IEEE Transactions on Quantum Engineering

Received September 10, 2021; revised February 3, 2022; accepted May 9, 2022; date of publication May 19, 2022; date of current version June 22, 2022.

Digital Object Identifier 10.1109/TQE.2022.3175587

Timing Constraints Imposed by Classical Digital Control Systems on Photonic Implementations of Measurement-Based Quantum Computing

JOHN R. SCOTT[1] AND KRISHNA C. BALRAM[2]

[1]Quantum Engineering Centre for Doctoral Training, Department of Physics, University of Bristol, BS8 1UB Bristol, U.K.
[2]Quantum Engineering Technology Labs and Department of Electrical and Electronic Engineering, University of Bristol, BS8 1UB Bristol, U.K.

Corresponding author: John R. Scott (e-mail: john.scott@bristol.ac.uk).

The work of J. R. Scott was supported by the Bristol Quantum Engineering Centre for Doctoral Training, EPSRC under Grant EP/L015730/1. The work of K. C. Balram was supported by the European Research Council under Grant ERC-StG 758843. The design and testing system is available in the following repository: https://gitlab.com/johnrscott/mbqc-fpga

ABSTRACT Most of the architectural research on photonic implementations of measurement-based quantum computing (MBQC) has focused on the quantum resources involved in the problem with the implicit assumption that these will provide the main constraints on system scaling. However, the "flying-qubit" architecture of photonic MBQC requires specific timing constraints that need to be met by the classical control system. This classical control includes, for example, the amplification of the signals from single-photon detectors to voltage levels compatible with digital systems; the implementation of a control system which converts measurement outcomes into basis settings for measuring subsequent cluster qubits, in accordance with the quantum algorithm being implemented; and the digital-to-analog converter and amplifier systems required to set these measurement bases using a fast phase modulator. In this article, we analyze the digital system needed to implement arbitrary one-qubit rotations and controlled-NOT gates in discrete-variable photonic MBQC, in the presence of an ideal cluster state generator, with the main aim of understanding the timing constraints imposed by the digital logic on the analog system and quantum hardware. We have verified the design using functional simulations and have used static timing analysis of a Xilinx field-programmable gate array (7 series) to provide a practical upper bound on the speed at which the adaptive measurement processing can be performed, in turn constraining the photonic clock rate of the system. The design and testing system is freely available for use as the basis of analysis of more complex designs, incorporating more recent proposals for photonic quantum computing. Our work points to the importance of codesigning the classical control system in tandem with the quantum system in order to meet the challenging specifications of a photonic quantum computer.

INDEX TERMS Field-programmable gate array (FPGA), measurement and feed-forward, measurement-based quantum computing (MBQC), photonic quantum computing, timing analysis.

I. INTRODUCTION

Quantum computers are enjoying a period of intense research activity. This is due to the possibility of very large speed-ups in finding effective solutions for certain classes of problems that are hard or impossible to solve using classical computers. These include, in the near-term, the simulation of other quantum mechanical systems with applications in quantum chemistry [1] and, in the longer term, certain kinds of search-related problems [2].

Currently, only relatively small quantum computers have been built, containing less than one hundred qubits, in a multitude of competing technologies [3]. However, it is hoped that with increasing understanding over how to engineer and control quantum systems at scale, there will be a substantial increase in the computational power of quantum computers in the near future.

However, a few hurdles lie in the way of this scaling process. Often discussed are the characteristics of the qubits

themselves: for example, the difficulty of achieving high enough fidelity gate operations on physical qubits [4] or how to fabricate devices at scale [5]. One aspect of the problem, which is discussed less often, is the classical electronic control requirements that need to be met for these machines.

At first glance, this is surprising since the common link between all the platforms for quantum computing is the need for traditional electronics to control them. However, there is a general tendency to view classical signal processing as a "solved problem," at least in comparison to the difficulty of the quantum information processing, and assume that any requisite performance can be achieved using custom application-specific integrated circuits (ASICs). This is understandable, given the high degree of performance and sophistication that modern (digital) complementary metal-oxide semiconductor (CMOS) electronics can routinely achieve. However, adapting and applying classical electronics to quantum control problems is not straightforward and is an active area of research across all quantum computing platforms. For example, in superconducting qubit-based quantum computers, it is an open question as to how best to integrate the microwave control electronics close to the qubits in order to reduce the number of interfacing wires that prevent scalability in that architecture [6].

The flip-side to modern CMOS electronics having reached the level of sophistication that it enjoys today is that there is not a great deal of room left for improvement in performance. For example, clock speeds in computers are plateauing [7], and transistor sizes are reaching their limits [8]. Quantum devices, on the other hand, being a relatively young technology, are expected to see a Moore's law-like improvement in the future [4]. Consequently, it is critical to understand the limitations that classical electronics and control will impose on current quantum computing platforms.

The majority of the prior work discussing photonic measurement-based quantum computing (MBQC) has focused on the quantum resources required and the theoretical architecture of the system [9]–[13]. Many of these proposals seek to address the pervasive issues of photon loss and lack of deterministic two-qubit entangling gates [14] that all quantum photonic systems must overcome. Recently, fusion-based quantum computation (FBQC) has been introduced, specifically aiming to incorporate the fusion operations necessary for entangling photons into the computation [12]. More generally, many modern schemes take explicit account of error correction and fault tolerance in order to provide a candidate system that works in theory [13], [15].

What is not present in the literature, to the best of our knowledge, are any concrete candidate electronic control system implementations for these schemes. We define here what we consider to be a valid implementation.

1) The implementation should clearly define the problem it solves, including any underlying assumptions, and provide a design for the electronic control system under discussion.
2) The control system should be tested and verified, so that there is some assurance that it does in fact do what it is designed to do.
3) The control system design should be freely available, so that the previous two criteria can be independently checked.

The purpose of creating such an implementation is to provide evidence that a candidate scheme for photonic quantum computing works or provide reasons why it does not. We believe that a thorough analysis of concrete control system implementations is critically important if photonics is to compete with superconducting qubit-based quantum computers as a candidate architecture for quantum computing.

Producing an implementation of the control system for a photonic quantum computing architecture is a difficult engineering undertaking, in the same way that the design of the control system for a superconducting qubit quantum computer involves the design of a complex cryogenic/RF electronic system. It is, therefore, challenging to lay out in one go a full implementation for any scheme for photonic quantum computing, even in the simplest cases. This is because most implementations of photonic quantum computing rely on MBQC, which is a difficult mathematical subject for electronic engineers to easily understand because it is highly nonintuitive and lacks the familiarity of the gate-based model.

The purpose of the current article is twofold. First, we analyze the timing constraints imposed by classical electronics on the operation of a photonic quantum computer based on an ideal cluster state generator. This is the first architecture that was seriously proposed for the implementation of photonic quantum computing, which does not suffer from the unlimited-depth problem of the Knill–Laflamme–Milburn scheme [16]. We show that even in this highly idealized scenario, classical electronics imposes significant timing constraints on photonic quantum computers. Based on this implementation alone, we cannot draw any conclusions relating to other photonic quantum computing architectures; rather, we wanted to provide a building block for future quantum engineers to produce candidate implementations for more realistic and modern proposals for photonic quantum computing.

The second purpose of this article is to provide a simple and graphical presentation of MBQC, which is of the kind that we would have liked to use while learning this subject. We believe that an important barrier to entry for electronic engineers is the abstract nature of the literature describing the subject.

There are several important constraints the electronic control system may impose on the implementation of photonic MBQC. First, the speed at which the electronics can be made to operate determines the maximum photonic clock cycle of the system. In particular, the "flying-qubit" architecture of photonic MBQC (discussed in Section III) requires adaptive measurement settings to be worked out before the arrival of

IEEE Transactions on Quantum Engineering

the next column of photons in the cluster state. Second, the complexity of the system—particularly the analog parts—determines how much on-chip area is taken up with classical processing. This is particularly important because the "unit cell"[1] of the electronic system must be duplicated once per qubit in photonic MBQC, and signal routing becomes challenging as the system starts to scale. Third, the noise that is introduced by the analog stages of the control system will potentially introduce logical errors that need to be accounted for.

In this work, we focus on the first of these questions: Specifically, what timing constraints does the digital part of the system impose upon the analog and wider photonic systems, and how does that affect the overall photonic clock rate of the quantum computer? We perform the analysis for a system with an ideal cluster state generator and consider the signal delay in the digital domain (after photon detection and logic-level amplification to the input of the analog system which is needed to set the bases for the next measurement round; details in Fig. 6).

The structure of this article is as follows. In Section II, we provide a practical description of the parts of MBQC needed to understand the results of this article. In Section III, we describe a simple model of photonic MBQC—based on an ideal cluster state source—showing the analog and digital subsystems which are necessary for a basic implementation of the system. In Section IV, we present an example design for the digital component of the classical processing, targeting a Xilinx 7-series field-programmable gate array (FPGA). In Section V, we describe the functional verification of the design. In Section VI, we use static timing analysis of the implemented design to derive constraints on the analog parts of the system and on the overall photonic clock frequency of the quantum computer. Section VII contains changes to the underlying model to make it more realistic, which would increase the complexity of the digital system. Finally, we discuss the wider implications of our results in Sections VIII and IX.

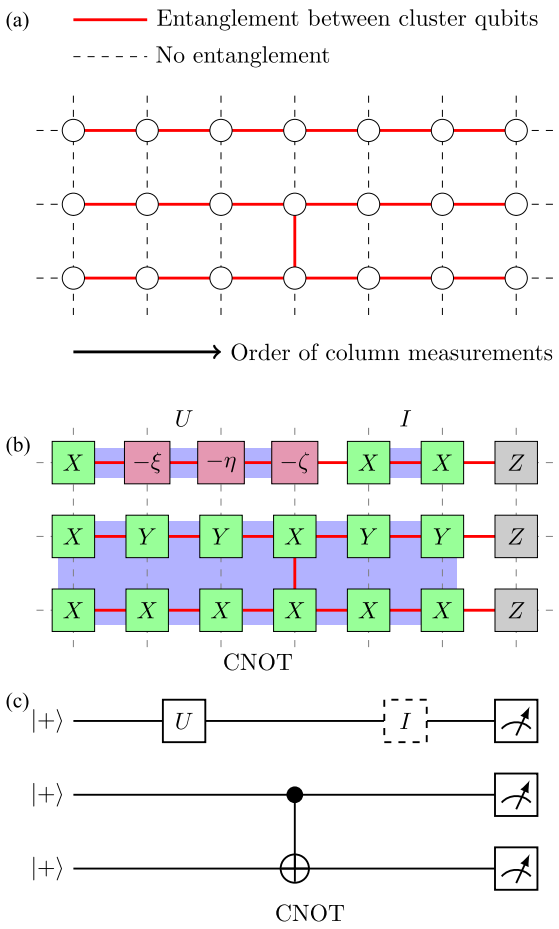

FIGURE 1. (a) The cluster state is made from a rectangular array of qubits (the white dots), each of which may be entangled with its four nearest neighbors. When a computation is performed, a specific pattern of entanglement is required that matches the shape of the circuit. (b) Quantum computation is performed by measuring the cluster qubits in bases derived from the measurement pattern. The shaded blue regions show which cluster qubits are involved in implementing which gates. The identity gate is included to pad the length of the one-qubit gate $U = R_x(\zeta)R_z(\eta)R_x(\xi)$ so that it matches the CNOT. (c) Quantum circuit that is performed by the measurement pattern in (b).

II. INTRODUCTION TO MEASUREMENT-BASED QUANTUM COMPUTING

Quantum computing in the gate-based model (see Appendix A for details) consists of the following steps.

1) An initial quantum state $|\phi\rangle$ is prepared on N qubits.
2) Quantum gates are applied to the qubits.
3) The resulting state $|\psi\rangle$ is measured, which constitutes the output from the quantum circuit.

MBQC is a different way to obtain the same resulting output state $|\psi\rangle$, by performing single-qubit measurements on a more complicated initial state called a cluster state. It consists of the following steps.

1) Prepare a special quantum state, called a cluster state, on a larger number $M > N$ of qubits. The main feature of the cluster state is that adjacent qubits are entangled together, which is represented using line segments in Fig. 1(a).
2) Measure qubits from the cluster state one at a time, according to rules that correspond to the quantum circuit, until all but N have been measured.
3) Finally, the resulting state $|\psi'\rangle$ on the N remaining qubits is measured in the computational basis, which constitutes the output from the circuit.

The initial state $|\phi\rangle$ in the gate-based model is a matter of convention; if each qubit is initially prepared in the $|+\rangle$ state, then the output states $|\psi\rangle$ and $|\psi'\rangle$ from the gate-based

[1]Throughout the article, the unit cell refers to the subsystem of the digital design which reads measurement outcomes for each logical qubit, processes these outcomes, and drives modulators relating to the basis settings of the logical qubit. The term "logical qubit" refers to the qubit which would be available to a user of the system (i.e., it is the qubits that they would use in performing their quantum algorithms with the quantum computer).

model and MBQC are the same, meaning that any algorithm expressed in the gate-based model can be equally well performed using MBQC.

For a comprehensive overview of MBQC, see [17]. A short pedagogical introduction is contained in [18]. What follows is a brief description of the main features of MBQC which are relevant to this article.

