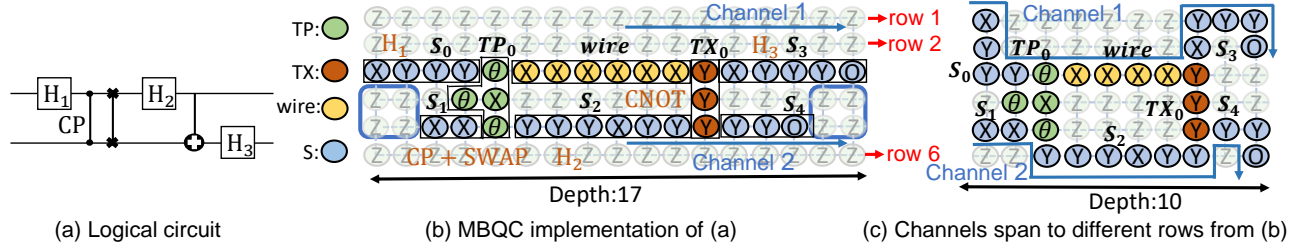
Yingheng Li, Aditya Pawar, Zewei Mo, Youtao Zhang, Jun Yang, and Xulong Tang

Figure 3. Example of a quantum algorithm mapped on a cluster state.

detector and S_p is the successful generation rate of a single photon. According to [7, 12, 21, 46, 47], S_p and S_d are constant for every photon measurement (i.e. Z, X, Y, and θ measurements). Therefore, the success rate for measurements of all photons in a cluster state can be calculated as $S_{total} = S_m^{n_p}$, where n_p is the total number of measurements and S_m is a constant [21, 46]. Since *all* photons in a cluster state are measured [40], the total number of measurements n_p in a cluster state with width w and depth d is $n_p = w \cdot d$.

- **Pauli/Computation error**: Each measurement in an MBQC circuit excluding the Z measurements (i.e., X, Y, and θ measurements) can cause Pauli (i.e., computation) error [3, 7, 12, 25]. Note that, Z measurements do not induce Pauli errors as they do not participate in computation. Assuming each non-Z measurement has a Pauli error probability P_c, the fidelity of a given quantum circuit is $(1 - P_c)^{n_c}$, where n_c is the total number of non-Z measurements in the cluster state. Since P_c does not change with cluster state depth and width [3, 7, 12, 25], the fidelity of the quantum circuit is affected by n_c (i.e., the total number of non-Z measurements).

4 Opportunities and Challenges

We use a real MBQC example to illustrate the opportunities and challenges in optimizing the cluster state depth. Fig. 3(a) shows a two-qubit quantum algorithm, where its corresponding MBQC implementation on a photonic cluster state is shown in Fig. 3(b). After the computation on a logical qubit (i.e., a photon row), an output photon is employed for readout measurement O. Fig. 3(b) shows the mapping from a quantum algorithm to a cluster state, which we refer to as an MBQC circuit in the rest of this paper.

The one-way computation in MBQC can result in long cluster state depth, which is often due to inefficient photon usage. For example, only three of six photon rows are being used in Fig. 3(b). That is, rows 1, 2, and 6 in Fig. 3(b) are not used by effective computation and are cut out by Z measurements. Moreover, there are Z measurements that cut out photons *within* the three used photon rows, as indicated by the blue boxes. This motivates the question: *Can we leverage these unused photons and rows to reduce the circuit depth?*

4.1 MBQC Circuit Variants

In practice, the measurements corresponding to a logical qubit in a cluster state are not necessarily mapped to the same row. This is because the quantum computation on a logical qubit remains unchanged as long as the same measurements are made on entangled photons in the same order as in a cluster state [39, 40]. That is, one can leverage the entangled photons in the same column, rather than being strictly limited to entangled photons in the same row. We define the measurement sequences on entangled photons in cluster states as "channels". Note that measurements within a column can be performed either simultaneously or sequentially [36, 44]. In the baseline MBQC circuit mapping, one channel is necessarily mapped to one photon row. However, if we allow the channel to span multiple rows, it can leverage unused photons and potentially reduce depth.

Different MBQC circuits can be generated when the channel spans multiple photon rows in a cluster state. To explore shorter MBQC circuits with the same function, we first partition an MBQC circuit into different components, then explore the variant of each component and connect them together. Each component is a group of adjacent measurements with a specific functionality. We define four types of components:

- **TP**: The TP component is used to connect the control qubit (i.e., control channel) and the target qubit (i.e., target channel) in the combined CP and SWAP gate pattern. It consists of one X and three θ measurements. The four measurements of a TP component are colored in green in Fig. 1(g).
- **TX**: The TX component consists of three Y measurements and is used to connect the control qubit and the target qubit in CNOT gate patterns. The TX component is colored in red in Fig. 1(h). The three Y measurements in the TX component are required to be measured simultaneously [40].
- **S**: The S component represents single-qubit measurements. An S component consists of consecutive measurements within individual channels, e.g., Fig. 1(d) and Fig. 1(e). Note that an S component cannot include pairs of X measurements because they only form wires.
- **wire**: A wire is a consecutive sequence of an even number of X measurements. As previously discussed, a wire only serves as a connector between two components and does not perform any quantum computation.

Based on these four types of components, we next define intra-component variants and inter-component variants.

Intra-component variants: For each component, there are multiple MBQC mappings that have the same quantum functionality as in a baseline MBQC circuit. Specifically, each type of component has its own unique set of variants:

- **TP variants**: A TP component has only one valid variant. Fig. 4(a) shows the TP component in the baseline MBQC circuit, whereas Fig. 4(b) shows its valid variant and one example of an invalid variant. The valid variant swaps the θ_0 and θ_2 measurements in the first and third rows, respectively. As discussed in Section 2.1, the measurements in a cluster state are performed from left to right. This implies that measurements in the first column should be executed before those in the second column. As such, the valid variant preserves the same order as the baseline because the same measurements are present in each column. In contrast, the invalid variant does not preserve the same measurement order, resulting in an incorrect computation due to a different measurement sequence [39, 40].

- **TX variants**: Similar to the TP component, the TX component also has only one valid variant. Fig. 4(c) shows the baseline TX component and Fig. 4(d) shows its valid variant and one example of an invalid variant. The valid TX variant also swaps the baseline TX component measurements. In both the baseline TX and the valid TX variant, the three Y measurements can be performed simultaneously as they are in the same column. However, in the invalid variant, Y_2 and Y_3 are measured after Y_1, leading to incorrect computation.

- **S variants**: Fig. 4(e) shows an example of an S component with three measurements. In the baseline S component, measurements follow a left-to-right sequence, from X to Y_1. Although the two Y measurements are in the same column in the valid variant shown in Fig. 4(f), X is measured first in the left column and Y_0 can be measured before Y_1 [36, 44]. Hence, the same measurement order can still be kept. In contrast, in the invalid variant, the X measurement is to the right of the two Y measurements and it would be measured after these two.

- **wire variants**: A wire component only requires an even number of X measurements. Fig. 4(g) shows three variants of a wire component. The valid variant I has six X measurements and the valid variant II has two. The invalid variant has one measurement, changing the wire's function.

Inter-component variants: Between two adjacent components, we refer to the component on the left as the "left component" and the component on the right as the "right component," and refer to measurements linking these adjacent components as "connected points." By adjusting the positions between two adjacent components, we can decrease the depth of an MBQC circuit by optimizing photon use in a cluster state. In this process, it is essential to maintain the connection between adjacent components, where the right
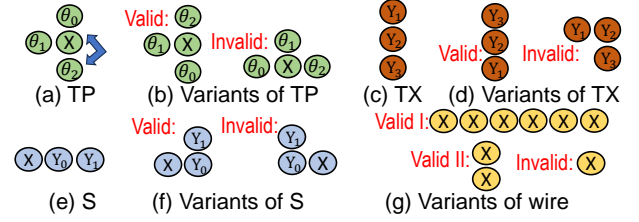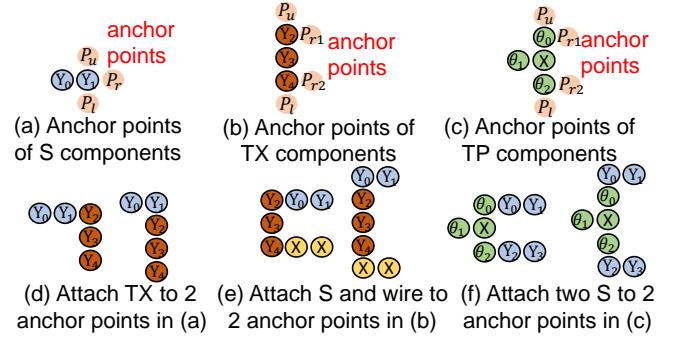

Figure 4. Intra-component variants.

