IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 August 2024, accepted 30 September 2024, date of publication 4 October 2024, date of current version 16 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3474014

RESEARCH ARTICLE

Full Software Control on MZI-Based Photonic Integrated Circuit

YONG KWON[1], MARTINO BERNARD[2], LEONARDO LIMONGI[2,3], GIOELE PICCOLI[2], MHER GHULINYAN[2], AND BYUNG-SOO CHOI[4]

[1]Basic Science Research Institute, Pukyong National University, Busan 48513, South Korea
[2]Centre for Sensors and Devices, Fondazione Bruno Kessler, 38100 Trento, Italy
[3]Department of Industrial Engineering, University of Trento, 38122 Trento, Italy
[4]Department of Scientific Computing, Pukyong National University, Busan 48513, South Korea

Corresponding author: Byung-Soo Choi (bschoi@pknu.ac.kr)

This work was supported in part by European Union's Horizon 2020 Research and Innovation Program under Grant 899368–EPIQUS. The work of Yong Kwon and Byung-Soo Choi was supported in part by the National Research Foundation of Korea (NRF) grant funded by Korean Government Ministry of Science and ICT (MSIT) under Grant 2020K1A3A1A78087782, and in part by the Pukyong National University Industry–University Cooperation Research Fund in 2023 under Grant 202312370001.

ABSTRACT A software platform for quantum computing is required to run a complex algorithm on a real quantum device through automatically converting a quantum circuit level algorithm to a control sequence of target device. Moreover, the software platform must be tailored for the target quantum system by exploiting the properties of qubits and gates. In this work, we show how to implement a software platform for a photonic integrated circuit (PIC) platform with path-encoded photon qubits and quantum gates realized by means of interference in a mesh of tuneable Mach-Zehnder interferometers (MZIs). For universal and programmable quantum computing, the software platform consists of three standard modules: decomposer, mapper, and controller. To verify the software we runs example circuit through software and hardware integration platform and confirm each module generates correct output. Also the validation is done by comparing two results for conventionally used manual way and the software supported automatic way. These two points ensure that the implemented software works correctly when applied to the PIC.

INDEX TERMS Quantum computing, software platform, Mach-Zehnder interferometer.

I. INTRODUCTION

A photon-based quantum computer can be implemented as a quantum computing (QC) platform in a number of ways [1], [2], which satisfies the well-known condition, the DiVicenzo's criteria, to become a quantum-gate based computer [3]. The first, realized by free-space optics, is made with polarization entangled photons where quantum states are defined in a vertical state $|V\rangle$ and a horizontal state $|H\rangle$ [4]. Second, path-encoded entanglement is established as a convenient method to integrate photon-entanglement in chip-based experiments. Mach-Zehnder interferometer (MZI) based linear optical quantum computing (LOQC) is realized by the second method, which defines a quantum state that distinguishes two separate paths, called ''path-encoded'' qubit [5], [6], [7]. Quantum gates in the system can be realized by their fundamental unit, MZI by adjusting values in phase shifters figuring as a tuneable parameter [7], [8], [9], [10], [11], [12].

On the other hand, in the perspective of controlling quantum hardware working as a quantum computer, a software platform with layers which has an own functionality is needed to control a quantum hardware automatically [13], [14], [15], [16], [17]. QC process can be divided in three steps: (i) compiling or decomposing quantum gates from user-written code and representing it as a simplified intermediate encoding that is close to a machine-readable, (ii) mapping logical qubits and gates from the translated code into an actual quantum hardware, and (iii) converting the mapped code to the hardware-compatible code that supports

The associate editor coordinating the review of this manuscript and approving it for publication was Tony Thomas.

the quantum hardware. When this QC process is applied to quantum hardware, stage (i) requires only a description at the quantum circuit level, expressed simply in a unified format for qubits, quantum gates, and measurement [2], [18]. In other words, the quantum circuit representation is used for a user code compilation without any dependency in physical system of the hardware. However, in stage (ii), due to the difference in the physical implementation of QC systems, a customized algorithm for controlling target hardware is required. In the case of LOQC, a qubit and its quantum state is represented by the addition of the light-path, which determines local phase-shifts. A quantum gate is realized by a MZI which acts on both the relative and the global phase of the qubit. Experiments of programming or software control exist on large-sacle photonic processors following this approach [7], [10], [11], [19], [20], [21], [22], [23]. However, even though these studies suggest the discovery of various characteristics by implementing hardware configuration methods or optical processors, they did not show an overall process of LOQC from software to hardware and a blueprint of software configuration for LOQC.