1) LOGICAL QUBITS AND GATES IN MBQC

Each horizontal line of entanglement in the cluster state corresponds to a single qubit in the gate-based model, which we will call a logical qubit, to distinguish it from the cluster qubits that make up the cluster state. One-qubit gates in the gate-based model involve measurements of the cluster qubits along a logical qubit row according to rules that determine the basis settings of each measurement, and define what to do with the measurement outcomes. Two-qubit gates require vertical lines of entanglement which join the logical qubit rows together, as shown in Fig. 1(a) between the second and third rows.

2) MEASUREMENT PATTERNS FOR GATES

Each gate G that is implemented in MBQC is defined by a measurement pattern, which is a set of rules describing the following.

1) How many cluster qubits are needed to realize the gate G and what pattern of entanglement is necessary between those cluster qubits.
2) Which basis to use for each cluster qubit measurement.
3) How to process the outcomes from the cluster qubit measurements.

A given computation involving multiple gates, such as the one shown in the gate-based model in Fig. 1(c), can be performed using MBQC by concatenating[2] the measurement patterns for each gate [the blue shaded regions in Fig. 1(b)]. The resulting pattern contains one row for each qubit in the gate-based model (here, $N = 3$), and a number of columns defined by the length of the concatenated measurement patterns (the total number of cluster qubits is $M = 21$).

In making the measurements defined by the measurement patterns, each cluster qubit is removed one by one until only the rightmost column remains unmeasured. The final column of the cluster state is measured in the computational basis as shown in Fig. 1(b), which represents the output from the quantum circuit.

The arbitrary one-qubit gate U in Fig. 1(c) is realized using a measurement pattern of four cluster qubits in the top row of Fig. 1(b), and the controlled NOT (CNOT) gate[3] is realized using a measurement pattern of 12 cluster qubits spanning the bottom two rows of Fig. 1(b) (note the vertical entanglement link).

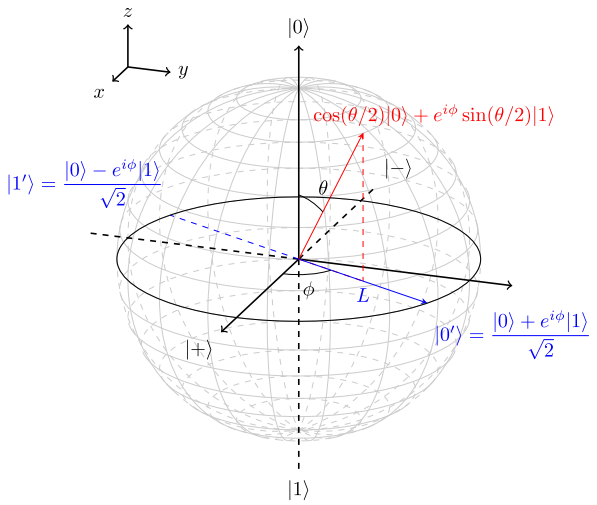

FIGURE 2. The state of a single qubit, represented as a point on the Bloch sphere. A measurement of a single qubit can be made along any straight line through the Bloch sphere. In MBQC, measurements in the purple and green boxes in Fig. 1 are made along lines L in the equator of the Bloch sphere, parametrized by a single angle ϕ. Computational basis measurements (denoted using gray boxes in Fig. 1) are made along the vertical line through $|0\rangle$ and $|1\rangle$.

All measurements shown in green and purple boxes in the figure are performed along lines L that lie in the equator of the Bloch sphere (Fig. 2). Green boxes containing X or Y are measurements along the x- or y-axes, respectively. Purple boxes are measured along a line L with an angle ϕ derived from the value in the box and measurement outcomes of other cluster qubits. The gray boxes represent computational basis measurements, which are made along the z-axis of the Bloch sphere.

3) PERFORMING THE CLUSTER QUBIT MEASUREMENTS

As described in Appendix A, the only physical measurements that can be performed are computational basis measurements. All the other measurements (in the equator of the Bloch sphere) are performed by applying a one-qubit gate to the given cluster qubit and then measuring it in the computational basis.

It is important to understand that the one-qubit gates that set the measurement bases in the measurement patterns are different from the one-qubit gates implemented by MBQC, such as U in Fig. 1. The former are basic operations that, together with computational basis measurements, are required for implementation of MBQC. They are analogous to the physical layer in a communication system because they must

[2]In [17], measurement patterns are taken to include the "output" qubits, which is the first column of qubits directly to the right of the measurement pattern. In this scheme, measurement patterns must overlap (because the output qubit column is also the input qubit column for the next gate pattern). In this article, we associate the output qubit with the next measurement pattern, so that patterns can be simply concatenated.

[3]The symbol for a CNOT gate shown in Fig. 1 is the same as a classical XOR applied to the target qubit. This is because the CNOT gate can be thought of as adding the value of the control qubit to the target qubit modulo 2. We make extensive use of the classical XOR operation in subsequent figures in this article. For clarity, we state here that all instances of the XOR symbol in this article—apart from that in Fig. 1—are classical XOR gates, not CNOT gates.

IEEE Transactions on Quantum Engineering

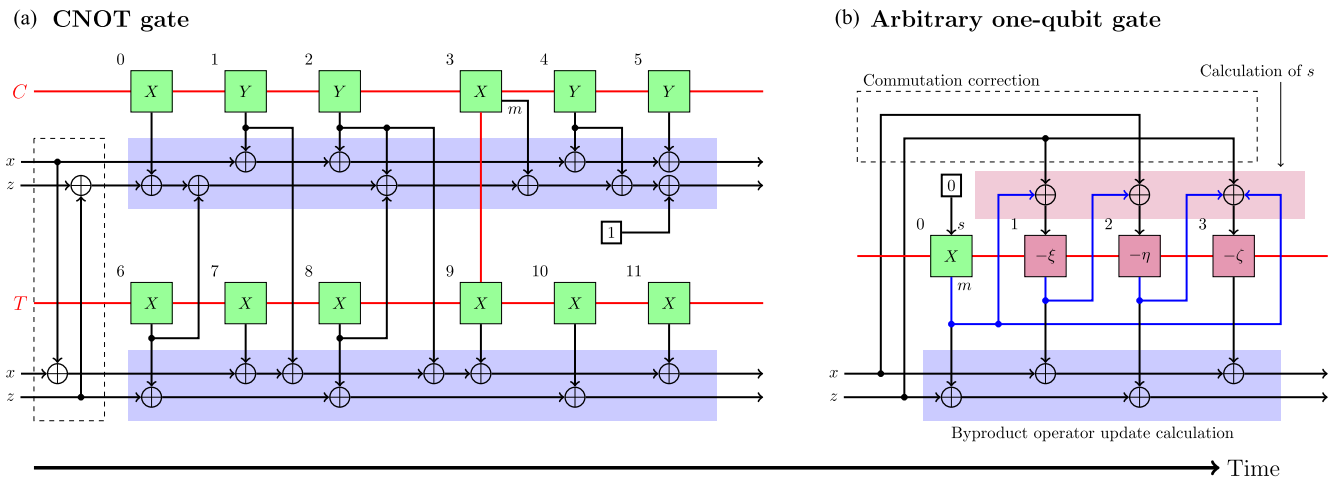

FIGURE 3. The two measurement patterns we consider in this article are the CNOT gate and the arbitrary one-qubit gate $U = R_x(\zeta)R_z(\eta)R_x(\xi)$. In each block, the black line connected to the top of the box is the adaptive measurement setting s. The line connected to the bottom of each box is the measurement outcome m. For the CNOT gate, on the left, there are no adaptive measurement settings because all the measurement bases are X or Y. The byproduct operator bits are shown using the lines x and y below each logical qubit row in the figure. The computation of the byproduct operators (shaded in blue) is more complicated and involves mixing outcomes from the control C and target T rows. On the other hand, for the arbitrary one-qubit gate, the byproduct operator calculation is simple, but the adaptive measurement settings depend on previous measurement outcomes (shaded purple). The commutation correction for each gate is enclosed in dashed lines. For the CNOT gate, it involves mixing the byproduct operators before applying the pattern. For the one-qubit gate, the byproduct operators must be stored because they are used in the adaptive measurement setting calculation. In Section III, the condition is imposed that columns are measured from left to right, so as to be compatible with photonic MBQC.

be realized by some physical mechanism; for example, using photonic qubits, as we discuss in Section III. The logical one-qubit gates U do not correspond to any basic physical operation and, instead, arise as a result of applying the measurement pattern to the cluster qubits. They are analogous to the logical data layers in a communication network, which use the resources of the physical layer to transmit logical information.

In the following sections, we describe in detail the measurement patterns for the one-qubit gate and the CNOT gate, which includes how to obtain the adaptive measurement settings and what to do with measurement outcomes.

4) MEASUREMENT BASIS ANGLES AND ADAPTIVE MEASUREMENTS

Every measurement that is part of a measurement pattern is measured along a line L in the equator of the Bloch sphere, as shown in Fig. 2. It is, therefore, specified by one real angle ϕ. In MBQC measurement patterns, this angle is made up of a value θ, and a sign bit s, such that $\phi = (-1)^s\theta$. The value of θ is shown in the purple boxes in Fig. 1(b). Note that θ may be negative.

For the green boxes in Fig. 1, the value of θ is 0 for X and $\pi/2$ for Y. In the first case, the value of s does not affect the basis angle ϕ at all. In the second case, the roles of $|0'\rangle$ and $|1'\rangle$ are swapped because L is reversed; however, since the outcome of the measurement is random, the swapped measurement outcomes can be corrected in the calculation of byproduct operators (see Section II-A5). Therefore, the X and Y measurements are not affected by the value of s.

The value of θ is a characteristic of the quantum circuit being implemented. The value s, however, depends on the outcomes of other (prior) measurements in the measurement pattern. The measurement bases in MBQC are, therefore, adaptive because the basis in which a cluster qubit is measured may depend on the outcomes of measurements of other cluster qubits that have been measured before. We will refer to s henceforth as the adaptive measurement setting.

Any measurement pattern, such as the CNOT gate, which contains only X and Y measurements, does not involve adaptive basis settings because the value of s has no effect. It can be shown that the set of gates implementable with these nonadaptive patterns is the Clifford gate set [17], which is not universal [19]. For universal quantum computing, it is necessary to include a gate such as the one-qubit gate that does require adaptive measurement settings.

In Fig. 3(b), the measurement pattern for the arbitrary one-qubit gate (corresponding to a rotation of the Bloch sphere) is shown in detail [17]. The shaded purple region (particularly the blue wires) shows how the adaptive measurement setting for each measurement is computed from previous measurement outcomes. The dependence between s and measurement outcomes implies that the measurements must be made from left to right, which is also indicated by the arrow of time at the bottom of the figure.

The measurement pattern for the CNOT gate is shown in Fig. 3(a). This is not the same pattern as that presented in the original MBQC paper [17], which uses three logical qubit rows. The derivation of the CNOT pattern in Fig. 3 is contained in Appendix B. We use this modified CNOT measurement pattern because it considerably simplifies our example digital implementation in Section IV, which only supports nearest-neighbor connectivity of logical qubits.

5) BYPRODUCT CALCULATIONS

As the measurement pattern proceeds, the random outcomes of the measurements introduce correctable errors in the computation. These errors are known as byproduct operators because they are unintended logical operations that occur as a byproduct of the MBQC measurements.

Specifically, after any N-qubit gate G has been applied to a state $|\psi\rangle$ using its measurement pattern, the resulting state is actually $BG|\psi\rangle$, rather than simply $G|\psi\rangle$, where B is a gate (called the byproduct operator) given by

$$B = \prod_{i=0}^{N-1} Z_i^{z_i} X_i^{x_i}, \qquad x_i, z_i \in \{0, 1\}.$$

The byproduct operator for the logical qubit i is specified by two bits x_i and z_i, which are updated as the computation proceeds. By an abuse of notation, we will refer to the pair (x_i, z_i) as the byproduct operator as well. For N logical qubits (N rows of the cluster state), $2N$ bits are needed to store the byproduct operators. At the start of the computation, they are all initialized to zero, because no gate has been performed; so no errors have been introduced. As the computation proceeds, the outcomes of the measurements in the pattern are XORed into the x_i and z_i according to prescribed rules, described below and shown in Fig. 3.

For the one-qubit gate in Fig. 3, the new byproduct operators (x', z') are calculated according to the rule

$$z' = z \oplus m_0 \oplus m_2$$
$$x' = x \oplus m_1 \oplus m_3$$

where m_k is the measurement outcome from the kth qubit, numbered according to Fig. 3.

For the CNOT pattern, two byproduct operators are involved: one for the control qubit row (x_c, z_c) and one for the target qubit row (x_t, z_t). The new byproduct operators (x_c', z_c') and (x_t', z_t') are calculated using

$$z_c' = z_c \oplus m_0 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_8 \oplus 1$$
$$x_c' = x_c \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \tag{1}$$

and

$$z_t' = z_t \oplus m_6 \oplus m_8 \oplus m_{10}$$
$$x_t' = x_t \oplus m_1 \oplus m_2 \oplus m_7 \oplus m_9 \oplus m_{11}. \tag{2}$$

Unlike for the one-qubit gate, the byproduct operators for a given logical qubit row are calculated using measurements from other rows. Note the addition of the constant 1 in the control qubit byproduct operator.