Figure 5. Anchor points in components.

component's connected point should attach to the "anchor points" on the left component. Anchor points are defined as the unused neighboring photons of the left component's connected point, where the right components can be attached. For example, Fig. 5(a) shows the three anchor points of the S component ($P = \{P_u, P_r, P_l\}$). Applying the TX component to P_u is not possible as it results in overlapping measurements, which is not allowed in MBQC [40]. As shown in Fig. 5(d), anchoring at P_l reduces circuit depth compared to P_r. Unlike the S component, TP and TX components offer four anchor points, as shown in Fig. 5(b) and (c). In Fig. 5(e) and (f), each right component of the TP and TX components applies to a different pair of anchor points, where the pair (P_u, P_l) minimizes circuit depth.

Illustrating intra- and inter-component opportunities: Exploiting both intra-component and inter-component variants allows us to reduce the circuit depth while maintaining the correct quantum function. An example is shown in Fig. 3(c) which is functionally equivalent to the MBQC circuit in Fig. 3(b). Specifically, for intra-component, we use variants of S_0, S_3, and S_4 that span more rows to reduce the depth. For inter-component, we attach S_2 to the lower anchor point of TP_0, attach TX_0 to the upper anchor point of S_2, and attach S_3 to the upper anchor point of TX_0. As a result, the depth is reduced from 17 (in Fig. 3(b)) to 10 (in Fig. 3(c)).

4.2 MBQC Constraints

While exploiting intra- and inter-component variants allows us to reduce the depth, it is a nontrivial task where certain constraints have to be satisfied to guarantee correctness.
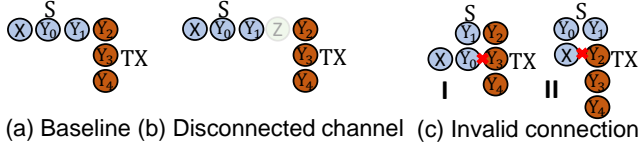
(a) Baseline (b) Disconnected channel (c) Invalid connection

Figure 6. Examples of violating constraints.

(a) Example component DAG (b) Partial circuits constructed at each iteration and components within them

Figure 7. Partial circuit set constructed at each iteration.

Constraint-I Channel connectivity. It is crucial to ensure channel connectivity in the MBQC circuit. That is, the sequence of measurements (i.e. the channel) should remain the same as in the baseline circuit. This constraint is violated if a channel is disconnected between two adjacent measurements. In Fig. 6(a), we show an S component adjacent to a TX component. Fig. 6(b) shows this constraint violation due to a disconnected channel.

Constraint-II Invalid connection. Measurements that have no connection in the baseline MBQC circuit should remain disconnected. This violation occurs when using an inappropriate variant or connecting components to inappropriate anchor points. For instance, in example I of Fig. 6(c), an invalid connection, marked by the red cross, is caused by the adoption of an inappropriate variant of the S component where the Y_0 and Y_3 measurements are connected. In example II of Fig. 6(c), there is an invalid connection between X and Y_2 due to the inappropriate anchor points.

Constraint-III Component dependency. The dependency among the components has to be maintained when exploring MBQC circuit variants to ensure quantum computation correctness. That is, any component (including all the measurements in it) should not be conducted before those components it depends on.

Constraint-IV Cluster state width. The depth-reduced MBQC circuit must not have variants that exceed the fixed width of the cluster state. Violating this constraint makes the quantum circuit inexecutable on the cluster state [7, 29, 40].

5 Our Approach

In this paper, we design FMCC (Flexible MBQC Compilation on Cluster State), an MBQC compilation framework that exploits both intra-component and inter-component variants while satisfying the aforementioned constraints. However, this process faces two challenges. First, the design space can be prohibitively large, making exhaustive search practically infeasible. For instance, given a circuit with n components, if each component has v variants and k anchor points, conceptually, one can have $v^n \times k^{n-1}$ MBQC circuits. While a simple heuristic may work, the second challenge is that among the $v^n \times k^{n-1}$ MBQC circuits, many of them may be invalid due to constraints violations. As such, a heuristic approach may lead to invalid MBQC circuits. To this end, our proposed FMCC employs dynamic programming (DP) with heuristics
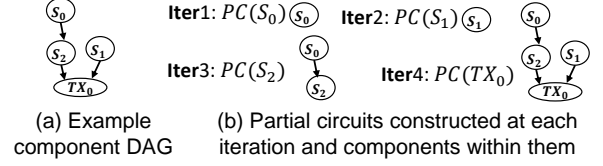
to construct a valid MBQC circuit with reduced depth, while addressing the two challenges.

5.1 Constructing MBQC Circuits

In FMCC, MBQC circuits are built component-wise. The construction sequence is guided by a Directed Acyclic Graph (DAG), which captures the dependencies among non-wire components, referred to as a component DAG[1]. We iterate through each component of the DAG, to generate a set of variants for that component. In each iteration, one component is chosen from the DAG. We define a variant set for a component chosen at iteration k as:

Definition 1. *(Variant set) The variant set $V(c)$ at iteration k is the set of all valid variants of the component c selected at iteration k, including its baseline variant.*

Starting from the independent components, i.e., components without parents in the DAG, a set of partial circuits is constructed iteratively by integrating the set of variants for the selected component. We define the partial circuit set as:

Definition 2. *(Partial circuit set) The set of partial circuits $PC(c_i)$ for the selected component c_i, where $i \in \{1, ..., n\}$ for n components in the DAG, has elements constructed using the set of partial circuits for c_{i-1} (i.e., $PC(c_{i-1})$) with the variant set of c_i (i.e., $V(c_i)$), where c_{i-1} is the parent component of component c_i in the DAG. That is,*

$$PC(c_i) = \cup_{z \in P}(PC_x \oplus V_y) \quad (1)$$
$$\forall PC_x \in PC(c_{i-1}), \text{ and } \forall V_y \in V(c_i)$$

where PC_x is a partial circuit in $PC(c_{i-1})$, V_y is a variant of component c_i in $V(c_i)$, $\oplus$ stands for both component integration and partial circuit combination, and P is the set of all anchor points in PC_x.

To illustrate the partial circuit set constructed in each iteration, consider the component DAG in Fig. 7(a) which includes $n = 4$ components. In the first iteration, the independent component S_0 is selected and all its valid variants are generated as $V(S_0)$. Since S_0 lacks parent components, partial circuit set $PC(S_0)$ is generated using the initial partial circuit set $\emptyset$ and $V(S_0)$. Similarly, in the second iteration, $PC(S_1)$ is generated using $\emptyset$ and $V(S_1)$. In the third iteration,

---

[1]The component dependency is inherited from the gate dependency in the original logic circuit. Capturing the gate dependency has been substantially explored by prior quantum compilers [28, 29, 33].
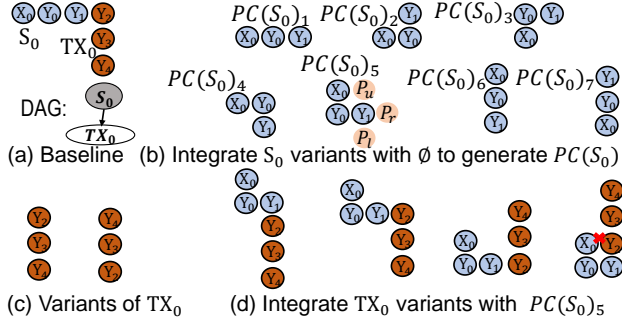
Figure 8. Example of component integration.

Figure 9. Example of partial circuits combination.

Figure 10. Example of wire insertion.

$PC(S_2)$ is generated by integrating $PC(S_0)$ and $V(S_2)$ since S_2 has a parent component S_0 in the DAG. In the fourth iteration, $PC(TX_0)$ is generated by integrating $PC(S_2)$, $PC(S_1)$, and $V(TX_0)$ according to Equation 1.

With the definitions of variant and partial circuits, we next elaborate on how FMCC constructs an MBQC circuit. At each iteration, one independent component is selected and removed from the component DAG. Based on the position of the component in the DAG, FMCC performs either component integration or partial circuit combination. Following this, wires may be inserted.