Studies on software framework for photonic platform have also been proposed, such as continuous variable photonic quantum information processing [24], efficient compilation through graph states [25], [26], quantum simulation and programming for photonic QC [27], and MZI-based LOQC [17], [28] following the Knill–Laflamme–Milburn scheme [8]. Among the proposed software frameworks, the reference [17] proposed a design of software framework targeting MZI hardware, which takes into account the reconfigurability of a photonic quantum processor to perform various QCs, comprising in two beam splitters (BSs) and four phase shifters (PSs). Unlike a direct, customized control of the device, the framework automatically translates from a quantum circuit into a compatible form of their physical implementation, and prepares the appropriate information for the quantum device hardware encoder. The actual quantum hardware MZI is operated by adjusting phase values, the software therefore finds compatible settings from quantum gates to phase values for each MZI. Also, the framework finds proper positions considering a topology of photonic processor (e.g. Reck or Clements scheme [5], [29]) to apply quantum gates of one quantum circuit. Nevertheless, this reference only presented the design of software framework for MZI-based LOQC, not showing an actual execution of hardware by the framework or guaranteeing the verification and validation whether the framework is applicable to real hardware, even if they proposed the framework with related references.

In the present work, the implementation of software platform and the integration with real MZI-based quantum hardware will be presented. Then, we will evaluate the platform where the blueprint of proposed framework works well in the viewpoint of software verification and validation [30], [31], [32] through operating with a integrated photonic device and reporting actual experiment results. The main text of this paper is divided as follows. Overall, since this paper mainly treats the integration and execution process between software framework and optical hardware, we holistically describe the whole structure of the software platform and the hardware platform in Section II. Both platforms briefly deal with the concept, role, and relation of modules. With this prepared environment, Section III demonstrates in detail how the modules of the software platform are implemented. We then present QC experiment results for verification and validation of the software platform in Section IV. In this section, we report and analyze all data running from the software, and we stress that the software platform satisfies the fulfillment of the intended purpose. Lastly, we conclude the integration work and propose future work in Section V.

II. FRAMEWORK

A. SOFTWARE PLATFORM

In the previous work, we have presented the scheme of software platform and optimal components tailored to the characteristics of integrated photonic processors for ensuring precise execution [17]. The reference suggests that the platform should contain three modules: a decomposer, a mapper, and a controller. Users can operate the software through a user-interface, and can execute each module step by step through the interface by accessing the input/output (I/O) of each step. The schematic diagram of the software platform about connection between modules is illustrated in Fig. 1.

The three modules are essentially required to operate as a quantum computer. Initially, users can write a general quantum algorithm and send it into the quantum computer, and the decomposer analyzes the user-written code, which especially investigates the number of used qubits, quantum gates, and bits to store information, and the order of circuit operation. The module then presents the user-written algorithm in a simplified format, such as quantum assembly language (QASM), as a generic quantum circuit representation. Next, the mapper uses the output data of the decomposer to combine it into a hardware structure according to the physical properties of the target quantum computer, and assigns quantum gates to proper physical locations in the device. The module redesigns the data from a circuit representation to a hardware-specific data format to describe it in a more targeted hardware-like representation. Lastly, the controller determines a final data form for sending a user program to quantum devices, which fits to an interface of their hardware configuration. Since the quantum algorithms written by users are not directly executable, the data must be reformated into a suitable form of the actual connected hardware.

By following this framework, a generic QC process from a software platform is established. In addition, if the physical characteristics of a quantum hardware are properly reflected in the software modules, the software platform is fully constructed. Our targeted hardware is MZI-based LOQC, we therefore describe information on LOQC hardware in more detail in the following section. Detailed I/O and execution results of the platform for LOQC are covered in Section III.