On the face of it, byproduct operators appear to introduce errors into the computation because the gate BG is performed instead of the desired gate G. However, the effect of this error can be corrected after the final column of Z-measurements in the MBQC process has been performed: the outcome from any logical qubit row i where $x_i = 1$ has its outcome flipped from a zero to a one or vice versa [17]. This action undoes the effect of the byproduct operators, leaving a circuit that effectively only implements the gate G as desired. The z_i components are not used because they correspond to a phase shift which does not affect the probability of measuring a zero or one in a computational basis measurement. However, as we describe in the next section, it is necessary to keep track of their values because they can affect the value of the x_i, through the process of commutation corrections.

6) COMMUTATION CORRECTIONS

The byproduct operators are used to correct the outcomes obtained after the MBQC circuit is finished. However, the correction only works if the byproduct operators are the last operation before the final column computational basis measurement, which is only the case if a single gate G is performed.

If multiple gates G_k are performed on a state $|\psi\rangle$, then the resulting state $|\phi\rangle$ will be

$$|\phi\rangle = (B_K G_K) \ldots (B_1 G_1)(B_0 G_0)|\psi\rangle. \tag{3}$$

These interleaved byproduct operators cannot be corrected at the end of the circuit. Instead, it is necessary to move all the byproduct operators to the end (the leftmost side of the equation). To do that, after each new gate G_{k+1} is applied, it is necessary to commute the current byproduct operators B_k and the gate G_{k+1}, so that the byproduct operators are always on the leftmost side of the equation. This is illustrated in the following for the application of the second gate G_1:

$$B_0 G_0|\psi\rangle \mapsto B_1 G_1 B_0 G_0|\psi\rangle \mapsto B_1 B_0' G_1' G_0|\psi\rangle \mapsto B_r G_1' G_0|\psi\rangle \tag{4}$$

where $G_1 B_0 = B_0' G_1'$, and the prime indicates the change that may occur in either gate. The byproduct operators B_1 and B_0' can be combined into a resulting byproduct operator B_r by adding together the values of (x_i, z_i) bitwise modulo 2 for each operator. The state on the right of (4) is, therefore, transformed to the same form of the state on the left, so that on the application of the next gate G_2, the process can be repeated and the byproduct operators are always kept on the left. We will call the process of commuting B through G a commutation correction.

In practical terms, the commutation correction is an operation that is performed before a gate is applied, by manipulating the current value of the byproduct operators and the upcoming gate so as to have the effect of (4). For the two measurement patterns we consider in Fig. 3, the commutation corrections are quite simple. In the case of the CNOT gate $G = \text{CNOT}$, $G' = G$, and only the byproduct operator B changes to B', according to the rule

$$z_c' = z_c \oplus z_t$$
$$x_c' = x_c$$
$$z_t' = z_t$$
$$x_t' = x_t \oplus x_c. \tag{5}$$

For the one-qubit gate $G = U$, the byproduct operators remain the same, $B' = B$, but the gate itself G must be modified. The modification is made by using the values of the byproduct operators to affect the adaptive measurement settings, by

IEEE Transactions on Quantum Engineering

XORing the byproduct operators with previous measurement outcomes to form the values of s for each cluster qubit [17], as shown in Fig. 3(b). The calculation of the adaptive measurement settings s_j for each cluster qubit j is shown in the following equations:

$$\begin{aligned} s_0 &= 0 \\ s_1 &= m_0 \oplus z \\ s_2 &= m_1 \oplus x \\ s_3 &= m_0 \oplus m_2 \oplus z. \end{aligned} \tag{6}$$

It is necessary to make a copy of the byproduct operators (x, z) before measuring the cluster qubits because, otherwise, they will be overwritten during the calculations described in the previous paragraph. For example, after the measurement of cluster qubit 1 in the arbitrary one-qubit gate in Fig. 3, both the x and z values have been updated by measurement outcomes from cluster qubits 0 and 1. However, the old values of x and z are necessary in the measurement settings for cluster qubits 2 and 3.

In addition to the timing constraints imposed by the calculation of the adaptive measurement settings, the need to track the byproduct operators and calculate commutation corrections leads to additional timing constraints on digital implementations of the system because they must be tracked in real time and may affect adaptive measurement settings.

7) CUTTING OUT THE RIGHT MEASUREMENT PATTERN FROM THE CLUSTER STATE

In the measurement pattern for the CNOT gate in Fig. 3(a), there are missing links between some of the cluster qubits. However, a fully connected cluster state does not contain missing links.[4] In order for the CNOT measurement pattern to work, it is necessary to use an ideal cluster state generator, meaning one which can produce arbitrary patterns of nearest neighbor entanglement in the cluster state.

In the more general approach to MBQC [17], the computation always begins from the full cluster state. The links around a cluster qubit can then be removed by performing a computational basis measurement on that qubit. Using this method, cluster states containing less entanglement can be obtained from fully connected cluster states.[5]

The computational basis measurement that cuts out a cluster qubit incurs an additional step in the calculation of basis angles for surrounding cluster qubits. If the outcome of this measurement is a 1, then a rotation $R_z(\pi)$ must be applied to the surrounding qubits, before they are measured according to any measurement pattern [17]. This gives rise to a more general form for the basis angle

$$\phi = \pi c + (-1)^s \theta \tag{7}$$

where c is a single bit that is formed by XORing the measurement outcomes from any cluster adjacent qubits that have been removed using computational basis measurements.

Since the cut-out correction caused by removing a given cluster qubit must be performed before making the measurement of that qubit, the cutting out of cluster qubits introduces a measurement dependency between the measurement outcome of the cut-out qubit and the measurement angle ϕ of the qubits above and below it in the same column. This is different from our previous discussion on adaptive settings where the measurement outcomes had dependencies across columns in the cluster state and there was no intracolumn dependence. This can be handled by measuring each column in two rounds. First, the qubits that must be cut out are measured; then, the surrounding cluster qubits are measured using the modified basis angle ϕ in (7).

This aspect of cutting out the right-shaped pattern from a fully connected cluster state can be avoided entirely if the right-shaped cluster state is available from the beginning, using an ideal cluster state generator that can generate arbitrary patterns of entanglement. We assume the existence of such a cluster state generator for the purposes of this article and do not consider cut-out corrections any further.

III. SIMPLIFIED MODEL OF PHOTONIC QUANTUM COMPUTING

In this section, we describe how to implement MBQC using photons as qubits. We do not consider the generation of photonic cluster states, which is a separate subject in its own right [9], [12]. Instead, we assume an ideal photonic cluster state generator, which can generate arbitrarily shaped rectangular cluster states, and describe how one can use it to perform photonic MBQC. We begin by describing how photons can be used as qubits.

A. QUANTUM COMPUTING USING PHOTONS AS QUBITS

In photonic quantum computing, a qubit is realized using a single photon. In the dual-rail encoding considered in this article, a single photon passes through one waveguide or another depending on whether the qubit it represents is in the state $|0\rangle$ or $|1\rangle$, as shown in Fig. 4. A qubit encoded like this can be measured in the computational basis by placing a single-photon detector at the end of the pair of waveguides. It is important to realize that this process destroys the qubit (by absorbing the photon), unlike a matter-based qubit which can be reused after measurement.

Modulators and beamsplitters can be used to realize an arbitrary one-qubit gate, as follows. First, a modulator in the $|1\rangle$ waveguide realizes an arbitrary z-rotation, shown in Fig. 4(d). Then, the variable beamsplitter shown in Fig. 5

[4]Some authors distinguish a fully connected cluster state from a partial cluster state, which is an example of the more general "graph state," because it has vertices corresponding to cluster qubits and edges corresponding to entanglement links. In this terminology, a fully connected cluster state is a graph state whose graph is the fully connected 2-D lattice. To simplify our discussion, we refer to all graph states of any entanglement pattern as cluster states.

[5]This is one reason for using the three-row CNOT pattern, as in [17]: It is necessary to preserve a buffer row of cluster qubits between any two logical qubit rows that should not be connected. These buffer qubits are measured so as to remove the links between logical qubit rows.

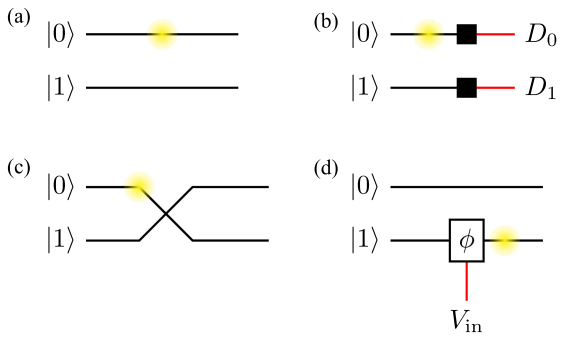

FIGURE 4. (a) A single photon in two waveguides can be used as a qubit. If the photon is in the top waveguide, then the qubit is in the $|0\rangle$ state, whereas if it is in the bottom waveguide, the qubit is in the $|1\rangle$ state. (b) Computational basis measurements can be performed by placing a single-photon detector at the end of the waveguides. Basic one-qubit operations can be realized using linear optical elements such as (c) beamsplitters and (d) modulators. Complex operations can be realized by placing the elements one after the other.

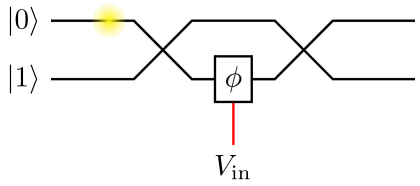

FIGURE 5. A variable beamsplitter, which realizes an $R_x(\phi)$ rotation, is formed by placing two fixed beamsplitters on either side of a modulator.

realizes an arbitrary x-rotation. Finally, a second modulator in the $|1\rangle$ waveguide realizes another arbitrary z-rotation, which completes the decomposition $R_z(\alpha)R_x(\beta)R_z(\gamma)$.

In contrast to many other physical realizations of quantum computing, including superconducting qubits and trapped ions, that have a natural way to implement two-qubit operations [20], there is no simple deterministic way to implement the CNOT gate, or any other two-qubit entangling gate, in terms of passive linear optical elements (modulators and beamsplitters). This is mainly due to the weakness of the direct photon–photon interaction. While this might appear to be a key limitation for photonic quantum computing, it was shown that one can implement an artificial nonlinear gate that works probabilistically by using additional auxiliary photons and photodetection [16]. By parallel multiplexing these entangling gates, one can overcome their inherently probabilistic operation [11].

One of the arguments in favor of photonic MBQC is the absence of two-qubit gates in the implementation of a quantum circuit [9]; after the cluster state has been generated, only one-qubit gates and computational basis measurements are necessary. Much of the complexity is pushed to the task of generating the cluster state [10], which is responsible for all the entanglement between the qubits. As we describe in the next section, it is possible to generate the cluster state one column at a time, so that each photon only has to travel through a fixed length cluster state generating system, followed by a fixed length measurement system so that the overall photon loss can be bounded irrespective of the length of the equivalent quantum circuit (in the gate-based model) being implemented. Given that photon loss is a primary source of error (and decoherence) for photonic quantum computing, this represents another important advantage of photonic MBQC [10].

B. PHOTONIC MEASUREMENT-BASED QUANTUM COMPUTING

For matter-based implementations of MBQC, the grid of qubits directly corresponds to a 2-D physical array of atoms. However, for photonic quantum computing, it is not feasible to maintain a static array of qubits long enough to perform the measurements. This is because a photon is always moving; so the only way to store it is to place it in a long waveguide, called a delay line, or keep it circulating in an on-chip cavity, such as a microring resonator. Both of these approaches eventually lead to photon decay, primarily due to scattering and absorption loss in the waveguide which is exacerbated in an integrated photonics platform (waveguide loss in a silicon platform is ~ 1 dB cm^{-1} [21] compared with ~ 0.2 dB km^{-1} for optical fibers [22]).

Instead, the cluster state can be generated one column at a time, and each column can be measured one after the other. This is opposite to the original presentation of MBQC [17], where the goal was to separate the processes of generating the cluster state and making the measurements. The motivation for generating the cluster state all at once was also due to physical considerations: A matter-based cluster state can be generated using a tunable Ising interaction that acts globally on the system [18]. However, it can be shown that the two approaches are equivalent [17, Section II.D]; there, the successive column approach is used as a tool for verifying measurement patterns.

When the cluster state generation and the photon measurement is alternated, a single photon only has to travel from its source, through the cluster state generator, through a fixed length waveguide, and finish at the measurement block.