Component integration. At the beginning, FMCC starts with the initial partial circuit set as $\emptyset$ and selects the first independent component from the DAG. If there are multiple independent components, FMCC randomly chooses one of them. Fig. 8(a) shows an example DAG with two components S_0 and TX_0. Since S_0 is independent, FMCC chooses S_0 as the first component. Next, FMCC generates the variant set of S_0, i.e. $V(S_0)$, which contains all the variants of S_0, as shown in Fig. 8(b). Based on equation 1, the partial circuit set $PC(S_0)$ is constructed by adding $V(S_0)$ to $\emptyset$, the initial partial circuit set. Next, TX_0 becomes independent as S_0 is removed from the DAG in the previous iteration. FMCC chooses TX_0 and generates its variants set $V(TX_0)$ as shown in Fig. 8(c). Then, FMCC performs component integration using the partial circuit set $PC(S_0)$ and the variants set $V(TX_0)$. According to equation 1, all the anchor points of each partial circuit PC_x in $PC(S_0)$ are explored with each variant V_y in $V(TX_0)$. Take as an example the 5th partial circuit ($PC(S_0)_5$ in Fig. 8(b)) and the both variants of TX_0, Fig. 8(b) shows the three anchor points ($P = \{P_u, P_r, P_l\}$) of PC_5. FMCC integrates the TX_0 variants to each of the anchor points, generating partial circuits shown in Fig. 8(d). This results in the first three variants being valid, while the fourth has an invalid connection. FMCC constructs the partial circuit $PC(TX_0)$ by exploring all the partial circuits in $PC(S_0)$ and all the variants in $V(TX_0)$.

Partial circuit combination. Selected components can have more than one parent, which additionally requires a combination. For example, in Fig. 9(a), TX_0 has two parent
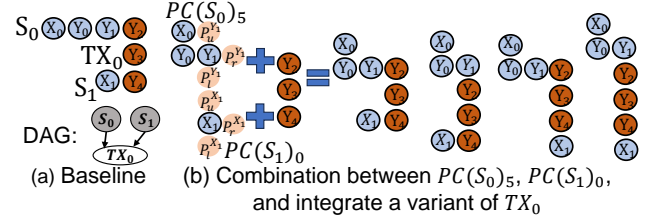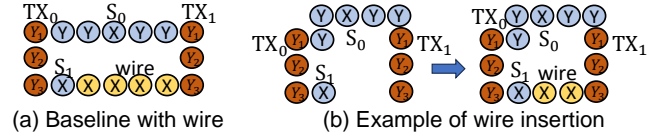
components (S_0 and S_1). In this case, partial circuit combination of the two dependent sets $PC(S_0)$ and $PC(S_1)$ is required before TX_0 can be integrated. The combination follows an all-to-all manner. That is, each partial circuit from $PC(S_0)$ is combined with each partial circuit from $PC(S_1)$ in combination. Fig. 9(b) illustrates this using $PC(S_0)_5$ from $PC(S_0)$, the only partial circuit $PC(S_1)_0$ in $PC(S_1)$, and the first variant of TX_0 to generate four new partial circuits by exploring the anchor points $P = \{(P_r^{Y_1}, P_r^{X1}), (P_l^{Y_1}, P_r^{X1}), (P_r^{Y_1}, P_u^{X1}), (P_l^{Y_1}, P_u^{X1})\}$. Note that other anchor point combinations in $PC(S_0)_5$ and $PC(S_1)_0$ are not possible. Consequently, $PC(TX_0)$ is formed by combining the two sets of partial circuits and integrating the selected component while exploring all anchor points.

Wire insertion. As discussed in Section 2.1, the function of a wire component is to connect a dependent component to its parent. Also, recall from Section 4.1, a wire can have valid variants with different even numbers X measurements. FMCC inserts wires on demand with appropriate wire variants to connect the dependent components whenever necessary. For example, Fig. 10(a) shows a baseline MBQC circuit. Consider a valid partial circuit generated from the previous iteration in Fig. 10(b). FMCC inserts a wire (two X measurements) to connect TX_1 to its parent S_1.

5.2 Partial Circuits Pruning

The partial circuit construction involves exploring all variants and all anchor points. It is difficult to track all the partial circuits as the design space increases exponentially with the number of components. Fortunately, many of the partial circuits are invalid due to violation of the constraints (Section 4.2), such as the example in Fig. 8(d) where there is an invalid connection between X_0 and Y_2. FMCC drops all the invalid partial circuits. However, FMCC may still generate so many valid partial circuits that it is impractical to track them all. Therefore, to reduce the design space,
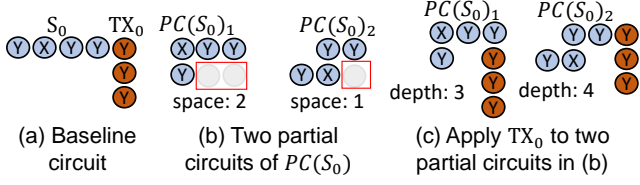
Figure 11. Example of leveraging space.

FMCC employs a threshold m to prune the set of partial circuits in each iteration. Specifically, for a partial circuit set $PC(c)$, we divide it into multiple sets where each set contains the partial circuits with the same width (i.e., $PC(c) = PC(c)^{w_1} \cup PC(c)^{w_2} \cup \cdots \cup PC(c)^{w_q}$, where $w_1, w_2, \cdots, w_q$ are the partial circuit width). For the partial circuit set of each width $PC(c)^{w_i}$, we retain the top m partial circuits and drop the others to reduce the size of $PC(c)$. We choose different widths because it is more likely to produce a valid MBQC circuit with reduced depth. As discussed in Section 4, an MBQC circuit can reduce its depth by using more photon rows. Similarly, the depth of a partial circuit can be decreased by utilizing more rows (wider width). However, keeping only the partial circuits with a larger width may lead to an MBQC circuit width that exceeds the cluster state width. On the other hand, retaining partial circuits with a smaller width reduces this risk but may result in a deeper MBQC circuit. As a result, we retain m partial circuits in $PC(c)^{w_i}$ for each width w_i to achieve a valid and shorter MBQC circuit.

We next answer the question: *which m partial circuits to retain?* We leverage two metrics: i) *depth* and ii) *space*. Since the depth is the objective of FMCC, a partial circuit with a shorter depth is preferred as it is more likely to generate a shorter MBQC circuit. If multiple partial circuits have the same width and depth, we use space to determine which m partial circuits to retain. Particularly, space is defined as the number of unused photons after each rightmost measurement in a partial circuit. For example, Fig. 11(a) shows a circuit with two components S_0 and TX_0. The space is annotated in Fig. 11(b) for two partial circuits of $PC(S_0)$. The unused photons in the space may be used by the next component integration. Therefore, a larger space indicates a larger potential to generate short depth when integrating the next component. We use the example in Fig. 11(c) to illustrate this. If we choose the $PC(S_0)_1$ with 2 spaces, we can generate the partial circuit with depth 3 after integrating TX_0. In contrast, if we choose $PC(S_0)_2$ with one space, we will have the partial circuit with depth 4 after integrating TX_0. In FMCC, the width, depth, and space are tracked using a table where each row of the table is a partial circuit.

5.3 Detailed Algorithm of FMCC

The detailed algorithm of FMCC is shown in Algorithm 1. The input is the component DAG G, a cluster state, and

the parameter m for retaining partial circuits. The output is the MBQC circuit with minimal depth. In each iteration, FMCC selects and removes an independent component c from the component DAG (lines 2-3). FMCC then generates all its variants as a variant set $V(c)$ (line 4) and identifies its parent components (line 5). If c has less than two parent components, FMCC conducts component integration that integrates each variant from $V(c)$ with each partial circuit PC_x in $PC(c_{parent1})$ where $c_{parent1}$ is the parent component of c, while considering all anchor points P (lines 8-14). When c has two parent components, FMCC performs partial circuit combination of $PC(c_{parent1})$ and $PC(c_{parent2})$, and then integrates variants in $V(c)$ while exploring all the anchor points (lines 15-24). In this process, for each $PC(c)_x$ in $PC(c)$, FMCC inserts a wire if necessary (lines 13 and 23) and removes invalid $PC(c)_x$ (lines 14 and 24). Also, FMCC maintains a table to record the width, depth, and space for each partial circuit $PC(c)_x$ in $PC(c)$ (line 25), and uses the table to retain the m partial circuits with the same width (line 26).