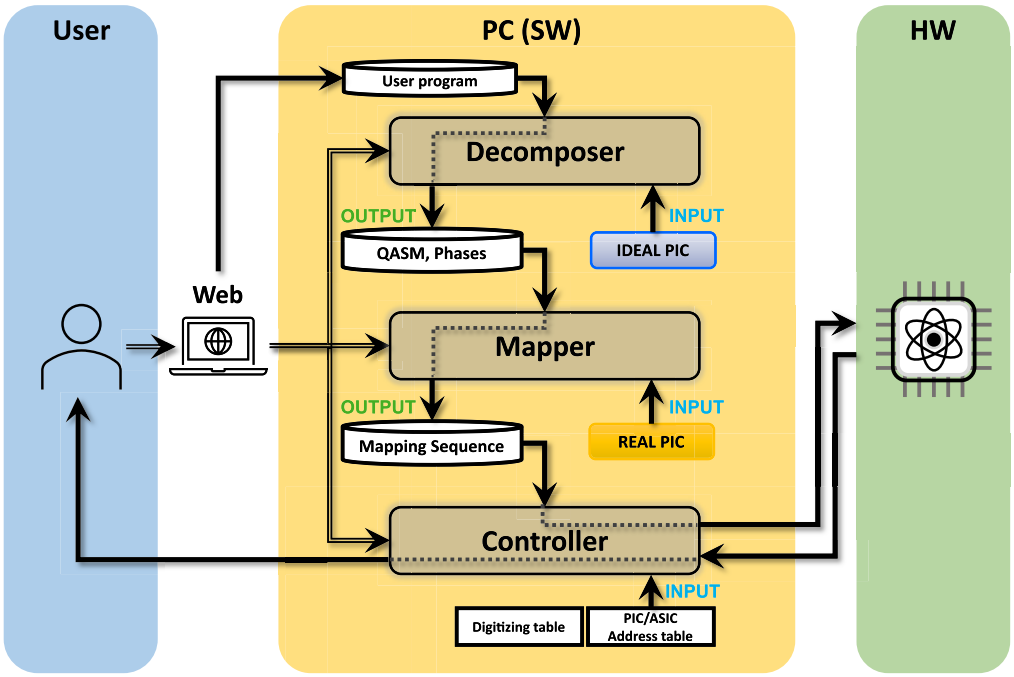

FIGURE 1. Schematic flow of LOQC software platform. Basically, the software includes three modules: decomposer, mapper, and controller. They receive input and send output to perform the role appropriate for each purpose. Users proceed with the QC process and manipulate each module to check the results of the process.

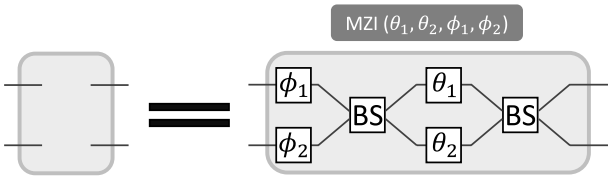

FIGURE 2. A Mach-Zehnder interferometer unit. The unit includes two beam splitters (white square box labeled BS) and four phase shifters (white boxes labeled in θ or ϕ).

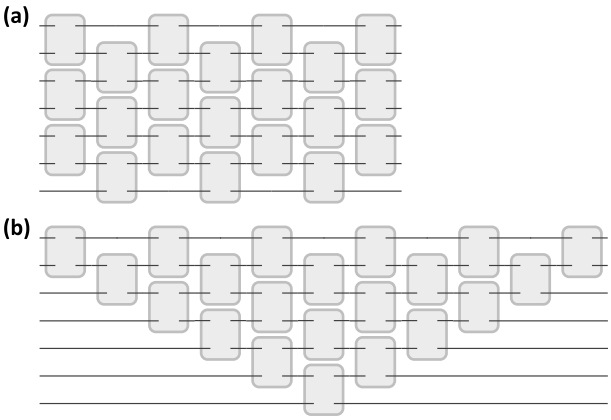

FIGURE 3. (a) 7 × 7 Clements scheme (b) 7 × 7 Reck scheme.

B. HARDWARE PLATFORM

Generally, the QC process in the MZI-based linear optical platform involves three processes. Firstly, a light source generates a physical quantum state representing a computational basis $|0\rangle$ and $|1\rangle$. The light source represents the quantum information carrier, which is selected by coherent light with continuous characteristics or single-photon source with discrete characteristics [4], [33]. Secondly, various quantum gates are implemented by MZIs and operated by propagating light through the MZIs. The light injected into an input waveguide propagates within the MZI network, and quantum gates are performed by manipulating the phase of the light. Thirdly, measurements are performed by external or internally integrated (single-photon) photodetectors (PDs).

The physical implementation of a MZI-based LOQC is achieved by fabricating a photonic integrated circuit (PIC) [34], that consists of a mesh of MZIs [1], [9], [10], [12]. Several different implementations of the MZI mesh are possible, which are popularly preferred (i) triangular shaped by Reck et al. [5] and (ii) densely packed by Clements et al. [29] (described in Fig 3), or recently suggested structure called "Bokun" mesh [35], but all of them the fundamental building block that operates as a quantum gate is a MZI. In fact, MZI meshes have already been demonstrated as linear optical processors with coherent light states [4], [33] and then also with quantum states of light [9], [36].

On the other hand, in a MZI, where various quanutm gates are implemented, PSs are only the tuneable parameter in a PIC. MZI phase shifting in a real-world environment can be implemented through controlling by means of thermo-optic modulators. The thermo-optic modulators are integrated inside the PIC as resistors. By controlling the tension at the ends of the resistor a current flows generating heat via Joule effect in the region of the waveguide. The rise in temperature induces a local change in the refractive index of

the waveguide, which ultimately results in a phase delay of the propagating light.