For photonic MBQC, in Fig. 3, the horizontal axis can, therefore, be interpreted as time, and the vertical axis as space. Each column of the cluster state is generated one at a time, progressing from left to right. Using this approach introduces a restriction which is not present in the matter-based realization of MBQC. The scheme is only viable if the measurement settings for the currently measured block only depend on the outcomes of previously measured columns. This is quite a severe restriction, ruling out many of the measurement patterns originally proposed in [17] (for example, the CPhase gate, a two-qubit gate that depends on a continuous parameter). However, this requirement is satisfied for the

IEEE Transactions on Quantum Engineering

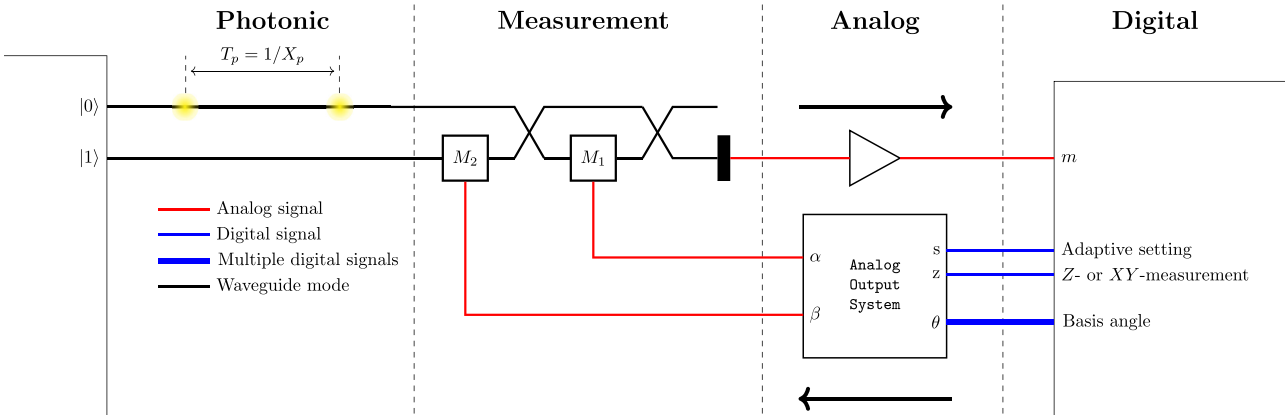

FIGURE 6. One row of the system diagram of the classical control required to implement photonic MBQC. The cluster state generator is assumed to be ideal, outputting columns of photons at the photon clock frequency X_p. The cluster qubits represented by these photons are measured in bases specified by the measurement pattern in the measurement block, which is controlled by the voltages α and β from the analog output system. The measurement results are amplified and processed by the digital system, which uses them to calculate subsequent adaptive measurement settings s and byproduct operators. A copy of the system shown is required for each logical qubit, but each block is independent apart from the cluster state generator and the digital system.

one-qubit gate and the CNOT gate described here. In the case of the CNOT gate, there are no measurement dependencies. For the one-qubit gate, all the measurement dependencies [the blue lines in Fig. 3(b)] point from left to right.[6]

C. TIMING CONSTRAINTS ON THE CLUSTER STATE

We do not consider the generation of the photonic cluster state, apart from making the following remark about the choice of time delay between the generation of columns, which is crucial for our timing analysis.

In order to entangle photons P_n and P_{n+1} from two adjacent columns n and $n+1$ of the cluster state, they must be brought to the same location (for example, a beamsplitter) at the same time. However, when performing the cluster qubit measurement for the MBQC measurement pattern, P_n (from column n) must arrive at the detector a finite time before P_{n+1} (from column $n+1$), to allow time for the processing of measurement settings, byproduct operators, and commutation corrections. Therefore, P_{n+1} must experience a delay T_p (realized using an on-chip delay line or optical fiber) after the entangling operation of adjacent columns and the measurement block. The inverse of this delay $X_p = 1/T_p$ is the photonic clock frequency, which is the rate at which columns are produced and measured, and which determines the speed at which the quantum computation progresses.

Two distinct physical mechanisms provide upper and lower bounds for this delay. An upper bound is given by the loss of the on-chip delay line, optical fiber, or routing system involved in the delay of the photon. The lower bound is given by the time required to process the measurement outcomes. The object of our analysis is to estimate the lower bound.

[6]In the context of photonic quantum computing, this is sometimes referred to as feedforward of measurement results.

D. FULL MBQC SYSTEM

Fig. 6 shows the full system required for processing one row of the MBQC measurement pattern, which corresponds to one logical qubit. It consists of the following six parts.

1) The cluster state generator, which outputs the dual-rail encoded photon in each column of the cluster state one after the other. The photon has been entangled with the previous photon in the same row, and with the photons in the rows above and below as necessary for the measurement pattern.
2) The delay line, described in the previous section, which is necessary to temporally separate the photons in adjacent columns after they have been entangled.
3) The measurement block, which consists of passive linear optical elements that apply a configurable one-qubit operation, followed by a computational basis measurement.
4) The photon detector amplifier which converts the output from a single-photon detector to a logic level suitable for processing by a digital system.
5) The digital system that processes measurement outcomes into adaptive measurement settings and keeps track of byproduct operators.
6) The analog output system, controlled by the digital system, which produces the analog voltage levels needed to drive the modulators in the measurement block.

The job of the digital system is to convert the measurement results into adaptive measurement settings for future measurements, and byproduct operators for interpreting the final measured outcomes.

The input to the digital system is the output pulse from the photon detector amplifier. This may be, for example, a

TABLE 1. One-Qubit Rotations Generated by Analog System Modulator Voltages

z	Rotation	Measurement basis
1	$R_z\left(\frac{\pi}{2}-\phi\right)R_x\left(\frac{\pi}{2}\right)$	XY-measurement at an angle ϕ
0	(None)	Computational basis measurement

When $z = 1$, a regular XY-basis measurement is performed, which accounts for the majority of cluster state measurements. A computational basis measurement is made at the end of the computation by setting $z = 0$.

superconducting-nanowire single-photon detector [23] followed by a low-noise amplifier [24].

The output from the digital system includes the digital form of the angle θ, the adaptive measurement setting output s, and a signal z which determines whether the measurement is in the XY-plane of the Bloch sphere, or if it is a computational basis measurement.

The analog output system is responsible for generating the voltages that control the modulators in the measurement block. It may be implemented using a combination of fast digital-to-analog converters (DACs) and modulator drivers. Two modulators are necessary: one (M_1 in Fig. 6) chooses between an XY-measurement and a computational basis measurement; and another (M_2) controls the basis angle ϕ for the XY-measurement. They are controlled by the voltages α and β, respectively, defined as follows:[7]

$$\alpha = \frac{\pi}{2}z$$
$$\beta = \frac{\pi}{2} - \phi = \frac{\pi}{2} - (-1)^s\theta. \tag{8}$$

These modulator voltages realize the one-qubit rotation $R_x(\alpha)R_z(\beta)$, which sets the basis for the measurement. The one-qubit rotations are summarized in Table 1.

The voltage α controls the R_x rotation portion of the measurement setting, which determines whether the measurement is a computational basis measurement ($z = 0$) or an XY-measurement ($z = 1$). The voltage β controls the angle of the XY-plane measurement ϕ, which is itself determined by the fixed value θ and the adaptive measurement setting s.

In this article, we focus on the digital control system and present a simple reference design capable of performing the one-qubit gate and CNOT gate described in Section II. We analyze the timing behavior of this design by implementing it with an FPGA and performing static timing analysis. The main objective of this analysis is to place timing constraints on the input and output analog systems and, therefore, on the overall quantum photonic clock rate of the system. In the interest of simplicity, we ignore the final computational basis measurement of MBQC, which can easily be incorporated by setting $z = 0$ for the final column of the pattern.

[7]Voltages are expressed in modulator-phase units, where $V = 1$ is chosen such that the modulator applies a 1 rad phase shift.

TABLE 2. Summary of the Notation Used in Fig. 8

Signal	Meaning
X_p	The photonic cycle clock
X_s	The measurement sample clock
X_r	The latch reset clock
`ops_above`	Byproduct operator values from the logical qubit above
`ops_below`	Byproduct operator values from the logical qubit below
`P[15:0]`	The two-byte program word input
`m[2:0]`	Latched measurement inputs to `control system`
`m[2]`	Measurement outcome from logical qubit above
`m[1]`	Measurement outcome from current logical qubit
`m[0]`	Measurement outcome from logical qubit below
`s`	`control system` adaptive measurement setting output
`ops`	`control system` byproduct operator output
`ops[0]`	X byproduct operator bit
`ops[1]`	Y byproduct operator bit

When a signal `s` is a bus (a bold line in Fig. 8, consisting of multiple parallel signals), the signals are indexed from 0 upwards, and the range is expressed using square brackets after the signal name. For example, `s[3:0]` is a bus of four signals. Subsets of the signals use the same notation (`s[2:1]` is signal 2 and 1), and a single signal is identified using one index (`s[0]` is signal 0).

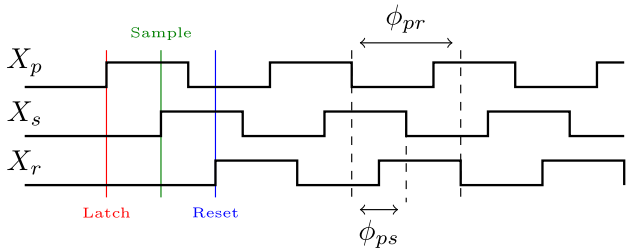

FIGURE 7. Clocks used in the design, and their phase relationships. The frequencies of all three clocks are the same and equal to the photonic clock cycle rate. The relative phases ϕ_{ps} and ϕ_{sr} of the clocks are design parameters, which we investigate in Section VI.

IV. DIGITAL SYSTEM DESIGN

In the following sections, we describe an example digital system design for processing measurement outcomes into adaptive measurement settings and byproduct operators. A schematic overview of the digital system is presented in Fig. 8. A summary of notation used in the diagrams is provided in Table 2.

A. CLOCK PLANNING

We present a design that can process measurements within a single clock cycle, by using three out-of-phase clocks. We consider a system-synchronous design, with the photonic clock X_p being the system clock in the system.

On the rising edge of X_p, the photon arrives in the measurement block, causing a pulse at the output of the single-photon detector. This measurement outcome is amplified and triggers a latch which provides a constant digital signal to the digital system.

The other two clocks, X_s and X_r, are internal to the digital system. On the rising edge of the measurement sample clock X_s, the measurement latch is sampled by the digital system. The rising edge of X_s must be sufficiently offset from the rising edge of X_p so that the output from the latch has settled

IEEE Transactions on Quantum Engineering

to a steady state. This delay must include the time required to amplify the photon detector output.

On the rising edge of the reset clock X_r, the latch is reset ready for the next measurement round. This event must occur after the rising edge of X_s, but before the rising edge of the next photon clock cycle X_p, to satisfy the hold time requirement of the sampling logic.

The computation of the adaptive measurement setting is performed using combinational logic at the earliest possible time that the latch output is valid, on the rising edge of X_s. The measurement setting for the next measurement is then computed and becomes available a short amount of time after the rising edge of X_s, corresponding to the combinational logic delay.

In addition, the byproduct operators are also computed on the rising edge of X_s using combinational logic. The commutation correction, which must be applied at the boundary of a quantum gate, is then computed on the rising edge of X_r because it requires the value of the byproduct operators computed on X_s. The program which controls the measurement pattern is loaded from memory on X_p so that it is ready for the computations that take place on X_s and X_r.

The design of each computational subsystem is described in detail below.

B. ADAPTIVE MEASUREMENT SETTING GENERATION

The most important feature of the adaptive measurement setting s is that it must be present as soon as possible, ready for the next measurement round. The earliest possible time that s can be computed is on the rising edge of X_s. From Fig. 3, the value of s can depend on previous measurement settings and stored byproduct operator values from the current qubit.

A shift register is used to store the past three measurement values,[8] m_0, m_1, and m_2, where m_0 is the most recent measurement outcome. The shift register is loaded sequentially with the next measurement on the rising edge of X_s. The output s is then obtained using a combinational circuit from the shift register; so it is present soon after the rising edge of X_s.

The outputs from the shift register are combined bitwise with a 3-b mask A_m and XORed together to produce the measurement contribution to s. The stored byproduct operators (x_s, z_s) are masked using a 2-b value A_b and XORed to produce a second contribution to s. These two contributions are XORed to produce s itself. Putting together these two contributions gives the following expression for s:

$$s = \left(\bigoplus_{i=0}^{2} A_m[i] m_i \right) \oplus (A_b[1] x_s \oplus A_b[0] z_s)$$

where square brackets denote bitwise access.

The masks A_m and A_b for each measurement round are chosen in such a way that they combine past measurement outcomes and byproduct operators correctly to realize the one-qubit gate, as shown in Fig. 3(b). The CNOT gate has no adaptive measurement settings; so $A_m = A_b = 0$ in that case.

The mask A_m must remain valid through the rising edge of X_p; so it is registered on the rising edge of X_s. The byproduct operator contribution due to A_b is also registered on X_s, so that the byproduct term persists through X_p. These registers are necessary because the program word, which contains the masks (see Section IV-E), is updated on the rising edge X_p.

A disadvantage of this design is that the output s may contain function hazards [25], due to the propagation delays from each of the flip-flops to the output s. These hazards do not affect the digital function of the (synchronous) digital system; however, they may contribute to the power dissipation of the system and/or noise in the analog output, depending on how it is implemented. In order to avoid the hazards, the output s could be registered; however, this would require another clock edge soon after X_s to preserve the setup time of the analog output stage.

The adaptive system is shaded in purple in Fig. 8(b).