Algorithm 1 FMCC

Input: Component DAG G, a cluster state CS, m
Output: Minimized-depth MBQC circuit
    Initialization: Partial circuits sets $PC = []$
1: **while** $G \mathrel{!=} []$ **do**
2:    $c = independent(G)$
3:    $G.remove(c)$
4:    $V(c) = Generate\_variants(c)$
5:    $c_{parent1}, c_{parent2} = Find\_parent(c)$
6:    $PC(c) = []$
7:    **for** V_y in $V(c)$ **do**
8:        **if** $c_{parent2} == \emptyset$ **then**
9:            **for** PC_x in $PC(c_{parent1})$ **do**
10:              $P = Find\_Anchor\_Points(PC_x)$
11:              **for** P_1 in P **do**
12:                  $PC(c)_x = integrate(V_y, PC_x, P_1)$
13:                  $PC(c)_x = insert\_wire(PC(c)_x)$
14:                  $PC(c).append(valid(PC(c)_x))$
15:        **else if** $C_{parent2} \mathrel{!=} \emptyset$ **then**
16:            **for** PC_{x1} in $PC(c_{parent1})$ **do**
17:              **for** PC_{x2} in $PC(c_{parent2})$ **do**
18:                  $P^{c_{parent1}} = Find\_Anchor\_Points(PC_{x1})$
19:                  $P^{c_{parent2}} = Find\_Anchor\_Points(PC_{x2})$
20:                  **for** P_1 in $P^{c_{parent1}}$ **do**
21:                    **for** P_2 in $P^{c_{parent2}}$ **do**
22:                      $PC(c)_x = combine(V_y, PC_{x1}, PC_{X2}, P_1, P_2)$
23:                      $PC(c)_x = insert\_wire(PC(c)_x)$
24:                      $PC(c).append(valid(PC(c)_x))$
25:    $Table = create\_table(PC(c))$
26:    $PC = PC + select\_top\_m(PC(c), Table, m, CS))$
27: **return** $shortest(PC[-1])$

5.4 A Walk-Through Example

We now go through a real benchmark (i.e., 3-qubit Bernstein–Vazirani — bv3) to illustrate the steps of FMCC in Fig. 12. The baseline MBQC circuit of bv3 and its component DAG are shown in Fig. 12(a). We assume the cluster state width w is 8 and the retained partial circuit count m is 2. There are a total of 12 iterations due to 12 non-wire components in the DAG. Due to the lack of space, Fig. 12 only shows five iterations, which are sufficient to illustrate the steps of FMCC. In each iteration, we plot the table of the width, depth,
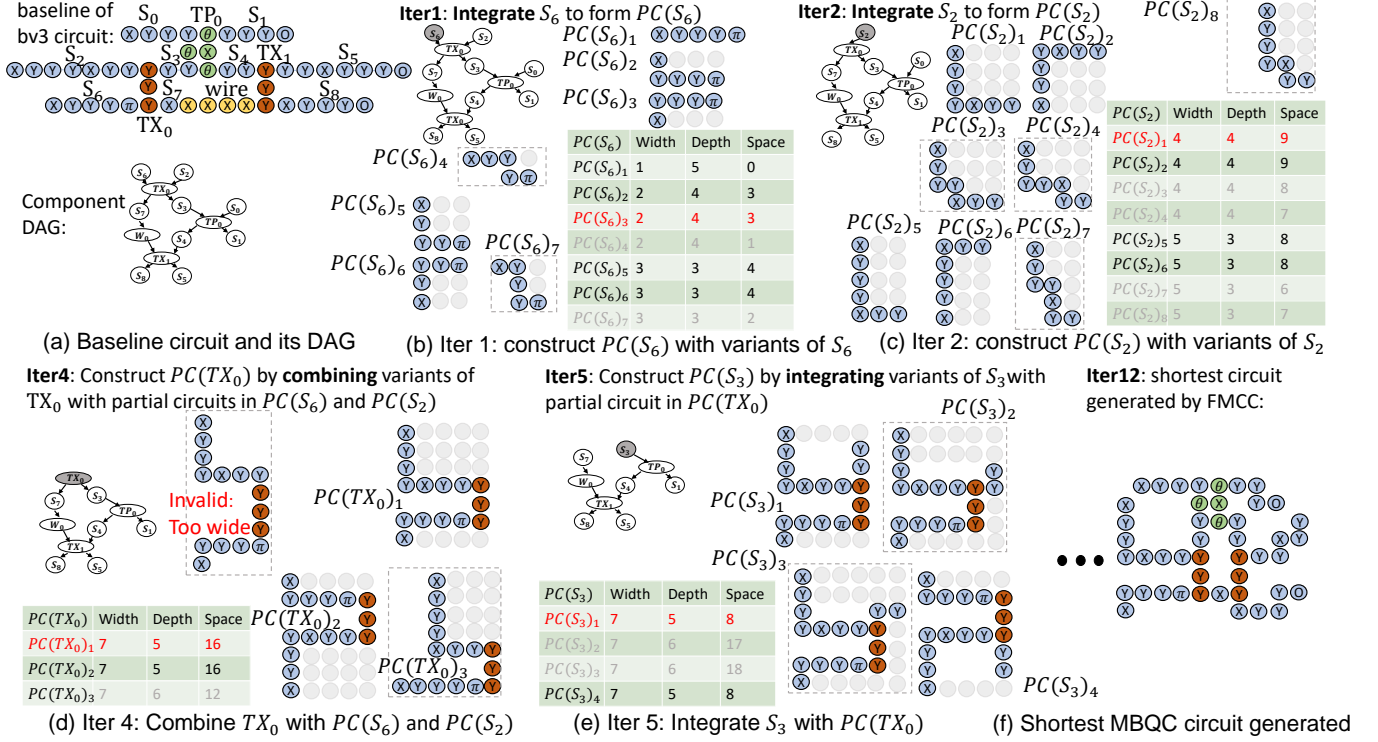
Figure 12. Example of FMCC Constructing circuit variant.

and space of each partial circuit. We also highlight the chosen component in each iteration and show the remaining components in the DAG. For each partial circuit, photons contributing to space are indicated with gray-shaded circles.

Initially, there are three independent components: S_6, S_2, and S_0. Thus, in the first three iterations, FMCC conducts the component integration for each independent component and generates $PC(S_6)$, $PC(S_2)$, and $PC(S_0)$. Fig. 12(b) shows the first iteration where S_6 is selected to construct $PC(S_6)$. There are more than 30 variants of S_6 and we show representative seven of them in Fig. 12(b), labeled from $PC(S_6)_1$ to $PC(S_6)_7$. Among the seven partial circuits, one has a width of 1, three have a width of 2, and three have a width of 3. Since m is 2, FMCC will retain the only variant with width 1, two out of three variants with width 2, and two out of three variants with width 3. For $PC(S_6)_2$, $PC(S_6)_3$, and $PC(S_6)_4$ that have the same width 2, they also have the same depth. Therefore, the two with large spaces are retained, and the one with smaller space ($PC(S_6)_4$) is pruned (shaded in the table). Similarly, For $PC(S_6)_5$, $PC(S_6)_6$, and $PC(S_6)_7$ that have the same width of 3, they also have the same depth. $PC(S_6)_7$ is pruned as it has the smallest space. In iteration 2, the procedure is similar. We show eight representative partial circuits of $PC(S_2)$ ($PC(S_2)_1$ to $PC(S_2)_8$) in Fig. 12(c). Among the eight partial circuits, $PC(S_2)_1$ to $PC(S_2)_4$ have the same width of 4 and the same depth of 4. $PC(S_2)_3$ and $PC(S_2)_4$ are pruned due to their smaller space. Similarly, $PC(S_2)_5$ to

$PC(S_2)_8$ have the same width of 5 and the same depth of 3. $PC(S_2)_7$ and $PC(S_2)_8$ are pruned. In iteration 3, FMCC performs the same procedure to construct $PC(S_0)$.

In iteration 4, TX_0 is independent and selected. Since TX_0's parent components are in two partial circuit sets $PC(S_6)$ and $PC(S_2)$, FMCC performs combination of $PC(S_6)$ and $PC(S_2)$ and integrates $V(TX_0)$ with the combined partial circuits. We plot four example partial circuits generated after TX_0 integration in Fig. 12(d). The first one has a width of 9 which is larger than the fixed cluster state width of 8. Therefore, FMCC prunes it from $PC(TX_0)$. The other three partial circuits $PC(TX_0)_1$, $PC(TX_0)_2$, and $PC(TX_0)_3$ have the same width of 7. Since $PC(TX_0)_1$ and $PC(TX_0)_2$ have a shorter depth (5) compared to $PC(TX_0)_3$'s depth (6), they are retained in $PC(TX_0)$, and $PC(TX_0)_3$ is pruned.