In our case, the PIC contains the following features. First, we adopted the Reck configuration to realize our demonstrator. Second, the specific configuration of a MZI is adaptable to the experimental context, with the present investigation employing a setup which consists of two BSs and four PSs as shown in Fig. 2. For this configuration, the full transfer matrix of a MZI can be expressed as:

$$\begin{aligned} &U_{\mathrm{MZI}}(\theta,\phi) \\ &\equiv \mathrm{BS}\cdot\mathrm{PS}(\theta)\cdot\mathrm{BS}\cdot\mathrm{PS}(\phi) \\ &= \mathrm{i}\,\mathrm{e}^{\mathrm{i}(\theta_1+\theta_2+2\phi_2)/2} \\ &\quad\cdot\begin{pmatrix} \mathrm{e}^{\mathrm{i}(\phi_1-\phi_2)}\sin((\theta_1-\theta_2)/2) & \cos((\theta_1-\theta_2)/2) \\ \mathrm{e}^{\mathrm{i}(\phi_1-\phi_2)}\cos((\theta_1-\theta_2)/2) & -\sin((\theta_1-\theta_2)/2) \end{pmatrix}, \end{aligned} \quad (1)$$

where $\theta = (\theta_1, \theta_2)$ and $\phi = (\phi_1, \phi_2)$. By adjusting the tuneable parameters θ and ϕ in this matrix, various basic quantum gates can be generated [12]. For example, $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi, 0, \pi, 0)$ for I gate, $(\theta_1, \theta_2, \phi_1, \phi_2) = (0, 0, 0, 0)$ for X gate, and $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi/2, 0, 0, 0)$ for H gate. Finally, measurements are performed with (single-photon) PDs which are integrated into the PIC substrate at the desired locations [33], [37].

The fabrication process of the photonic chips is schematically described in Fig. 4. It starts with the definition of the PD and single-photon avalanche detectors (SPADs) using four consecutive ion implantation steps, each requiring its own lithographic mask (more details in [37]). Once the photon detectors are defined within the epitaxial Si layer of the substrates, the fabrication proceeds with the realization of the photonic level. As a first step, a 1.5 μm thick silicon dioxide (SiO_2) film is deposited on top of the wafers, which acts as the bottom cladding which will isolate the Q-PIC from the Si substrate. Next, at the positions of the SPADs and selected PDs, the SiO_2 cladding is 3D shaped in a shallow-wedge form using a wet-chemical etching step. The wedge-shaped cladding will allow the output waveguides of the Q-PIC (to be defined in the next step) to "land" softly on top of the bare detector devices, thus, realizing the photonic-electronic coupling as in [38]. After the wedge definition, therefore, the process proceeds with the deposition of the dielectric film (low pressure chemical vapor deposition (LPCVD) SiN) and the definition of the Q-PIC layer by means of lithography and dry etching of the photonic circuitry. In the following steps, the Q-PIC is covered again with a 1300 nm thick SiO_2 film (top cladding), followed by a spattering of a Ti / titanium nitride (TiN) / Ti multilayer film of 190 nm with a sheet resistance of 5 Ω/sq. The TiN film will serve as the material for the PSs. Next, contact-holes for the Detector devices are opened via a lithographic step (VIAS) and dry etching through the TiN multilayer and the SiO_2 top and bottom claddings down to the epitaxial Si film. At this stage, a 1200 nm Al film is spattered on top of the wafer and patterned. Note, that due to the presence of the contact-holes, the aluminum film directly contacts the electronic devices

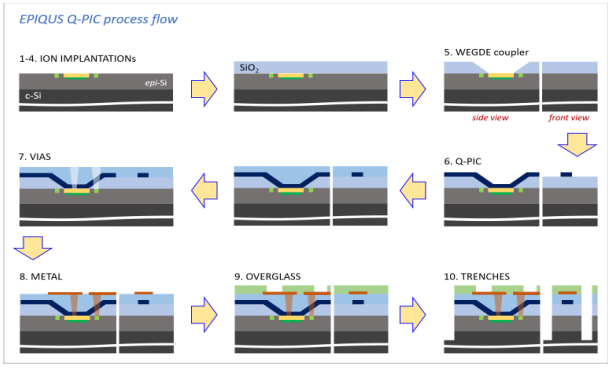

FIGURE 4. Fabrication process flow of EPIQUS' fully-integrated chips. The top-down fabrication starts with four ion implantation levels (definition of PDs and SPADs) and finishes with the definition of deep-trenches which define chip boundaries, facets of the Q-PIC waveguides and efficiency-boosting openings at the positions of MZIs.

in the Si substrate, while it accumulates on top of the Ti / TiN / Ti multilayer elsewhere. The photoresist pattern is then transferred via dry etching, defining thus also the contact pads, wide metal lines, and phase-shifting elements. The Al film is removed locally from the top of only the PSs by means of locally-enhanced wet etching of the Al film using the FBK technology. Once the metal level is defined, a 500 nm thick protective SiO_2 film is deposited on top of the chips, followed by a lithographic step and removal of the SiO_2 film only on top of the contact pads by dry etching. Finally, a last lithography defines areas through which the whole stack of SiO_2 films (∼3 μm) and a 150 μm of the Si substrate are removed by a combination of different dry etching techniques. The deep trenches allow for the formation of good waveguide facets for photonic I/O, and also for the realization of thermally insulating deep trenches around the heaters that allow us to reach a pi-shift of the MZI with about 130mW of thermal power.