C. BYPRODUCT OPERATOR CALCULATION

The byproduct operators must be updated after each measurement round. Since they only depend on the measurement outcomes, they can also be computed on the rising edge of X_s.

The byproduct operators comprise two bits (x, z), which are updated according to the measurement outcomes from the current logical qubit, $m_0^{(1)}$, and the two neighboring logical qubits, $m_0^{(2)}$ above and $m_0^{(0)}$ below. Any of these three measurements may be XORed in any combination, together with the old byproduct operator values (x, z), to produce new (x', z'). Two 3-b masks B_x and B_z control of which the three measurement outcomes should be XORed together to produce the updated x and z, so that the byproduct operators are obtained using the following equations:

$$x' = x \oplus \left(\bigoplus_{j=0}^{2} B_x[j] m_0^{(j)} \right)$$

$$z' = z \oplus \left(\bigoplus_{j=0}^{2} B_z[j] m_0^{(j)} \right).$$

The masks B_x and B_z for each measurement round are chosen in such a way that they combine measurement outcomes from the current and surrounding logical qubit rows to form the updates to the byproduct operators that are shown in Fig. 3.

It is sometimes necessary to add a constant [the 1 in (1) for z_c'] to the byproduct operators, as in the case of the CNOT pattern. This constant addition is controlled by the commutation correction program, as described in the following section.

The main byproduct operator calculation is shaded in Fig. 8(b).

[8] For more complicated measurement patterns, it may be necessary to store more than three measurements. However, for the arbitrary one-qubit gate and CNOT gate, three measurements are sufficient.

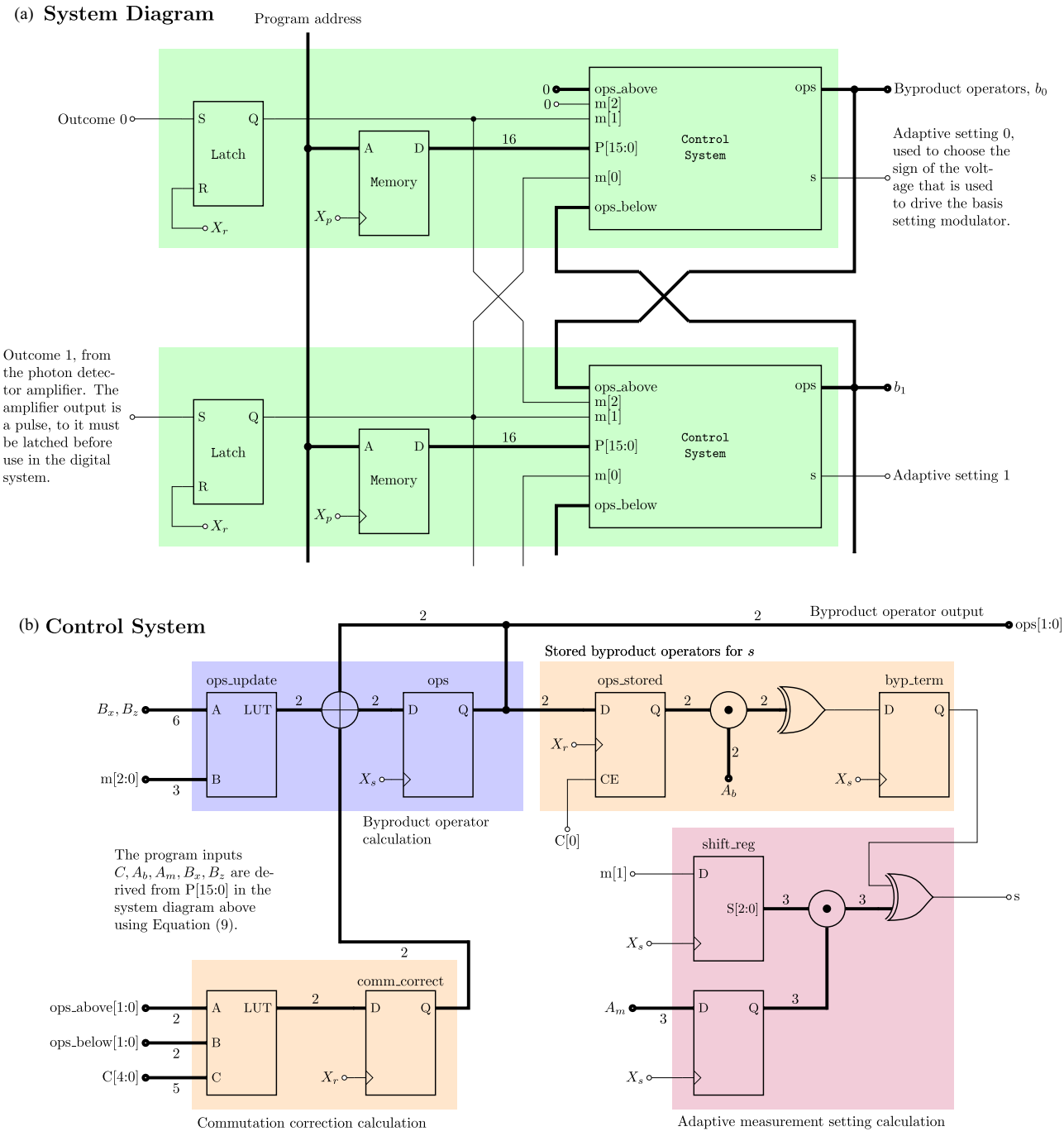

FIGURE 8. (a) Digital system diagram for multiple qubits. The "unit cell" for each logical qubit (shaded green) has a measurement latch, a program memory, and a control system for calculating measurement settings and byproduct operators. (b) Design of the control system. In the high level schematic diagram of the control system, buses are denoted with bold lines, and the bus width is written next to the wire. The circles apply bitwise operations between their inputs: the cross stands for XOR and the dot stands for AND. The right port of the circle is the output, and all other ports are inputs. The logic gates are multi-input, with inputs from all the buses and wires connected on their left (i.e., wires inside a bus will be combined in the logic operation). Each part of the diagram is shaded according to its function, using the same coloring as in Fig. 3. Flip-flops are clocked on the rising edge of their clock input, and elements whose output is LUT represent combinational logic. Reset signaling is omitted from the diagram for simplicity.

D. COMMUTATION CORRECTIONS

For the CNOT gate, the commutation correction is performed by mixing the values of the byproduct operators between the control and target logical qubits, as described in (5).

For an arbitrary one-qubit gate, the correction is more complicated, requiring the use of the byproduct operators in the calculation of the measurement settings. However, in order to avoid overwriting these correctional byproduct operators prematurely, it is necessary to store them in a separate register, called the stored byproduct operator register. The correction for the one-qubit gate then amounts to loading this register from the current byproduct operators.

IEEE Transactions on Quantum Engineering

TABLE 3. Interpretation of the Bit Fields of *C*, Which Controls the Commutation Correction for the Arbitrary One-Qubit Gate and CNOT Gate, and Also Controls the Addition of Constants to the Byproduct Operator

Bit	Meaning
0	If high then store the byproduct operators
1	If high then a commutation correction is necessary, in which case:
2	If high then current logical qubit is the control
3	If high then other qubit in CNOT is above
4	If high then add ones to the byproduct operators, in which case:
2	Constant value to add to z
3	Constant value to add to x

The meaning of bits 2 and 3 depend on whether bits 1 or 4 are set, which are mutually exclusive. If $C = 0$, then no operation is performed.

TABLE 4. Example Two-Qubit Computation Comprising a One-Qubit Gate $U = R_x(0.3)R_z(0.2)R_x(0.1)$ on Qubit 0, Followed by a CNOT Between Qubits 0 and 1 (Qubit 0 is the Control)

		Qubit 0					Qubit 1				
Gate	A	m_0	P_0	θ_0	s_0	b_0	m_1	P_1	θ_1	s_1	b_1
U	0	0	0302	0	0	00	0	0002	0	0	00
	1	1	0510	-0.1	1	10	1	0010	0	0	10
	2	1	0342	-0.2	1	11	0	0002	0	0	10
	3	0	3010	-0.3	0	11	1	5010	0	0	00
CNOT	4	1	0003	0	0	10	0	0002	0	0	10
	5	0	0010	$\pi/2$	0	10	1	0030	0	0	00
	6	0	a013	$\pi/2$	0	10	0	0022	0	0	00
	7	1	0002	0	0	10	1	0010	0	0	10
	8	1	0012	$\pi/2$	0	01	0	0002	0	0	10
	9	1	0010	$\pi/2$	0	11	0	0010	0	0	10

The program P_i (written in hexadecimal in the table) combines the measurement outcomes m_i (randomly generated) to produce the adaptive measurement setting s_i and the byproduct operators b_i (the least significant bit is z) for the ith qubit. The basis measurement angles θ_i are included for completeness (s_i is combined with θ_i to produce the measurement angle ϕ_i).

Both these corrections, for the CNOT and the one-qubit gate, require the byproduct operator values and must therefore be calculated on the rising edge of X_r rather than X_s. The behavior of this correction is controlled by a 5-b value C, whose interpretation is shown in Table 3.

Most of the time, $C = 0$ and the commutation correction does nothing. It is only directly before gate boundaries that a commutation correction must be performed.

The commutation corrections are shaded in orange in Fig. 8(b).

E. PROGRAM WORD

The digital system is controlled using a 16-b program word P which is formed by concatenating the masks and control bits in the previous sections as follows:

$$P = CA_bA_mB_xB_z. \tag{9}$$

Each logical qubit requires its own set of program words, one per measurement round.

Table 4 shows an example calculation for the two qubit circuit containing an arbitrary one-qubit gate $U = R_x(0.3)R_z(0.2)R_x(0.1)$ on the first qubit, followed by a CNOT gate between the first and second qubit. The table contains randomly chosen measurement outcomes and the resulting adaptive measurement settings and byproduct operators that result from the measurement pattern, including the program word that is used to make the calculations.

It is clear that the program word could be compressed to save on memory usage. In our example design, we have prioritized program simplicity over memory usage.

TABLE 5. Utilization of Flip-Flops, Lookup Tables and Input/Output Pads (I/O) in the Design, for $N = 1$ Logical Qubit and $N = 20$ Logical Qubits, for the Control System (CS) in Fig. 8 and the Full Design

	Flip-flops			Look-up tables			Input/output	
N	CS	Full	Util.	CS	Full	Util.	Full	Util.
1	10	24	0.03 %	5	11	0.03 %	8	2.8 %
20	237	476	0.89 %	137	364	0.58 %	84	29.5 %

The proportion of device resources is included in the utilization (Util.) columns.

F. FPGA IMPLEMENTATION OF THE DESIGN

In order to analyze the timing characteristics of the system, we wrote an FPGA implementation of the design using the VHSIC Hardware Description Language very high-speed integrated circuit hardware description language (VHDL), targeting a Xilinx Kintex-7 FPGA (part no. xc7k70tfbg484-2). We used the synthesis tool Xilinx Vivado 2020.2 to implement the design and perform static timing analysis.

We used the mixed-mode clock manager (MMCM) [26] to generate the two out-of-phase clocks X_s and X_r from the (external) system clock X_p. The program was stored in memory generated by an instance of the distributed memory generator IP [27], configured as ROM so that we could store the program in a coefficient file for the purpose of verifying the design.

The utilization of logic and input/output (I/O) pads in the design is provided for 1 logical qubit and 20 logical qubits in Table 5. The data were obtained from the utilization report generated by Vivado after implementing the system for each number of logical qubits. The number of logic elements scales more than linearly between 1 and 20 logical qubits because the synthesis tool optimizes away logical qubit interconnects in the single logical qubit case. However, the overall utilization of flip-flops and lookup tables in the design is very low ($< 1\%$ of device resources) because the calculations involved in the design are quite simple.

The use of I/O pads is quite high, due to the need for one measurement input m, one adaptive measurement setting s, and two byproduct operator lines per logical qubit. In our design, the total number of I/O pads required is

$$K = 4N + 4$$

where N is the number of logical qubits. This includes four common signals: the input clock X_p; the clock-is-locked output signal from the MMCM; a reset signal; and an enable signal. By accessing the byproduct operators via a low speed serial interface, it would be possible to reduce this pin count to

$$K \sim 2N$$

which includes only the measurement inputs m and adaptive measurement setting outputs s. On the largest FPGA in the 7-series family [28], the Virtex-7 xc7v2000t device (which

has 1200 user I/O pads), this provides an upper bound on the number of logical qubits (cluster state rows) of $N \sim 600$.

Input/output delays are also a bottleneck for performance in the FPGA design, as we show in Section VI. The Xilinx 7-series devices were chosen because they have a level-sensitive latch built into their input logic slice (LCDE) [29], which forms the first stage of the digital system. In our design, the availability of I/O resources on the FPGA device places a greater restriction on the scalability of the design than the utilization of logical resources does. However, when the system is scaled to maximally utilize the I/O resources of the target device, it is likely that timing constraints in addition to those shown in Section VI would follow from routing delays inherent in managing this I/O bottleneck at large numbers of qubits.

A disadvantage of the design is that it is not possible to place the output s in the output logic slice because there is combinational logic between the final register and the output port [29]. It is also not possible to place the byproduct operator registers in output logic slices because the output is rerouted to the internal FPGA fabric for use in updating the byproduct operators [see the feedback loop in Fig. 8(b)].