In iteration 5, S_3 becomes independent and $V(S_3)$ is integrated with the partial circuits in $PC(TX_0)$. We show four generated partial circuits with the same width in Fig. 12(e). $PC(S_3)_1$ and $PC(S_3)_4$ are kept due to their shorter depth of 5. For the remaining iterations, similar integration and combination processes are conducted by FMCC to generate sets of partial circuits. Fig. 12(f) shows the MBQC circuit with the minimized depth after iteration 12.

5.5 Optimizing Multiple Rounds

The same quantum circuit usually executes multiple times to obtain stabilized output distribution [35]. This equivalently
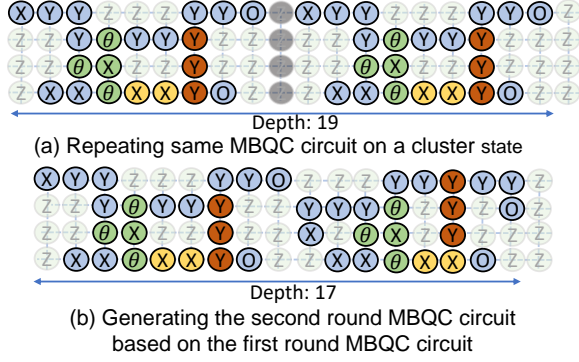
Depth: 19

(a) Repeating same MBQC circuit on a cluster state

Depth: 17

(b) Generating the second round MBQC circuit
based on the first round MBQC circuit

Figure 13. Executing two rounds of MBQC circuits.

maps to the same MBQC circuits executing multiple times, which we referred to as "rounds". Fig. 13(a) shows a quantum circuit that executes two rounds on a cluster state, where a column of photons is cut out to separate the two rounds, annotated by gray photons, which extends the overall circuit depth. Our proposed FMCC can be seamlessly applied to consecutive rounds of a quantum circuit. Specifically, for MBQC circuits with the same width of a round, FMCC keeps track of the top m for that round and applies the same procedure (discussed in Section 5.1 and Section 5.2) when picking up the components from the next round. This allows us to utilize the cut-out photons at the end of each MBQC round. For instance, Fig. 13(b) shows a valid two-round execution where the second round does not have any connection with the first round, but is able to utilize the cut-out photons and reduce the depth. Note that, the second round has a different valid MBQC circuit compared to the first round.

5.6 Complexity of FMCC

We next discuss the complexity of FMCC. Let us assume there are N_C components in the component DAG G of a given circuit. At each iteration of Algorithm 1, FMCC picks one component c from G and generates a set of valid variants $V(c)$ in the size of N_V (lines 2-3). Each variant is either i) integrated with a partial circuit in a partial circuit set (lines 8-14) or ii) combined with two partial circuits that are chosen from different partial circuit sets (lines 15-24). Recall from Section 5.2, FMCC keeps the top m partial circuits for each width. Suppose there are w different widths in a partial circuit set; hence, there are at most mw partial circuits. Additionally, assuming there are a anchor points. Therefore, the worst-case complexity for integration is $O(N_C \cdot N_V \cdot a \cdot m \cdot w)$. Similarly, for combination, as there are two partial circuit sets with each having mw partial circuits, the worst-case complexity is $O(N_C \cdot N_V \cdot (amw)^2)$.

Note that, for the evaluated benchmarks, we observe that most of the components (more than 90%) are integrated with a single partial circuit set. Only a few components (less than 10%) are combined with two partial circuit sets. We also

observed that the average number of widths in a partial circuit set is $w = 5.8$, the average number of variants is $N_V = 5.4$, and the average number of anchor points is $a = 2$. These parameters do not vary when we increase the number of qubits (from 5 to 27 in our experiments). The reason is twofold. First, the components used in FMCC have a finer granularity compared to gates. Most of the gates share the same components as basic building "primers". Second, the number of variants of components does not change with the number of gates. Since the structure (i.e., pattern) of a component is simpler than gate patterns, there are only a few variants for each component. As a result, the complexity of FMCC is $O(N_C \cdot N_V \cdot a \cdot m \cdot w) = O(62.6N_Cm) = O(N_Cm)$ for integration. Since combination only involves TP or TX components with variants number $N_V = 2$, the complexity is $O(N_C \cdot N_V \cdot (amw)^2) = O(269.1N_Cm^2) = O(N_Cm^2)$. The number of components N_c increases with the number of gates and the number of qubits in a quantum circuit (details given in Section 6.7). The parameter m (i.e., the number of partial circuits to retain at each iteration) is set to 12 in all our evaluations as the depth reduction saturates (sensitivity study of m is given in Section 6.6). This further reduces FMCC's execution time.

5.7 FMCC-nz

While the primary optimization target for FMCC is to reduce circuit depth, it does not guarantee minimizing the number of non-Z measurements. Specifically, in some circuits, FMCC reduces the number of wires so that the total number of non-Z measurements can be reduced. In other circuits, FMCC might introduce more wires (e.g., for aligning the partial circuits to perform component combination), which can consequently increase the Pauli error (introduced in Section 3). To this end, we propose FMCC-nz to minimize the number of non-Z measurements. Instead of prioritizing optimizing depth, FMCC-nz sets the number of non-Z measurements as the primary optimization objective and depth as the secondary optimization objective. That is, for partial circuits with the same width, FMCC-nz keeps the m partial circuits with the fewest non-Z measurements. If more than m partial circuits have the same minimum number of non-Z measurements, FMCC-nz then uses depth to determine which m partial circuits to retain in each iteration.

6 Evaluation

6.1 Experiment Setup

Benchmarks and Metric: We use six representative quantum applications (shown in Table 1) to evaluate FMCC: Quantum Fourier Transform (qft) [11], hardware efficient ansatz (hwea) [34], Bernstein–Vazirani (bv) [5], instantaneous quantum polynomial time (iqp) [6], linear hydrogen atom chain (hc) [45], and 3-regular Quantum Approximate Optimization Algorithm(qaoa) [16]. For each application, we evaluate

three different numbers of qubits: 5 qubits (*small*), 15 qubits (*medium*), and 27 qubits (*large*). Note that, hc and qaoa require an even number of qubits [16, 45]. Therefore, we use 6, 14, and 26 qubits as the *small*, *medium*, and *large* for hc and qaoa, respectively. We use the cluster state depth reduction ratio as the metric to show the effectiveness of our approach.

Baseline: The baseline employs the state-of-the-art MBQC compilation on cluster state [29]. This baseline optimizes the depth of the MBQC circuit by leveraging gate reordering, redundant wire removal, and consecutive single-qubit gate combinations. We use this baseline to demonstrate the depth reduction brought by FMCC over the state-of-the-art.

Table 1. Number of qubits and width for each benchmark.

Appli-	*small*			*medium*			*large*		
cation	*qubit*	*C1*	*C2*	*qubit*	*C1*	*c2*	*qubit*	*C1*	*C2*
bv	5	12	14	15	37	44	27	67	80
iqp	5	12	14	15	37	44	27	67	80
hwea	5	12	14	15	37	44	27	67	80
qft	5	12	14	15	37	44	27	67	80
hc	6	14	17	14	34	41	26	64	77
qaoa	6	14	17	14	34	41	26	64	77

Fixed cluster state width: As discussed in Section 2.2, the minimum cluster state width for a N_q-qubit benchmark is $2N_q - 1$. We use two different fixed cluster state widths to evaluate FMCC: i) $1.25 \times (2N_q - 1)$, referred to as *C1*, and ii) $1.5 \times (2N_q - 1)$, referred to as *C2*.

As discussed in Section 5.2, FMCC retains m partial circuits for each width. For the main results, we empirically choose m to 12, which we find is enough to provide the minimized depth (sensitivity study of m is given in Section 6.6).

6.2 Single Round Results

We show the depth reduction ratios achieved by FMCC of a single round in Fig. 14 and multiple rounds (i.e., 100 rounds) in Fig. 16. Fig. 14 and Fig. 16 also contain results under different fixed cluster widths (i.e., C1 and C2). As shown in Fig. 14(a), FMCC achieves an average of 31.9%, 27.6%, and 17.0% depth reductions for *small*, *medium*, and *large* under *C1* width, respectively. Similarly, under *C2* width, FMCC achieves an average of 37.1%, 31.7%, and 27.8% depth reductions, respectively. For all three numbers of qubits, the reductions increase from *C1* to *C2*, indicating that FMCC effectively leverages the redundant photons rows in wider cluster states to reduce the depth.