Based on these characteristics, the current prepared environment is organized as depicted in Fig. 5, which shows schematically the connection diagram of the experiment, with relevant instrument pictures in the insets. Physically, the hardware takes the software platform input from the USB input of the Qontrol voltage driver, while the hardware output is fed to the software via the USB interface of the National Instruments (NI) acquisition board NI-USB-6002. The realized system features a 96-channel voltage driver (from Qontrol [39]) to control all 85 Degres of Freedom (DoF) of the Reck structure. The PIC is mounted and wire-bonded to a hosting PCB board that manages the I/O of electrical signals through standard connectors. During an experiment, the injected light travels through the PIC and the phase in each waveguide segment is controlled by a dedicated thermo-optic phase shifter. At the end of the interferometric circuit, the light is directed from the waveguide toward the substrate where the PD transforms the optical signal into a current [37]. The detection system is instead realized with custom TransImpedance Amplifiers (TIA) operating at $-3V$ bias that transforms the photocurrent into a voltage signal.

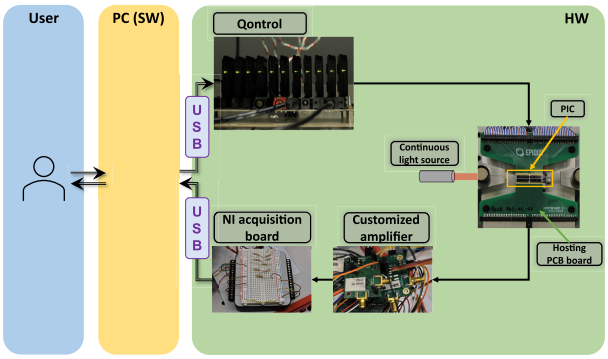

FIGURE 5. Experiment devices of MZI-based LOQC and schematic diagram. In the green box, light source, PIC, and another electronics for assisting QC devices are described and their connections are depicted with a black arrow. Devices that interface with a PC are physically connected using the USB protocol.

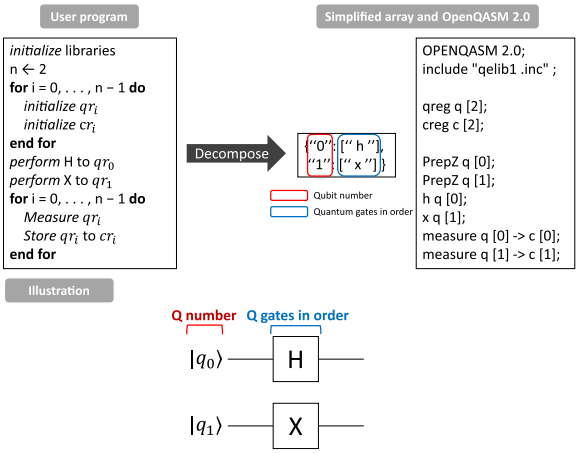

FIGURE 6. Decomposed result from user-written quantum circuit. Any quantum algorithm can be written by user, then the algorithm is converted by the decomposer. The algorithm is expressed as a simplified array and OpenQASM 2.0 simultaneously. The simplified array of the quantum circuit contains two elements, qubit number and quantum gates in order, which shows the quantum circuit depicted on illustration.

Ultimately, the voltage signal is digitalized by an NI-USB-6002 acquisition board, after that processing the data is finally transferred to the PC by USB. This detection arrangement is suited for the calibration of the MZI parameters with classical light sources. The same PDs can also be alternatively operated as single-photon detectors in geiger-mode once quantum states of light are injected as described in [37].

III. IMPLEMENTATION

This section presents the implementation of LOQC software platform with detailed I/O demonstration of necessary modules, which are demonstrated in terms of their features and optical hardware properties covered in Section II. Without loss of generality for demonstrating the software execution process and thoroughly checking an entire QC process, we have chosen to illustrate the implementation of a 1-depth circuit. The circuit describes applying the H gate to the zeroth qubit q_0 and the X gate to the first qubit q_1 as described in Algorithm 1.