As we show in Section VI, the clock frequency is not a bottleneck in the system; so it may be possible to create another design with multicycle latency, where the outputs are stored in separate registers and eligible for placing in the output logic slice. This may remove some of the output delay and allow a slightly higher clock frequency. It would also remove the logic hazards present in the output s.

V. VERIFICATION OF THE DESIGN

Due to the nonintuitive nature of the measurement patterns and the complexity of the digital hardware design, it is not possible to verify the functional correctness of the design simply by looking at the output of simulations. This section describes the verification of the measurement patterns and the program logic, and also the hardware design.

A. MEASUREMENT-BASED QUANTUM COMPUTING SIMULATOR

We wrote an MBQC simulator in C++ for the purpose of generating data to verify the digital system design. The program can simulate a cluster state containing up to 14 logical qubits by only holding two columns of the cluster state in memory at any one time.

The program is designed to mimic the operation of the hardware, using the program word P to process measurement outcomes and apply quantum operations to the simulated quantum state according to the resulting adaptive measurement settings. At the end of the quantum circuit, the byproduct operators are applied to the state to obtain the result from the quantum computation.

The quantum circuit is also performed in the gate-based model on a state vector containing the same number of logical qubits. At the end, this state vector is compared with that obtained from the cluster state computation, to check that they agree with each other.

The state of the cluster state simulator at each measurement round is written to a file, which is used as the input to the hardware simulator. It contains the program word and the measurement outcomes, which are the inputs to the digital system. It also contains the value of the adaptive measurement settings, the byproduct operators and the stored byproduct operators, which are the outputs from the digital system.

B. VHDL TESTBENCHES

The function of the digital system was verified using testbenches written in VHDL. The testbenches read stimulus and output data from the simulation output file described in the previous section.

The output from the system, the adaptive measurement setting, and the byproduct operators are compared with the values from the simulation file. The simulation passes if all the values are equal, which is tested automatically. An example waveform output from the testbench, for a single logical qubit, is shown in Fig. 9.

VI. TIMING ANALYSIS

We used static timing analysis to establish the maximum operating frequency of the design and to obtain the input/output delays associated with the system. The critical path is made up of two components:

1) the path from the input port m (clocked on the rising edge of X_p) to the byproduct operator register (loaded on the rising edge of X_s);
2) the path from the shift register output (loaded on the rising edge of X_s) to the output port s (clocked on the rising edge of X_p).

By modifying the phase shift of X_s relative to X_p (ϕ_{ps} in Fig. 7), it is possible to allocate more time to one path or the other. The phase of X_r (ϕ_{pr} in Fig. 7) must also be adjusted to allow timing closure of paths between the X_s and X_r clock domains. We established the maximum operating frequency F_{max} of the system by manually adjusting the phase of X_s and X_r to balance the worst negative setup slack between the critical paths, while increasing the frequency of the design, until both paths fail to meet timing. Using this method, we obtained $F_{\text{max}} = 190$ MHz using $\phi_{\text{ps}} = 220^\circ$ and $\phi_{\text{pr}} = 300^\circ$. The phase difference 80° between X_s and X_r represents the amount of the time taken for the internal FPGA logic to process the latched measurement outcome before it is reset.

We then performed the timing analysis at each frequency between 10 and 190 MHz, in steps of 10 MHz, to establish the most generous input and output constraints that still allow timing closure at each frequency. All input/output constraints are expressed with respect to the external clock X_p (the system clock).

The input constraint is specified by the clock-to-out time t_{co} of the input signal m, which is equal to the time delay between the rising edge of X_p and the pulse generated by the

IEEE Transactions on Quantum Engineering

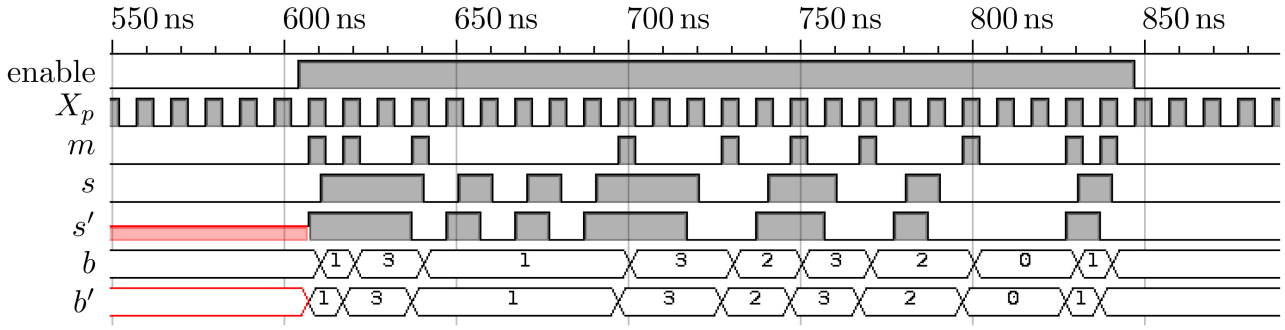

FIGURE 9. Example of the verification method we used to establish the working of the system. The outputs s and b are compared with the true values s' and b' from the simulation file. It is clear from the figure that the output from the digital system agrees with the simulation file in each clock cycle. The numbers in the rows b and b' are the result of concatenating the two byproduct operator bits in the format xz and interpreting the result as an integer. (The hardware outputs slightly lag the true values because the file is loaded on X_p in the testbench, whereas the design outputs the measurement settings and byproduct operators on the rising edge of X_s.)

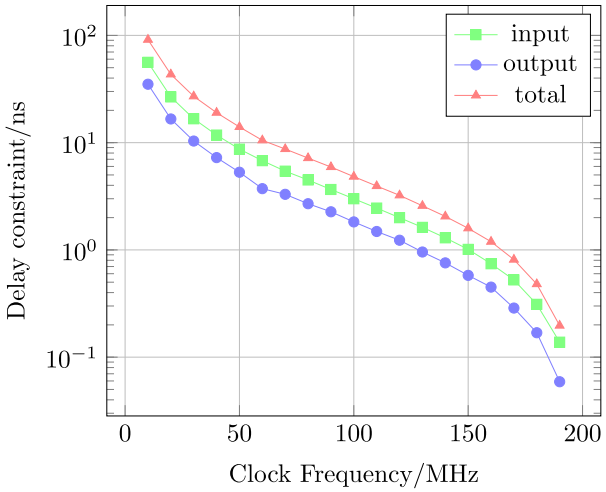

FIGURE 10. Most generous input and output delay constraints that allow implementation of the design at each frequency. The total delay, which can be apportioned between input and output analog systems by adjusting the phase of X_s, represents the maximum amount of time available to the analog system shown in Fig. 6.

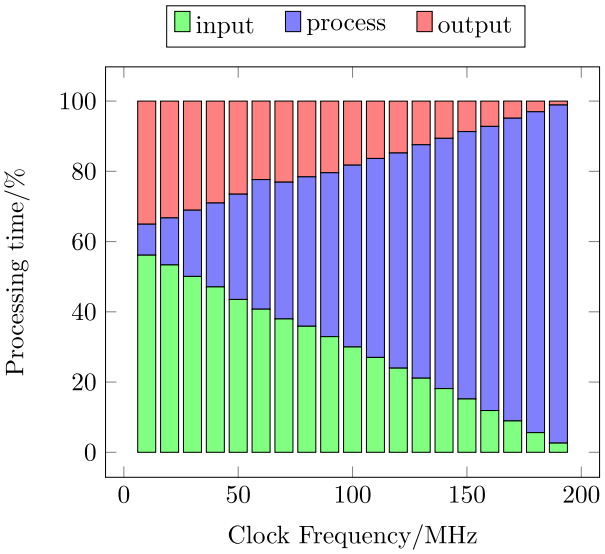

FIGURE 11. Proportion of the clock cycle devoted to processing the adaptive measurement settings and the byproduct operators, as a function of photon clock frequency. At the higher frequencies, nearly all of the cycle is spent processing the measurements, leaving almost no time for the analog amplification at the input and output (shown in green and red).

input analog system at m. This time constrains the analog characteristics of the single-photon detector amplifier.

The output constraint is the setup time t_{su} of the output signal s with respect to the system clock X_p, which is the delay between the time that s transitions at the boundary of the FPGA and the next rising edge of X_p. This time determines the required operating speed of the output DAC system and modulator drivers, which must be able to set the voltages of the modulators before the next photon arrives on the rising edge of X_p.

The input/output timing constraints are plotted as a function of frequency in Fig. 10. The input constraint is systematically more generous than the output constraint because of the choice of phase of X_s. The sum of the input and output constraints must be less than the total input/output slack, also shown in the figure.

Fig. 11 shows a graph of the proportion of the clock cycle X_p taken up with digital processing, as a function of frequency. It is clear that at higher frequencies, the digital processing dominates the clock cycle, leaving very little time for the analog amplifier systems.

At a representative clock frequency of 150 MHz, the photons would need to be delayed for 6.67 ns in either an optical fiber or a waveguide delay line. Assuming a standard silicon-on-insulator (SOI) platform, the delay line must be approximately 83 cm, assuming a mode index of ~ 2.4 [21].

We reimplemented the design targeting a higher end FPGA (Xilinx Kintex Ultrascale+, part no. xcku5p-ffvd900-3-e) to see whether the maximum clock frequency could be improved. We found that the maximum clock frequency increased to $F_{\max} = 220$ MHz using $\phi_{ps} = 140^\circ$ and $\phi_{pr} = 230^\circ$. In this case, at the maximum clock frequency, less time is allocated to the input analog system compared with the 7-series FPGA. The phase difference of 90° between X_s and X_r indicates that

approximately the same time (1.125 ns) is taken by the internal digital system compared with the 7-series FPGA (1.152 ns).

VII. EXTENDING THE DESIGN

There are several improvements that could be made to the implementation we have presented in this article.

It is likely that a performance improvement could be obtained by implementing the digital design using an ASIC. Critical path delays due to logic have been found to decrease by 3–4 times in standard-cell ASIC designs [30]. However, this may not translate to a performance improvement in this design because the majority of the critical path delays come from the input/output buffers, not the logic. To improve this, it may be possible to utilize very high speed latches and output buffer designs, with delays on the order of 100 ps [31]. A full analysis of the input/output buffer delays should be performed in tandem with the design of the input/output analog systems, to ensure compatibility between the two systems. At this point, the requirement for absolute synchronization between the cluster state generator and the digital control system, using a system synchronous architecture [32] may become the bottleneck to the design. Such schemes are often limited to speed up to 200–300 MHz, due to clock skew and data path delays [33].

For measurement patterns like the CPhase gate [17, Section IV-C], it is necessary to be able to arbitrarily reorder the measurements between pairs of columns. This is to satisfy the requirement that measurement outcomes are always available before any dependent adaptive measurement settings are required. Depending on the complexity of the measurement pattern, it may be necessary to merge more than two measurement columns. Extending the implementation to support these situations would increase the number of gates (measurement patterns) that are realizable with the system.

The design could be extended to support nonlocality of the byproduct operator calculation. In the design discussed in this article, the byproduct operators depend only on measurement outcomes from adjacent logical qubits. However, there are measurement patterns for which byproduct operators for a given logical qubit may depend on cluster qubits that are further away [17, Section IV-C]. This may lead to a routing problem in FPGA and ASIC designs, especially as the number of qubits increases, which are important to quantify.

It is clear that there are many realistic features of the design that we could have incorporated into our model, but have not. For example, a realistic system would have to account for the finite efficiency and dead-time inherent in single photon detectors [34]. This dead time would place a lower bound on the photonic cycle time of the system, and the nonunit efficiency would introduce the need for error correction even in our simple model.

Ultimately, although the architecture we have analyzed in this article is highly idealized, it still shows that, even in this simple scenario, classical electronics places significant constraints on the design of photonic quantum computers. A more realistic implementation would incorporate the effects of fault-tolerance, error correction, photon loss, and detector deficiencies. Producing such an implementation would require the specification of the architecture, the error models, and the device models, as part of its underlying assumptions. A much more thorough and wide-ranging design analysis would, therefore, be necessary to understand how all elements of the resulting design interact to place constraints on the operation of the photonic quantum computer.

VIII. DISCUSSION

Although we have analyzed a highly simplified scenario, there are important takeaways which we believe will apply to any implementation of photonic quantum computing.

In contrast to every other approach to building quantum computers, photonic MBQC relies on manipulating and measuring flying-qubit states. This means that the effective "lifetime" of a qubit in these platforms is ultimately bounded by the length of time that photons can be kept circulating inside an optical delay line, either on- or off-chip. The fact that the spatial and temporal properties of the system cannot be decoupled is at the root of many of the unique timing constraints that photonic approaches need to satisfy. This is in contrast to other matter-based systems where the qubit lifetime is, to first order, unrelated to its spatial footprint.