The depth reduction ratio decreases with more number of qubits for bv, iqp, hwea, and qaoa. To illustrate the reason, let us consider 3-qubit bv in Fig. 15(a) and 5-qubit bv in Fig. 15(b). In the figure, we annotate the layers of the single-qubit gate using red boxes and the layers of the two-qubit gate using green dash boxes. A layer is a collection of independent gates that can be performed concurrently. As one can observe, the number of two-qubit gate layers increases with the number
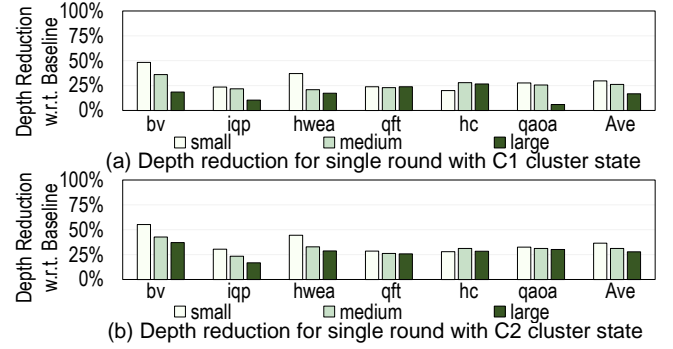

(a) Depth reduction for single round with C1 cluster state

(b) Depth reduction for single round with C2 cluster state

Figure 14. Depth reduction for a single round.
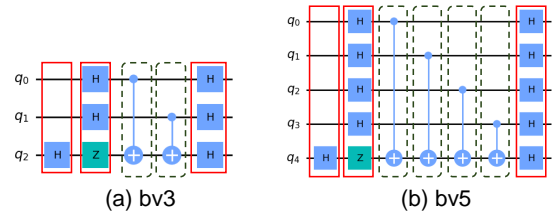

(a) bv3   (b) bv5

Figure 15. Different qubit number bv and the gate layers.

of qubits. As such, there are more dependent TX components in the component DAG. For instance, Fig. 15(a) has two layers of two-qubit gates, and Fig. 15(b) has four layers of two-qubit gates. Recall that in Section 4, the TX and TP components have fewer variants and less reduction in depth compared to the S components. Therefore, with more TX components in large qubit counts, the depth reduction brought by FMCC also decreases. Similar results exist in the iqp, hwea, and qaoa. With a larger qubit size, iqp has more TP components while hwea and qaoa have more TX components.

In the hc, the depth reduction increases from *small* to *medium* qubit count, and then decreases from *medium* to *large* qubit count. This is because hc has long S components where many of its variants are invalid in the *small* qubit count since it exceeds the fixed cluster state width (14 in *C1* and 17 in *C2*). In the *medium* qubit count, the cluster state is wider, so more valid S components can be accommodated. However, in the *large* qubit count, the depth reduction decreases because hc also has an increased number of two-qubit gate layers as bv, iqp, hwea, and qaoa. qft exhibit similar depth reduction in all qubit counts. This is because the number of S components is proportionally increased with more qubits, providing a constant potential for depth reduction.

6.3 Multiple Round Results

We evaluate one hundred rounds of each benchmark in Fig. 16. As one can observe, for *C1*, FMCC achieves average of 42.2%, 46.5%, and 42.4% depth reductions for *small*, *medium*, and *large*, respectively. The depth reductions are 51.7%, 57.4%, and 56.8% on *C2*. Compared to the single-round
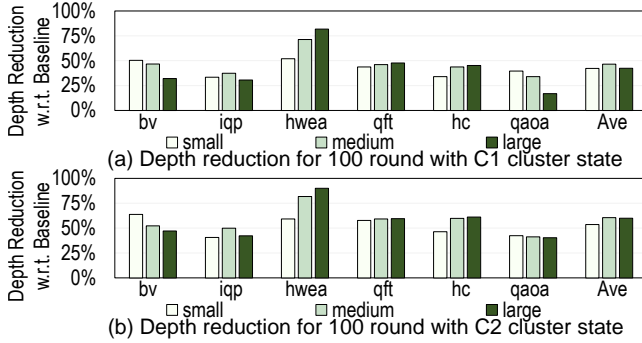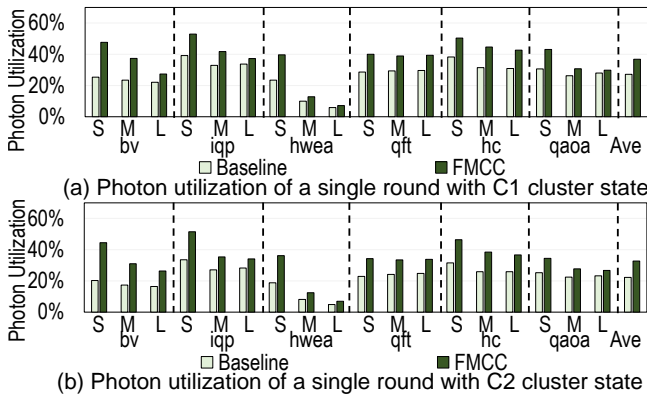
Figure 16. Depth reduction for 100 rounds.

Figure 17. Photon utilization for one round.

Figure 18. Photon utilization for 100 rounds.

results in Fig. 14, FMCC achieves higher depth reduction, indicating its effectiveness in utilizing the cut-out photons between consecutive rounds as we discussed in Section 5.5.

For the bv and qaoa, the trend of depth reduction across the three qubit counts is similar to the trend observed in the single round results (Fig. 14), indicating that FMCC effectively exploits both single round depth reduction and multi-round depth reduction. For iqp, the depth reduction increases from *small* to *medium* qubit count due to the effectiveness of FMCC multi-round optimization. However, the depth reduction drops from *medium* to *large* qubit count due to the increasing number of two-qubit gate layers (the same as the iqp single round case). In contrast, the depth reduction of hwea, qft, and hc increase with more qubits. This is because with more number of qubits, the cut-out photons at the end of a round (i.e., after the last component in a round is integrated) increase. Recall our discussion in Section 5.5 where the cut-out photons can be leveraged by consecutive rounds. Therefore, with more cut-out photons in the large qubit count, the depth reduction is also larger.

6.4 Photon Utilization

We plot the photon utilization of the single round in Fig. 17 and the multi-round in Fig. 18. Photon utilization is defined as the ratio of non-z measurements in a cluster state to all the
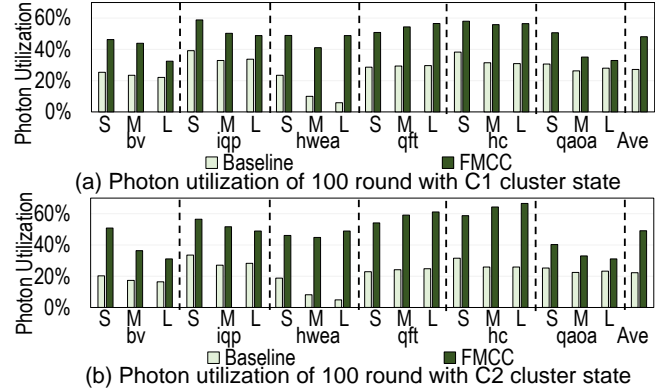
number of measurements in the cluster state. FMCC yields significant improvement in photon utilization compared to the baseline. On average, FMCC achieves the photon utilization of 36.8%, 32.7%, 48.1%, and 49.1% in single round C1, C2, and multi-round C1, C2, respectively. This is significantly higher compared to the baseline (27.1%, 22.2%, 27.1%, and 22.3% in single-round C1, C2, and multi-round C1, C2, respectively), indicating that FMCC effectively exploits the cutout photons to reduce the MBQC circuit depth.

6.5 Error Evaluation

We report the error evaluation according to the error models discussed in Section 3. Due to the lack of space, we report the results for 100 rounds of *large* benchmarks under *C2* width. Other experimental configurations have similar trends.

Table 2 shows the results for baseline, FMCC, and FMCC-nz. i) For emitter decoherence, we report the circuit depth and the improvement in fidelity. Specifically, the average fidelity improvements are 59.4% and 58.0% (from the last two rows in Table 2) for FMCC and FMCC-nz, respectively. These improvements are yielded by effectively reducing the cluster state depth. FMCC has a better fidelity improvement over FMCC-nz as it minimizes the depth compared to FMCC-nz which gives priority to minimizing non-Z measurements (Section 5.7). ii) For fusion failure, we report the number of fusion operations and the percentage of fusion reduction. Recall that the number of fusion operations required to generate a cluster state is proportional to the cluster state width and depth. As both FMCC and FMCC-nz employ fixed cluster state width, the reduction of fusion operations has a similar trend to the depth reduction. For instance, hwea has more depth reduction, so it has more fusion reduction compared to qaoa, which has less depth reduction and fusion reduction. On average, FMCC and FMCC-nz attain a 56.8% and a 55.5% fusion reduction, respectively. iii) For photon loss, we report the total number of measurements and the percentage of measurements reduction. Similar to the number of fusion

Table 2. Error evaluation of FMCC and FMCC-nz.