Algorithm 1 An Example Code for Applying Hadamard to the Zeroth Qubit and Pauli-X to the First Qubit

```
initialize libraries
n ← 2
for i = 0, . . . , n − 1 do
    initialize qr_i
    initialize cr_i
end for
perform H to qr_0
perform X to qr_1
for i = 0, . . . , n − 1 do
    Measure qr_i
    Store qr_i to cr_i
end for
```

A. DECOMPOSER

LOQC decomposer, the starting module of the LOQC software platform, simply converts the user-written high-level code into a clear and concise low-level representation, as mentioned in Section I and Section II-A where the QC process is described. No special conditions related to physical properties need to be considered, and therefore the output from this module is demonstrated in the same quantum circuit representation as the user-written code. However, controlled-NOT (CNOT) gate should be decomposed into H, controlled-Z, and H sequences, since the direct physical realization of CNOT is unavailable based on the post-selection method in PIC and has been demonstrated with the sequence of gates in photonic experiments [40], [41]. Meanwhile, for higher QC performance on the quantum circuit level, the decomposer can contain several quantum gate decomposition method [42], [43].

In our implementation, we construct the LOQC decomposer providing "OpenQASM 2.0" and an "simplified array" as a result data. The former one is a widely used format for describing a quantum circuit in near-machine-readable text, including quantum bits, classical bits, and the number of quantum gates used [44]. In contrast, the latter one only contains a qubit number and quantum gates expressed as

$$\{qr_0 : [\mathcal{U}_0^0, \mathcal{U}_1^0, \mathcal{U}_2^0, \cdots], qr_1 : [\mathcal{U}_0^1, \mathcal{U}_1^1, \mathcal{U}_2^1, \cdots], \cdots, qr_{n-1} : [\mathcal{U}_0^{n-1}, \mathcal{U}_1^{n-1}, \mathcal{U}_2^{n-1}, \cdots]\} \quad (2)$$

for n qubits where qr_n is a n-th qubit and $\mathcal{U}_m^n$ is a m-th unitary quantum gate assigned in n-th qubit. Compared with the OpenQASM format, while the former format illustrates all quantum gates applied to each qubit, the simplified array first sorts by qubit number and then lists the gates applied to the qubits in order. However, the array format has disadvantages that are not user-friendly and merely describe a simple listing. Instead, the array is useful for sending information to the next software module, LOQC mapper, since the two elements in the array are only required. A more detailed mapping process with the array is presented in Section III-B.

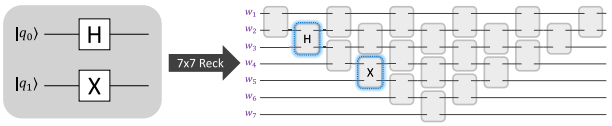

FIGURE 7. Illustration of the mapping process. The main role of the mapper reflects a quantum circuit level data generated by the decomposer to a PIC scheme. Here, the circuit is mapped to the 7 × 7 Reck structure, which describes that $|q_0\rangle$ is defined with three waveguides w_1, w_2, w_3 equal to $|A\rangle, |0\rangle, |1\rangle$, where $|A\rangle$ is for an auxiliary purpose. Another waveguides w_4, w_5, and w_6 defines $|0\rangle, |1\rangle$, and $|A\rangle$ for the qubit $|q_1\rangle$. The other waveguide is not used. According to the definition of quantum states in Section II-B, H gate on q_0 and Z-gate on q_1 is mapped to the location which is explicitly indicated in the blue contour.

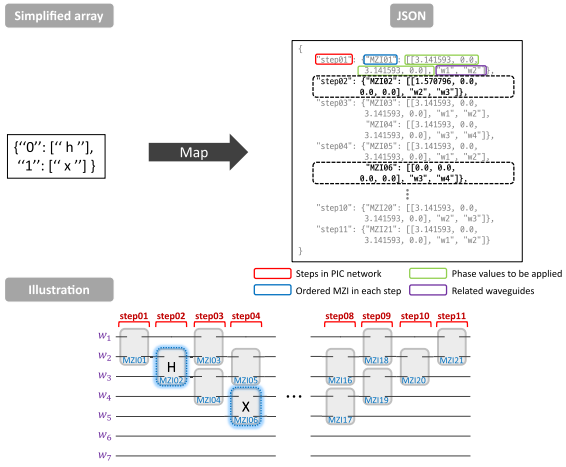

FIGURE 8. Mapped result from decomposed data. When simplified array generated by decomposition goes through the mapper, the data results in the classified string regarding a MZI network. To describe the mapped gates in a string data clearly, the mapped parts are expressed in black with dashed contour and the others are colored in grey. Based on the string data, the schematic diagram is also represented in illustration.

We demonstrate the result of the example circuit from the decomposer with an illustration in Fig. 6. When the user program is written as Algorithm 1, the decomposer swipes all codes from the beginning to the end, thereby determining the number of qubits used and the quantum gates assigned to a logical qubit. For example, the decomposer finds the circuit part in the user program that is Hadamard on q_0 and Pauli-X on q_1. The module then generates the simplified array, {"0":["h"], "1":["x"]}, in accordance with (2) and the OpenQASM 2.0 format, as illustrated in Fig. 6.