In an integrated photonic approach, the only way to get longer qubit lifetimes is by increasing the length of the on-chip delay line. Even with a high index contrast platform like SOI, which allows low-loss bend radii $< 5\mu$m, getting realistic delays beyond 2–3 ns is extremely challenging, both due to the increasing insertion loss (1–2 dB cm^{-1}) and the increasing on-chip footprint (spiral delay lines with lengths $\sim$ 10 cm) [21]. One solution to the timing constraints is to use an integrated photonic quantum memory [35] which would make photonic MBQC implementations closer to their matter-based counterparts by allowing one to map quantum information on to a long-lived spin/hyperfine transition.

Longer delays can be obtained in principle by using low-loss (less than 0.2 dB km^{-1}) optical fibers off-chip, which is being currently considered in the context of FBQC [15], although this approach is not without its own tradeoffs. Losses in the grating couplers involved in getting the light on and off the chip must be accounted for, in addition to losses involved in the optical switching network needed to get the cluster states to the grating couplers. These requirements can, in principle, be satisfied by state-of-the-art lithium niobate modulators; however, the size and form factors are not really suitable for very large-scale integration, which is a critical requirement from a systems perspective. State-of-the-art silicon modulators are very far from ideal, especially in terms of insertion loss ($\sim$ 6dB/device) [36]. Often, architectural proposals for photonic quantum computing are highly theoretical [12], [13], [15], by which we mean that it is often quite difficult for an engineer to see how to construct the hardware components required for the architecture. It is our belief that a thorough study of the implementation details of

these schemes is necessary to fully establish the feasibility of these proposals for the realization of quantum computers.

In the design we consider here, there is almost no algorithmic complexity involved. The system must perform a few arithmetic operations per clock cycle, which are hardcoded using a program word. In every other approach to photonic quantum computing, there is a substantial increase in algorithmic complexity, for example, due to the need to either search for paths in an incomplete cluster state (using breadth-first search, or some equivalent search process) [37], [38] or perform calculations relating to error correction [39]. We believe it is highly likely that these systems will be constrained by digital system imposed timing constraints, arising solely due to the increased complexity of performing the algorithms involved in real time. Any difficulties of this kind involved in implementing these approaches can only be understood by analyzing candidate implementations, in a similar manner to the way we have analyzed our simple system.

IX. CONCLUSION

We have provided a practical description of the measurement patterns for one-qubit gates and the CNOT gate and shown in detail how to implement a digital control system for photonic MBQC, in the presence of an ideal path-encoded photonic cluster state generator. It is clear from the timing analysis of our FPGA implementation of this system that it places substantial constraints on the input and output analog systems needed at the interface between the classical and quantum subsystems. For example, at a photon clock frequency of 150 MHz, the total time available for the input and output analog processing is 1.59 ns out of the total period of 6.67 ns. The remaining 5.08 ns is consumed by the logic delays inside the FPGA design. At the same time, a photon clock period of 6.67 ns corresponds to a long delay line ($\sim$ 83 cm), which will occupy quite a large footprint in an integrated implementation of photonic MBQC.

While in this work we have implemented a proof-of-principle design to study the constraints, it is clear that the digital system and implementation can be further optimized. For example, since the maximum frequency of our design is less than 200 MHz and the maximum clock frequency of the target FPGA is greater than 600 MHz, it may be possible to create a multicycle digital design so as to properly register the inputs and outputs and place them in dedicated input/output slices. This would likely increase the maximum clock frequency somewhat while maintaining the input/output delay constraints.

Incorporating the features of a realistic photonic quantum computing system, based on any of the modern approaches to photonic quantum computing, adds further algorithmic complexity to the design. Our work provides a building block for quantum engineers to produce detailed, verifiable, and open-source implementations of these systems to establish what electronic control system constraints are present in those cases.

APPENDIX A
GATE-BASED QUANTUM COMPUTING

This appendix contains a brief overview of quantum computing in the gate-based model. The basic unit of quantum computation is the qubit, which is a two-state system, analogous to a bit, except complex linear combinations of the zero-state (denoted $|0\rangle$) and the one-state (denoted $|1\rangle$) are also valid states. The states $|\psi\rangle$ of a qubit can be expressed as

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad a \in \mathbb{R}, b \in \mathbb{C}. \tag{10}$$

The qubit can only ever be observed in the state $|0\rangle$ or $|1\rangle$, with probabilities given by the ratio of $|a|^2$ to $|b|^2$. The act of observing the qubit is called a measurement. The absolute values of a and b have no independent physical meaning; so the condition $|a|^2 + |b|^2 = 1$ is imposed so that the probabilities are equal to $|a|^2$ and $|b|^2$. Likewise, the arguments of the complex numbers a and b have no physical meaning; so it is possible to impose $a \in \mathbb{R}$ without loss of generality. The argument of b is then the relative phase between $|0\rangle$ and $|1\rangle$.

The states of a single qubit can be identified with points on the surface of a sphere, called the Bloch sphere, as shown in Fig. 2. The mapping between the a and b and the real number angles θ and ϕ is given by the following identity:

$$a|0\rangle + b|1\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle.$$

The angle ϕ in the equator of the Bloch sphere is the relative phase between $|0\rangle$ and $|1\rangle$, and the angle θ controls the probability of observing $|0\rangle$ or $|1\rangle$ upon measurement.

1) ONE-QUBIT GATES

The state of the qubit can be changed by applying a quantum gate. The valid gates on a single qubit, called one-qubit gates, are those which correspond to a rotation of the points on the Bloch sphere about any axis, by any angle. The gates that perform rotations of the state about the x, y, and z axes are denoted as $R_x(\alpha)$, $R_y(\alpha)$, and $R_z(\alpha)$, where α is the angle of rotation according to the right-hand rule. An arbitrary one-qubit rotation can be formed by applying x- and z-rotations in sequence as $R_x(\zeta)R_z(\eta)R_x(\xi)$ (applied from right to left). This follows from the decomposition using Euler angles of an arbitrary rotation into x- and z-rotations.

2) MEASUREMENT

When a qubit is measured, it always collapses to either the state $|0\rangle$, with probability $|a|^2$, or the state $|1\rangle$, with probability $|b|^2$. This is called a computational basis measurement.

However, it is possible to generalize the concept of measurement so that an "observation" causes the qubit to collapse into the state $|0'\rangle$ or the state $|1'\rangle$, which are any two antipodal points on the Bloch sphere, joined by a line L. This observation is made by using one-qubit gates to transform the line L to the line through $|0\rangle$ and $|1\rangle$, and then making a computational basis measurement. For example, to measure along the line denoted by L in Fig. 2, it is necessary to apply a z-rotation $R_z(-\phi + \pi/2)$ to align the state $|0'\rangle$ with the

positive y-axis, followed by an x-rotation $R_x(\pi/2)$ to obtain $|0\rangle$.

It is possible to measure along any line in this way by applying an arbitrary one-qubit gate $R_z(\alpha)R_x(\beta)R_z(\gamma)$ and then measuring in the computational basis. It is important to realize that general measurements involve the application of a one-qubit gate before making a computational basis measurement.

3) TWO-QUBIT GATES

The states of two qubits can be expressed analogously to (10) as

$$|\psi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \tag{11}$$

where $a \in \mathbb{R}$ and $b, c, d \in \mathbb{C}$. The sum is over all the four possible states that the two qubits could be observed in. As with the single-qubit case, $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$ is imposed, and the probability of obtaining, for example, $|01\rangle$, is given by $|b|^2$.

There is no equivalent of the Bloch sphere for graphically presenting the states of two qubits. An example of a two-qubit gate is the CNOT gate. The action of this gate on the state (11) above is

$$\begin{aligned} &|\psi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \\ \mapsto\ &|\psi\rangle = a|00\rangle + b|01\rangle + c|11\rangle + d|10\rangle \end{aligned} \tag{12}$$

that is, the states $|10\rangle$ and $|11\rangle$ are reversed. The interpretation of this gate is that the first (leftmost) qubit controls whether a NOT gate is applied to the second (rightmost) qubit. The first qubit is called the control qubit, and the second qubit the target.

Analogously to the way that a NAND gate is universal for digital logic, the CNOT gate combined with the basic rotations $R_x(\alpha)$, $R_y(\alpha)$, and $R_z(\alpha)$ are universal for quantum computation. To build up any complicated computation, all that is required is to apply the correct string of one- and two-qubit gates, one after the other, to a set of qubits. For example, in Fig. 1(c), an arbitrary one-qubit gate $U = R_x(\zeta)R_z(\eta)R_x(\xi)$ is applied to the top qubit, and a CNOT gate is applied between the bottom two qubits.

APPENDIX B
CNOT MEASUREMENT PATTERN

We use a reduced measurement pattern for the CNOT gate that only uses two rows of cluster qubits, instead of the three row pattern in [17]. The pattern is derived using the same method outlined in Section II-G7 of that paper for the calculation of the three-row CNOT gate. In order to explain the derivation, we begin by discussing some technical aspects of cluster states and describe what it means for a measurement pattern to realize a gate.

A cluster state $|\phi_C\rangle$ on N qubits is created by placing all the qubits in the $|+\rangle$ state, and then applying CZ gates between each pair of qubits that should have an entanglement link (shown as red line segments in Fig. 12). It can be shown [17]

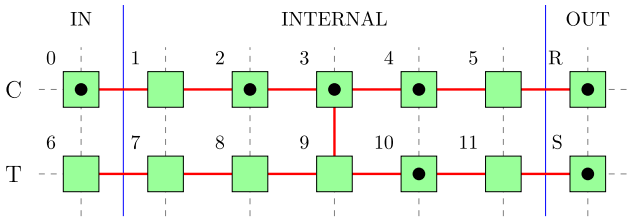

FIGURE 12. Labeling of the cluster qubits for the purpose of deriving the CNOT measurement pattern. When a gate is realized in MBQC, the input state starts on the IN column and is teleported to the OUT column *R* and *S* by applying the measurement pattern. The black dots show the location of the correlation operators K_a in (20).

that cluster states satisfy the eigenvalue equations

$$K_a|\phi_C\rangle = \left(X_a \prod_{b\sim a} Z_b\right)|\phi_C\rangle = |\phi_C\rangle \tag{13}$$

where the first equality defines the correlation operator K_a on the cluster qubit a. There is one such equation for each cluster qubit a, and in each equation, the product is over all other neighboring cluster qubits b joined by red line segments to a (denoted as $b \sim a$).

To state what it means for a measurement pattern to realize a gate G, we use the arrangement of qubits shown in Fig. 12, on which the CNOT measurement pattern is defined. Instead of placing all the qubits in the $|+\rangle$ state, assume qubits 0 and 6 (the IN qubits) are in an arbitrary state $|\phi\rangle$. As before, place all the other qubits (including the OUT qubits) in the $|+\rangle$ state and apply CZ gates wherever there are red line segments in Fig. 12. Now, after the measurement pattern for the CNOT gate has been applied, meaning that all the IN and INTERNAL qubits have been measured out, there remains a two-qubit state $|\psi\rangle$ on the OUT qubits R and S. The sense in which the measurement pattern has realized the gate G is that input and output states are related by

$$|\phi\rangle = BG|\psi\rangle \tag{14}$$

where B is the byproduct operator for the measurement pattern. In other words, the measurement pattern has the effect of moving the state of the IN column to the OUT column, and transforming it according to the gate which is being realized by the measurement pattern.

The measurement pattern for the CNOT gate is obtained by using a theorem [17, Theorem 1] that relates eigenvalue equations derived from (13) and a given measurement pattern, to the gate G which that measurement pattern realizes. The content of the theorem is that it is only necessary to check how a cluster state $|\phi_C\rangle$ is affected by the measurement pattern (where the state of qubits 0 and 6 are $|+\rangle$) in order to establish that the measurement pattern works for any other IN state $|\phi\rangle$. In the interest of simplicity, We state the theorem for the case of a two-qubit gate G like the CNOT gate.