Appli-cation	Method	Decoherence		Fusion		Photon Loss		Pauli Error	
		Depth	Fidelity improvement	#fusions	#fusions reduction	#measurements	#measurements reduction	#non-Z measurements	#non-Z measurements reduction
bv	Baseline	13,499		4,278,943		1,079,920		199,400	
	FMCC	7,218	33.0%	2,287,866	46.5%	577,440	46.5%	151,600	24.0%
	FMCC-nz	7,380	32.1%	2,339,092	45.4%	590,368	45.3%	150,200	24.7%
iqp	Baseline	17,899		5,673,743		1,431,920		413,800	
	FMCC	9,814	44.4%	3,110,798	45.2%	785,120	45.2%	400,700	3.2%
	FMCC-nz	10,082	42.7%	3,195,190	43.7%	806,576	43.7%	392,300	5.2%
hwea	Baseline	11,499		3,644,943		919,920		45,000	
	FMCC	1,224	59.5%	387,768	89.4%	97,920	89.4%	45,000	0%
	FMCC-nz	1,249	59.3%	395,563	89.1%	99,887	89.1%	45,000	0%
qft	Baseline	10,899		3,454,743		871,920		216,200	
	FMCC	4,466	34.0%	1,415,561	59.0%	357,300	59.0%	216,200	0%
	FMCC-nz	4,654	32.8%	1,475,066	57.3%	372,317	57.3%	216,200	0%
hc	Baseline	10,899		3,454,743		871,920		225,800	
	FMCC	4,284	35.1%	1,357,683	60.7%	342,694	60.7%	225,800	0%
	FMCC-nz	4,562	33.4%	1,446,031	58.1%	364,990	58.1%	225,800	0%
qaoa	Baseline	50,299		1,5340,964		3,873,023		899,700	
	FMCC	30,085	150.6%	9,175,694	40.2%	2,316,545	40.2%	720,100	20.0%
	FMCC-nz	30,352	147.6%	9,257,163	39.6%	2,337,113	39.6%	716,100	20.4%
Ave	**FMCC**	**9,515**	**59.4%**	**2,955,895**	**56.8%**	**746,170**	**56.8%**	**293,233**	**7.8%**
	FMCC-nz	**9,713**	**58.0%**	**3,018,122**	**55.5%**	**761,875**	**55.5%**	**290,933**	**8.4%**

operations, reducing the cluster state depth reduces the total number of measurements. Therefore, FMCC achieves a 56.8% total measurement reduction while FMCC-nz achieves a 55.5% total measurement reduction. iv) For Pauli error, we report the number of non-Z measurements and the percentage of non-Z measurements reduction. As shown in Table 2, the average number of non-Z measurements is reduced by 7.8% and 8.4% using FMCC and FMCC-nz, respectively. FMCC-nz achieves a better non-Z measurement reduction than FMCC because it prioritizes non-Z measurement reduction over depth reduction. To be more concrete, we observed that the number of non-Z measurements in both FMCC and FMCC-nz is reduced for bv, iqp, and qaoa while the number of non-Z measurements remains the same for qft, hwea, and hc. This is because there is no wire in qft and hwea, while hc only has wires with two X measurements (i.e., the shortest possible wires necessary for connection). Eliminating wires in hc results in invalid connections introduced in Section 4.2. bv and qaoa have higher non-Z measurement reduction compared to iqp, as they have more and longer wires than iqp.

FMCC versus FMCC-nz. Based on the results in Table 2, it is clear that FMCC achieves a better depth reduction, which in turn mitigates the error related to emitter decoherence, fusion failure, and photon loss. In contrast, FMCC-nz prioritizes the reduction of non-Z measurements over depth. While its Pauli error probability is lower than that of FMCC, it may have higher errors in other types. This is a key tradeoff design in FMCC based on different error mitigation targets.
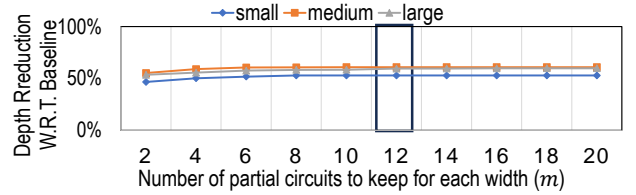
Figure 19. Depth reduction with various of m.

6.6 Sensitivity Study

Large number of qubits. We evaluate bv and qft using 50, 100, and 125 qubits to study the scalability of FMCC using *C2* fixed cluster state width. Overall, FMCC effectively reduces the circuit depth by 43.3% and 59.8% for bv-50 and qft-50. The depth reductions are 42.1% and 60.0% for bv-100 and qft-100, and the depth reductions are 41.4% and 59.4% for bv-125 and qft-125. For bv, the depth reduction ratio slightly drops with a larger qubit count due to the increased layers of the two-qubit gate (discussed in Section 6.2). In contrast, qft has a similar depth reduction in 50, 100, and 125 qubits due to the proportionally increased S components to the number of qubits [11] (discussed in Section 6.2).

Number of partial circuits (m). Recall that we empirically set the number of partial circuits (i.e., m) to 12 for each cluster width in all previously reported results. A large m enhances the likelihood of finding a shorter cluster state depth, whereas a smaller m has a lower searching cost due to the pruned search space. Fig. 19 shows the averaged circuit depth reduction ratio when varying m from 2 to 20 for *small*, *medium*, and *large* qubit counts. One can observe that
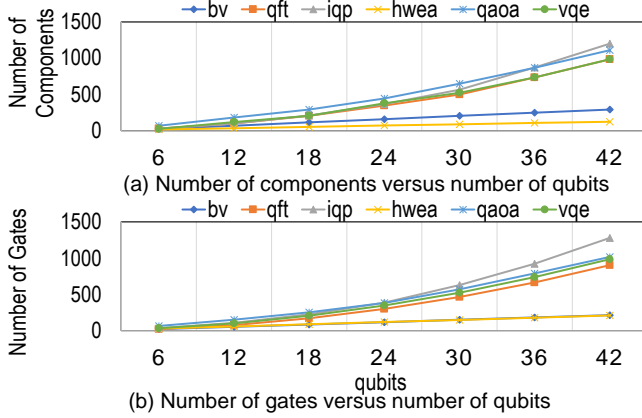
Figure 20. Scalability of FMCC.

the depth reduction ratios saturated at 8 for the *small* and *medium*, while for the *large*, it saturated at 12.

6.7 Scalability

Recall from Section 5.6, the complexity of FMCC is linear to the number of components. We plot the number of components versus the number of qubits in Fig. 20(a) to evaluate the scalability of FMCC. The number of components is linear with the number of qubits in bv and hwea while the number of components is quadratic with the number of qubits in qft, iqp, qaoa, and vqe. This is because the number of gates (including SWAP gates) also increases linearly in bv and hwea and quadratic in qft, iqp, qaoa, and vqe when the number of qubit increases (shown in Fig. 20(b)), implying that the complexity of FMCC depends on the applications and is proportional to the number of gates in the application.

7 Related Works

There exist many compilation frameworks for superconducting and trapped ion quantum computers [1, 13, 14, 22, 26, 28, 31, 32]. Recently, some pioneering steps have been taken to improve the fidelity or performance of MBQC through MBQC compilations [49–51]. These works aim to reduce the impact of fusion failures in the graph state, which is a multi-qubit state represented by a graph with a flexible geometry [4]. For example, Oneq [50] proposes three optimization modules to reduce the number of fusion operations to generate a graph state while OnePerc [49] proposes a real-time compilation method to construct graph states in the presence of fusion failures. However, these works only focus on reducing the impact of fusion failures, while our work aims to reduce the depth of a cluster state. They are not applicable for reducing cluster state depth.

8 Conclusion

In this paper, we propose FMCC, which is flexible MBQC computation on cluster state with the goal of minimizing

the cluster state depth. The proposed FMCC leverages both intra-component flexibility and inter-component flexibility to effectively utilize the redundant photons and rows in a fixed-width cluster state. Specifically, FMCC employs dynamic programming with heuristics to reduce the huge design space searching while avoiding sub-optimal results using simple heuristics. Experimental results using six representative quantum applications have shown that, compared to the state-of-the-art MBQC compilation, FMCC achieves 51.7%, 57.4%, and 56.8% average depth reductions in small, medium, and large qubit counts, respectively.