B. MAPPER

LOQC mapper performs the assignment of logical qubits and quantum gates from a decomposer result to a real environment of physical hardware. The operation mechanism of LOQC hardware, as mentioned in Section II-B, is that quantum states of a qubit are interpreted as two waveguides according to the dual-rail encoding, and quantum gates are realized by MZIs which are packed with a pair of waveguides. Considering these properties, as schematically illustrated in Fig. 7, the primary function of the mapper is to transform the decomposer data into a generic quantum circuit representation, which is then mapped to a photonic circuit. Thus, the mapper identifies a potential location for a MZI to implement a quantum gate, referencing the structure of a MZI mesh, the Reck and Clements scheme [5], [29] in Fig. 3 as a representative.

We demonstrate the implementation of the LOQC mapper as follows. Concerning MZI schemes described in Fig. 3 and the mapping process in Fig 7, a mapped data set is classified into sectors to represent an overall figure of a PIC. The simplified array resulting from the decomposer is re-categorized in JSON format, which is divided into "steps" and "MZIs". The steps are divided in the column of the PIC, and the numbered MZIs indicate the vertical direction of MZIs sequentially counted from top to bottom. Subsequently, four elements of a single array in each MZI include phase values $\{\phi_1, \phi_2, \theta_1, \theta_2\}$ of a quantum gate to be applied regarding the U_{MZI} in (1). Finally, the final two elements of the array are designated for waveguide information, which allows to distinguish which waveguide the MZI is associated with. By putting these conditions all together, the format is generalized and expressed as

$$\begin{aligned}
\{\ \text{step}_{01}: &\ \{\text{MZI}_{01}: [[\theta_{1,01}, \theta_{2,01}, \phi_{1,01}, \phi_{2,01}], \\
&\quad \text{“}w_1''\text{, “}w_2''\text{]}\} \\
\text{step}_{02}: &\ \{\text{MZI}_{02}: [[\theta_{1,02}, \theta_{2,02}, \phi_{1,02}, \phi_{2,02}], \\
&\quad \text{“}w_2''\text{, “}w_3''\text{]}\} \\
&\ \vdots \\
\text{step}_{j-1}: &\ \{\text{MZI}_{i-1}: [[\theta_{1,i-1}, \theta_{2,i-1}, \phi_{1,i-1}, \phi_{2,i-1}], \\
&\quad \text{“}w_2''\text{, “}w_3''\text{]}\} \\
\text{step}_{j}: &\ \{\text{MZI}_{i}, : [[\theta_{1,i}, \theta_{2,i}, \phi_{1,i}, \phi_{2,i}], \\
&\quad \text{“}w_{m-1}''\text{, “}w_m''\text{]}\}\}
\end{aligned} \quad , \tag{3}$$

where the number of j steps, the number of i MZIs, and m-th waveguide which has the range $[1, 2, \ldots, m, \ldots, n]$ for $n \times n$ scheme.

For the circuit example, Hadamard on q_0 and Pauli-X on q_1, the mapped result can be described as Fig. 8. When the mapper loads the simplified array for the circuit, the module identifies the optimal location for a quantum gate. The selected structure of the PIC is 7×7 Reck scheme, which allows for the placement of quantum gates within MZIs, which are comprised of two pairs of waveguides, (w_2, w_3) and (w_4, w_5). The MZI that is closest to the starting point is designated as "MZI02" for the Hadamard gate and "MZI06" for the Pauli-X. To summarize, the data for this circuit is written in

$$\begin{aligned}
\{\ &\text{step01}: \{\text{MZI01}: [[\pi, 0, \pi, 0], \text{“}w_1''\text{, “}w_2''\text{]}\} \\
&\text{step02}: \{\text{MZI02}: [[\pi/2, 0, 0, 0], \text{“}w_2''\text{, “}w_3''\text{]}\} \\
&\qquad\qquad \vdots \\
&\text{step04}: \{\text{MZI05}: [[\pi, 0, \pi, 0], \text{“}w_1''\text{, “}w_2''\text{]} \\
&\qquad \text{MZI06}: [[0, 0, 0, 0], \text{“}w_3''\text{, “}w_4''\text{]}\} \\
&\qquad\qquad \vdots \\
&\text{step11}: \{\text{MZI21}: [[\pi, 0, \pi, 0], \text{“}w1''\text{, “}w2''\text{]}\}\ \}
\end{aligned} \quad . \tag{4}$$

Note that phase values for I, H, and X are calculated by (1).