Theorem 1: Suppose that a cluster state $|\phi_C\rangle$ is prepared on the pattern of 14 qubits shown in Fig. 12, for the purpose of realizing a two-qubit gate G acting on logical qubits labeled C and T. Suppose that a set of measurements M is

performed on the INTERNAL cluster qubits 1–5 and 7–11, resulting in a state $|\psi_C\rangle$ on the remaining qubits (0, 6, R, and S), which satisfies the following sets of eigenvalue equations:

$$X_0 \left[G X_C G^\dagger \right]_{R,S} |\psi_C\rangle = (-1)^{\lambda_x} |\psi_C\rangle$$
$$Z_0 \left[G Z_C G^\dagger \right]_{R,S} |\psi_C\rangle = (-1)^{\lambda_z} |\psi_C\rangle \quad (15)$$

and

$$X_6 \left[G X_T G^\dagger \right]_{R,S} |\psi_C\rangle = (-1)^{\mu_x} |\psi_C\rangle$$
$$Z_6 \left[G Z_T G^\dagger \right]_{R,S} |\psi_C\rangle = (-1)^{\mu_z} |\psi_C\rangle. \quad (16)$$

Then the measurement pattern in which the inner qubits are measured according to M, and the IN cluster qubits 0 and 6 are measured in the X-basis, realizes the gate GB, where the byproduct operators B for the logical qubits C and T are given by

$$(x_C, z_C) = (\lambda_z, m_0 + \lambda_x)$$
$$(x_T, z_T) = (\mu_z, m_6 + \mu_x) \quad (17)$$

where m_a is the outcome of the measurement of the ath cluster qubit. ■

The square bracketed terms in (15) and (16) are computed in terms of the logical qubits C and T, without reference to cluster qubits. Any terms involving C and T are then interpreted as applying to the cluster qubits R and S. For example, when $G = \text{CNOT}$

$$\left[G X_T G^\dagger \right]_{R,S} = [X_C X_T]_{R,S} = X_R X_S.$$

To apply the theorem to the CNOT gate, it is therefore necessary to obtain the following eigenvalue equations:

$$X_0 \, (X_R X_S) \, |\psi_C\rangle = (-1)^{\lambda_x} |\psi_C\rangle$$
$$Z_0 \, (Z_R) \, |\psi_C\rangle = (-1)^{\lambda_z} |\psi_C\rangle \quad (18)$$

and

$$X_6 \, (X_S) \, |\psi_C\rangle = (-1)^{\mu_x} |\psi_C\rangle$$
$$Z_6 \, (Z_R Z_S) \, |\psi_C\rangle = (-1)^{\mu_z} |\psi_C\rangle. \quad (19)$$

To obtain these equations, begin with the cluster state $|\phi_C\rangle$ on the two-row CNOT shape shown in Fig. 12, and multiply together the correlation operators in (13) so as to obtain the following four equations:

$$\begin{aligned} |\phi_C\rangle &= K_0 K_2 K_3 K_4 K_R K_{10} K_S |\phi_C\rangle \\ &= -X_0 Y_2 X_3 Y_4 X_R X_{10} X_S |\phi_C\rangle \\ |\phi_C\rangle &= K_1 K_2 K_4 K_5 |\phi_C\rangle \\ &= Z_0 Y_1 Y_2 Y_4 Y_5 Z_R |\phi_C\rangle \\ |\phi_C\rangle &= K_6 K_8 K_{10} K_S |\phi_C\rangle \\ &= X_6 X_8 X_{10} X_S |\phi_C\rangle \\ |\phi_C\rangle &= K_4 K_5 K_7 K_9 K_{11} |\phi_C\rangle \\ &= Y_4 Y_5 Z_R Z_6 X_7 X_9 X_{11} Z_S |\phi_C\rangle. \end{aligned} \quad (20)$$

The right-hand sides are obtained by repeated application of the equation $X_a Z_a = iY_a = -Z_a X_a$. Note that Pauli operators on different qubits commute.

As with any pattern derived using this method, the choice of operators K_a in the above equations is motivated by two goals.

1) The equations must contain the correct IN and OUT terms in (18) and (19). These terms are colored red in the equations.
2) The Pauli operators on the INTERNAL cluster qubits agree between all the equations. That is, for each cluster qubit a, only X_a or Y_a appears across all the equations. For example, when $a = 4$, only Y_4 appears (three times, shown in blue), and there are no instances of X_4. It is these operators that define the measurement bases M for each qubit a in the INTERNAL group of cluster qubits.

When the INTERNAL qubits are measured according to M, the Pauli terms disappear [19, Section 10.5.3], and each one contributes a sign according to its measurement outcome m_a, to give the following equations on the reduced state $|\psi_C\rangle$:

$$\begin{aligned} X_0 X_R X_S |\psi_C\rangle &= (-1)^{1+m_2+m_3+m_4+m_{10}} |\psi_C\rangle \\ Z_0 Z_R |\psi_C\rangle &= (-1)^{m_1+m_2+m_4+m_5} |\psi_C\rangle \\ X_6 X_S |\psi_C\rangle &= (-1)^{m_8+m_{10}} |\psi_C\rangle \\ Z_6 Z_R Z_S |\psi_C\rangle &= (-1)^{m_4+m_5+m_7+m_9+m_{11}} |\psi_C\rangle. \end{aligned}$$

These equations are in the form of (18) and (19), and define the values of λ_x, λ_x, μ_x, μ_z in terms of the measurement outcomes m_a. As a result, it follows from the theorem above that the measurement pattern consisting of M, plus X measurements on the IN qubits, realizes the gate (CNOT)B, where the byproduct operator B found using (17) to be

$$\begin{aligned} (x_C, z_C) = \; & (m_1 + m_2 + m_4 + m_5, \\ & 1 + m_0 + m_2 + m_3 + m_4 + m_{10}) \\ (x_T, z_T) = \; & (m_4 + m_5 + m_7 + m_9 + m_{11}, \\ & m_6 + m_8 + m_{10}). \end{aligned} \quad (21)$$

Finally, the byproduct operator can be commuted past the CNOT gate to obtain

$$\begin{aligned} & \left(Z_C^{1+m_0+m_2+m_3+m_4+m_6+m_8} X_C^{m_1+m_2+m_4+m_5} \right. \\ & \left. Z_T^{m_6+m_8+m_{10}} X_T^{m_1+m_2+m_7+m_9+m_{11}} \right) \text{CNOT}. \end{aligned} \quad (22)$$

The contributions to the byproduct operators given in this formula are depicted in Fig. 3 and stated in (1) and (2).

ACKNOWLEDGMENT

J. R. Scott would like to thank L. Mineh for help working out the reduced CNOT measurement pattern, and for assistance in programming the C++ MBQC simulator, and O. Thomas for many interesting discussions regarding the implementation of photonic quantum computing. The authors would like to

thank J. Nunez-Yanez for very helpful discussions regarding FPGA design.

REFERENCES

[1] K. Bharti *et al.*, "Noisy intermediate-scale quantum (NISQ) algorithms," *Rev. Mod. Phys.*, vol. 94, 2021, Art. no. 015004, doi: 10.1103/RevModPhys.94.015004.

[2] A. Montanaro, "Quantum algorithms: An overview," *NPJ Quantum Inf.*, vol. 2, no. 1, Jan. 2016, Art. no. 15023, doi: 10.1038/npjqi.2015.23.

[3] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Comput. Sci. Rev.*, vol. 31, pp. 51–71, Feb. 2019, doi: 10.1016/j.cosrev.2018.11.002.

[4] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[5] A. Osman *et al.*, "Simplified Josephson-junction fabrication process for reproducibly high-performance superconducting qubits," *Appl. Phys. Lett.*, vol. 118, no. 6, 2021, Art. no. 064002, doi: 10.1063/5.0037093.

[6] F. Lecocq, F. Quinlan, K. Cicak, J. Aumentado, S. A. Diddams, and J. D. Teufel, "Control and readout of a superconducting qubit using a photonic link," *Nature*, vol. 591, no. 7851, pp. 575–579, Mar. 2021, doi: 10.1038/s41586-021-03268-x.

[7] P. E. Ross, "Why CPU frequency stalled," *IEEE Spectr.*, vol. 45, no. 4, pp. 72–72, Apr. 2008, doi: 10.1109/MSPEC.2008.4476447.

[8] T. N. Theis and H.-S. P. Wong, "The end of Moores law: A new beginning for information technology," *Comput. Sci. Eng.*, vol. 19, no. 2, pp. 41–50, Mar. 2017, doi: 10.1109/MCSE.2017.29.

[9] D. E. Browne and T. Rudolph, "Resource-efficient linear optical quantum computation," *Phys. Rev. Lett.*, vol. 95, no. 1, Jun. 2005, Art. no. 010501, doi: 10.1103/PhysRevLett.95.010501.

[10] M. Gimeno-Segovia, P. Shadbolt, D. E. Browne, and T. Rudolph, "From three-photon Greenberger-Horne-Zeilinger states to ballistic universal quantum computation," *Phys. Rev. Lett.*, vol. 115, no. 2, Jul. 2015, Art. no. 020502, doi: 10.1103/PhysRevLett.115.020502.

[11] T. Rudolph, "Why I am optimistic about the silicon-photonic route to quantum computing," *APL Photon.*, vol. 2, no. 3, Mar. 2017, Art. no. 030901, doi: 10.1063/1.4976737.

[12] S. Bartolucci *et al.*, "Fusion-based quantum computation," 2021, *arXiv:2101.09310*, doi: 10.48550/arXiv.2101.09310.

[13] J. E. Bourassa *et al.*, "Blueprint for a scalable photonic fault-tolerant quantum computer," *Quantum*, vol. 5, p. 392, Feb. 2021, doi: 10.22331/q-2021-02-04-392.

[14] S. Morley-Short, M. Gimeno-Segovia, T. Rudolph, and H. Cable, "Loss-tolerant teleportation on large stabilizer states," *Quantum Sci. Technol.*, vol. 4, no. 2, 2019, Art. no. 025014, doi: 10.1088/2058-9565/aaf6c4.

[15] H. Bombin *et al.*, "Interleaving: Modular architectures for fault-tolerant photonic quantum computing," 2021, *arXiv:2103.08612*, doi: 10.48550/arXiv.2103.08612.

[16] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *Nature*, vol. 409, no. 6816, pp. 46–52, Jan. 2001, doi: 10.1038/35051009.

[17] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Phys. Rev. A*, vol. 68, no. 2, Aug. 2003, Art. no. 022312, doi: 10.1103/PhysRevA.68.022312.

[18] D. E. Browne and H. J. Briegel, "One-way quantum computation—A tutorial introduction," 2006, *arXiv:0603226*, doi: 10.48550/arXiv.quant-ph/0603226.

[19] M. Nielsen, *Quantum Computation and Quantum Information*, 10th ed. Cambridge, NY, USA: Cambridge Univ. Press, 2010.

[20] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineers guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021318, doi: 10.1063/1.5089550.

[21] L. Chrostowski, *Silicon Photonics Design*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[22] Y. Tamura *et al.*, "The first 0.14-dB/km loss optical fiber and its impact on submarine transmission," *J. Lightw. Technol.*, vol. 36, no. 1, pp. 44–49, Jan. 2018, doi: 10.1109/JLT.2018.2796647.

[23] C. M. Natarajan, M. G. Tanner, and R. H. Hadfield, "Superconducting nanowire single-photon detectors: Physics and applications," *Supercond. Sci. Technol.*, vol. 25, no. 6, Apr. 2012, Art. no. 063001, doi: 10.1088/0953-2048/25/6/063001.

[24] C. Cahall, D. J. Gauthier, and J. Kim, "Scalable cryogenic readout circuit for a superconducting nanowire single-photon detector system," *Rev. Sci. Instrum.*, vol. 89, no. 6, Jun. 2018, Art. no. 063117, doi: 10.1063/1.5018179.

[25] E. B. Eichelberger, "Hazard detection in combinational and sequential switching circuits," *IBM J. Res. Dev.*, vol. 9, no. 2, pp. 90–99, Mar. 1965, doi: 10.1147/rd.92.0090.

[26] "7 Series FPGAs Clocking Resources User Guide," Jul. 30, 2018, UG472 (v1.14).

[27] "Distributed Memory Generator v8.0 Product Guide," Nov. 18, 2015, PG063.

[28] "7 Series FPGAs Data Sheet: Overview," Sep. 8, 2020, DS180 (v2.6.1).

[29] "7 Series FPGAs SelectIO Resources User Guide," May 8, 2018, UG471 (v1.10).

[30] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007, doi: 10.1109/TCAD.2006.884574.

[31] P. Heydari and R. Mohavavelu, "Design of ultra high-speed CMOS CML buffers and latches," in *Proc. Int. Symp. Circuits Syst.*, 2003, doi: 10.1109/iscas.2003.1205938.

[32] A. Athavale, *High-Speed Serial I/O Made Simple: A Designers' Guide, With FPGA Applications*, 1st ed. San Jose, CA, USA: Xilinx, 2005.

[33] S. H. Hall, G. W. Hall, and J. A. McCall, *High Speed Digital System Design: A Handbook of Interconnect Theory and Design Practices*. New York, NY, USA: Wiley, 2000.

[34] I. Holzman and Y. Ivry, "Superconducting nanowires for single-photon detection: Progress, challenges, and opportunities," *Adv. Quantum Technol.*, vol. 2, no. 3/4, 2019, Art. no. 1800058, doi: 10.1002/qute.201800058.

[35] L. Ma, O. Slattery, and X. Tang, "Optical quantum memory and its applications in quantum communication systems," *J. Res. Nat. Inst. Standards Technol.*, vol. 125, Jan. 2020, Art. no. 125002, doi: 10.6028/jres.125.002.

[36] J. Witzens, "High-speed silicon photonics modulators," *Proc. IEEE*, vol. 106, no. 12, pp. 2158–2182, Dec. 2018, doi: 10.1109/JPROC.2018.2877636.

[37] S. Morley-Short, S. Bartolucci, M. Gimeno-Segovia, P. Shadbolt, H. Cable, and T. Rudolph, "Physical-depth architectural requirements for generating universal photonic cluster states," *Quantum Sci. Technol.*, vol. 3, no. 1, 2017, Art. no. 015005, doi: 10.1088/2058-9565/aa913b.

[38] D. Herr, A. Paler, S. J. Devitt, and F. Nori, "A local and scalable lattice renormalization method for ballistic quantum computation," *NPJ Quantum Inf.*, vol. 4, no. 1, pp. 1–8, 2018, doi: 10.1038/s41534-018-0076-0.

[39] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, 2021, doi: 10.22331/q-2021-12-02-595.