Acknowledgments

References

[1] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Circuit compilation methodologies for quantum approximate optimization algorithm. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 215–228, 2020.

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[3] Sean D Barrett and Thomas M Stace. Fault tolerant quantum computation with very high threshold for loss errors. *Physical review letters*, 105(20):200502, 2010.

[4] Sara Bartolucci, Patrick Birchall, Hector Bombin, Hugo Cable, Chris Dawson, Mercedes Gimeno-Segovia, Eric Johnston, Konrad Kieling, Naomi Nickerson, Mihir Pant, et al. Fusion-based quantum computation. *Nature Communications*, 14(1):912, 2023.

[5] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2126):459–472, 2011.

[6] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2126):459–472, 2011.

[7] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009.

[8] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[9] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.

[10] Dan Cogan, Zu-En Su, Oded Kenneth, and David Gershoni. Deterministic generation of indistinguishable photons in a cluster state. *Nature Photonics*, 17(4):324–329, 2023.

[11] Don Coppersmith. An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002.

[12] Christopher M Dawson, Henry L Haselgrove, and Michael A Nielsen. Noise thresholds for optical quantum computers. *Physical review*

letters, 96(2):020501, 2006.

[13] Y. Ding, P. Gokhale, S. Lin, R. Rines, T. Propson, and F. T. Chong. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 201–214, Los Alamitos, CA, USA, oct 2020. IEEE Computer Society.

[14] Casey Duckering, Jonathan M. Baker, Andrew Litteken, and Frederic T. Chong. Orchestrated trios: Compiling for efficient communication in quantum programs with 3-qubit gates. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 375–385, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Sophia E Economou, Netanel Lindner, and Terry Rudolph. Optically generated 2-dimensional photonic cluster state from coupled quantum dots. *Physical review letters*, 105(9):093601, 2010.

[16] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[17] Vinicius S Ferreira, Gihwan Kim, Andreas Butler, Hannes Pichler, and Oskar Painter. Deterministic generation of multidimensional photonic cluster states with a single quantum emitter. *Nature Physics*, pages 1–6, 2024.

[18] Mercedes Gimeno-Segovia, Terry Rudolph, and Sophia E Economou. Deterministic generation of large-scale entangled photonic cluster state from interacting solid state emitters. *Physical review letters*, 123(7):070501, 2019.

[19] Mercedes Gimeno-Segovia, Pete Shadbolt, Dan E Browne, and Terry Rudolph. From three-photon greenberger-horne-zeilinger states to ballistic universal quantum computation. *Physical review letters*, 115(2):020502, 2015.

[20] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[21] Robert H Hadfield. Single-photon detectors for optical quantum information applications. *Nature photonics*, 3(12):696–705, 2009.

[22] Fei Hua, Yuwei Jin, Yanhao Chen, Suhas Vittal, Kevin Krsulich, Lev S. Bishop, John Lapeyre, Ali Javadi-Abhari, and Eddy Z. Zhang. Caqr: A compiler-assisted approach for qubit reuse through dynamic circuit. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 59–71, New York, NY, USA, 2023. Association for Computing Machinery.

[23] L Huthmacher, R Stockill, E Clarke, Maxime Hugues, C Le Gall, and M Atatüre. Coherence of a dynamically decoupled quantum-dot hole spin. *Physical Review B*, 97(24):241413, 2018.

[24] Richard Jozsa. An introduction to measurement based quantum computation. *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment*, 199:137–158, 2006.

[25] Pieter Kok, William J Munro, Kae Nemoto, Timothy C Ralph, Jonathan P Dowling, and Gerard J Milburn. Linear optical quantum computing with photonic qubits. *Reviews of modern physics*, 79(1):135, 2007.

[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] Mikkel V Larsen, Xueshi Guo, Casper R Breum, Jonas S Neergaard-Nielsen, and Ulrik L Andersen. Deterministic generation of a two-dimensional cluster state. *Science*, 366(6463):369–372, 2019.

[28] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming*

Languages and Operating Systems, pages 1001–1014, 2019.

[29] Yingheng Li, Aditya Pawar, Mohadeseh Azari, Guo Yanan, Youtao Zhang, Jun Yang, Kaushik Seshadreesan, and Xulong Tang. Orchestrating measurement-based quantum computation over photonic quantum processors. In *Proceedings of the 60th ACM/IEEE Design Automation Conference*, 2023.

[30] Netanel H Lindner and Terry Rudolph. Proposal for pulsed on-demand sources of photonic cluster state strings. *Physical review letters*, 103(11):113602, 2009.

[31] Andrew Litteken, Lennart Maximilian Seifert, Jason Chadwick, Natalia Nottingham, Frederic T. Chong, and Jonathan M. Baker. Qompress: Efficient compilation for ququarts exploiting partial and mixed radix operations for communication reduction. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 646–659, New York, NY, USA, 2023. Association for Computing Machinery.

[32] Andrew Litteken, Lennart Maximilian Seifert, Jason D. Chadwick, Natalia Nottingham, Tanay Roy, Ziqian Li, David Schuster, Frederic T. Chong, and Jonathan M. Baker. Dancing the quantum waltz: Compiling three-qubit gates on four level architectures. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[33] Ji Liu, Peiyi Li, and Huiyang Zhou. Not all swaps have the same cost: A case for optimization-aware qubit routing. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 709–725. IEEE, 2022.

[34] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

[35] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. *Phys. Today*, 54(2):60, 2001.

[36] Jeremy L O'brien. Optical quantum computing. *Science*, 318(5856):1567–1570, 2007.

[37] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. Photonic quantum technologies. *Nature Photonics*, 3(12):687–695, 2009.

[38] Hannes Pichler, Soonwon Choi, Peter Zoller, and Mikhail D Lukin. Universal photonic quantum computation via time-delayed feedback. *Proceedings of the National Academy of Sciences*, 114(43):11362–11367, 2017.

[39] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001.

[40] Robert Raussendorf, Daniel E Browne, and Hans J Briegel. Measurement-based quantum computation on cluster states. *Physical review A*, 68(2):022312, 2003.

[41] Robert Raussendorf and Tzu-Chieh Wei. Quantum computation by local measurement. *Annu. Rev. Condens. Matter Phys.*, 3(1):239–261, 2012.

[42] Adam Sawicki, Lorenzo Mattioli, and Zoltán Zimborás. Universality verification for a set of quantum gates. *Physical Review A*, 105(5):052602, 2022.

[43] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[44] Sergei Slussarenko and Geoff J Pryde. Photonic quantum information processing: A concise review. *Applied Physics Reviews*, 6(4):041303, 2019.

[45] Nicholas H Stair, Renke Huang, and Francesco A Evangelista. A multireference quantum krylov algorithm for strongly correlated electrons. *Journal of chemical theory and computation*, 16(4):2236–2245, 2020.

[46] Ravitej Uppu, Leonardo Midolo, Xiaoyan Zhou, Jacques Carolan, and Peter Lodahl. Quantum-dot-based deterministic photon–emitter interfaces for scalable photonic quantum technology. *Nature nanotechnology*, 16(12):1308–1317, 2021.

[47] Michael Varnava, Daniel E Browne, and Terry Rudolph. Loss tolerance in one-way quantum computation via counterfactual error correction. *Physical review letters*, 97(12):120501, 2006.

[48] Arian Vezvaee, Paul Hilaire, Matthew F Doty, and Sophia E Economou. Deterministic generation of entangled photonic cluster states from quantum dot molecules. *Physical Review Applied*, 18(6):L061003, 2022.

[49] Hezi Zhang, Jixuan Ruan, Hassan Shapourian, Ramana Rao Kompella, and Yufei Ding. Oneperc: A randomness-aware compiler for photonic quantum computing. *arXiv preprint arXiv:2403.01829*, 2024.

[50] Hezi Zhang, Anbang Wu, Yuke Wang, Gushu Li, Hassan Shapourian, Alireza Shabani, and Yufei Ding. Oneq: A compilation framework for photonic one-way quantum computation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–14, 2023.

[51] Felix Zilk, Korbinian Staudacher, Tobias Guggemos, Karl Fürlinger, Dieter Kranzlmüller, and Philip Walther. A compiler for universal photonic quantum computers. In *2022 IEEE/ACM Third International Workshop on Quantum Computing Software (QCS)*, pages 57–67. IEEE, 2022.