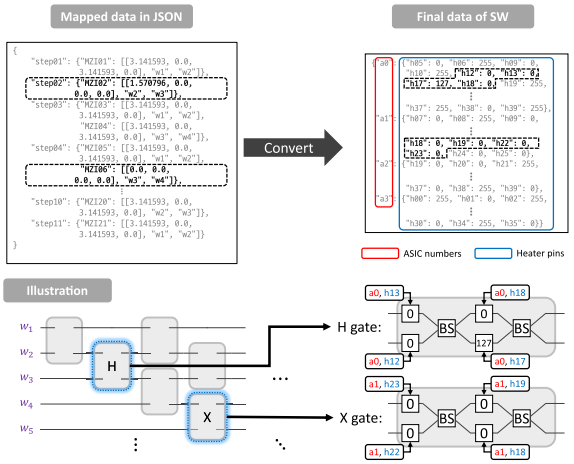

FIGURE 9. Converted hardware-compatible data by the controller. Since the data generated by the mapper is based on the MZI location in a PIC, the controller converts it to match the input interface of the auxiliary hardware that connects to the actual processor. Each MZI has connection addresses associated with 4 devices with 40 communication channels. In this example, the address of H or X gate mapped location of MZI is depicted.

C. CONTROLLER

LOQC controller performs more tasks than other software modules, which are closely correlated with hardware. In implementation, two functions are assigned to the module: (i) the generation of the final software output, and (ii) the transmission of the software output and the receipt of the result from the hardware platform. The following sections provide more detailed explanations of the roles.

1) CONVERTING MAPPING DATA SUITABLE FOR QUANTUM DEVICES

LOQC controller translates the mapping sequence generated by the mapper into a vector including control values. As previously discussed in Section II-B, the PIC is embedded with a customized daughterboard. The address for sending a physical signal to PIC is divided into four sectors, with up to 40 channels per sector. This allows the assisting device to receive values from a total of 160 channels per trial.

Accordingly, the LOQC controller processes mapped data in accordance with the electronic devices that support the PIC. The mapper data, which represents the MZI mesh, is translated into the 160 channels supported by the device through the MZI connection address table. Furthermore, since the equipment can receive the voltage value in integer, which will act on the phase shifter of the PIC, it is not possible to express all phase values represented in real number. Consequently, the phase value in the mapper data must be digitized. The value has a range of 0 to $2^9 - 1$, which is equivalent to a range of phase values by the regularization $phase \times (2^9 - 1)/2\pi$. Following the completion of the translation process, the final data is determined in the form of a single vector, which is then ready for the voltage injection command by the assisting electronic device for the PIC.

The result of the LOQC controller for the example circuit is described in Fig. 9 and implemented as follows. According to the mapped array, the quantum gate is mapped in "MZI02" and "MZI05", while identity gates are applied to all other parts with the phase $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi, 0, \pi, 0)$. The addresses of phase shifters that should be connected are "a0, h13", "a0, h12", "a0, h18", and "a0, h17" for H gate and "a1, h23", "a1, h22", "a1, h19", and "a1, h18" for X gate, as displayed in the illustration part of Fig. 9. The normalization of a phase allows the phase values of the two gates to be converted into the following format: (a0 h18, a0 h17, a0 h13, a0 h12) = (0, 127, 0, 0) for H gate, and (a1 h19, a1 h18, a1 h23, a1 h22) = (0, 0, 0, 0) for X gate. The phase values for the remaining unused MZIs are set in $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi, 0, \pi, 0)$ by the mapper, and then converted to (255, 0, 255, 0) on compatible locations.

2) SENDING AND RECEIVING DATA

The controller then performs sending and receiving programming data as an interface between the software platform and the hardware platform. This is the closest process between the two platforms, and it is therefore crucial to ensure that two platforms and that the input and output data is flowing back and forth correctly.

On the output side of software, the controller attempts to verify the connection with Qontrol device which is physically connected via a USB interface (described in Fig. 5). Once the connection has been confirmed, the module instructs the Qontrol to execute the vector array generated in the previous process on the quantum device. Subsequently, the hardware platform initiates a photonic experiment on the prepared hardware platform. In contrast, in the case of the input side of the software, the module waits while the experiment is conducted by the software and checks the connection with the measurement device. Once the connection is established, the module attempts to collect measurement data from the device. The data is received as a bit "0" or "1" for each logical qubit, with the detector located at the end of the PIC serving as the measurement device. Ultimately, the controller presents the measurement data to the user, thereby indicating the completion of the QC process on the MZI-based LOQC platform.

Regarding the execution for the example circuit, the second process of the LOQC controller for the circuit is not described here because this part deals with the hardware platform and is not relevant to the software implementation. Instead, the entire process of the LOQC platform including data transformation on the hardware platform is presented in the following Section IV-A.

IV. VERIFICATION AND VALIDATION

In this section, in order to prove that the implemented LOQC software in the previous section correctly works with real hardware platform described in Section II-B, we address the results of the framework in the context of software verification and validation, which includes the demonstration

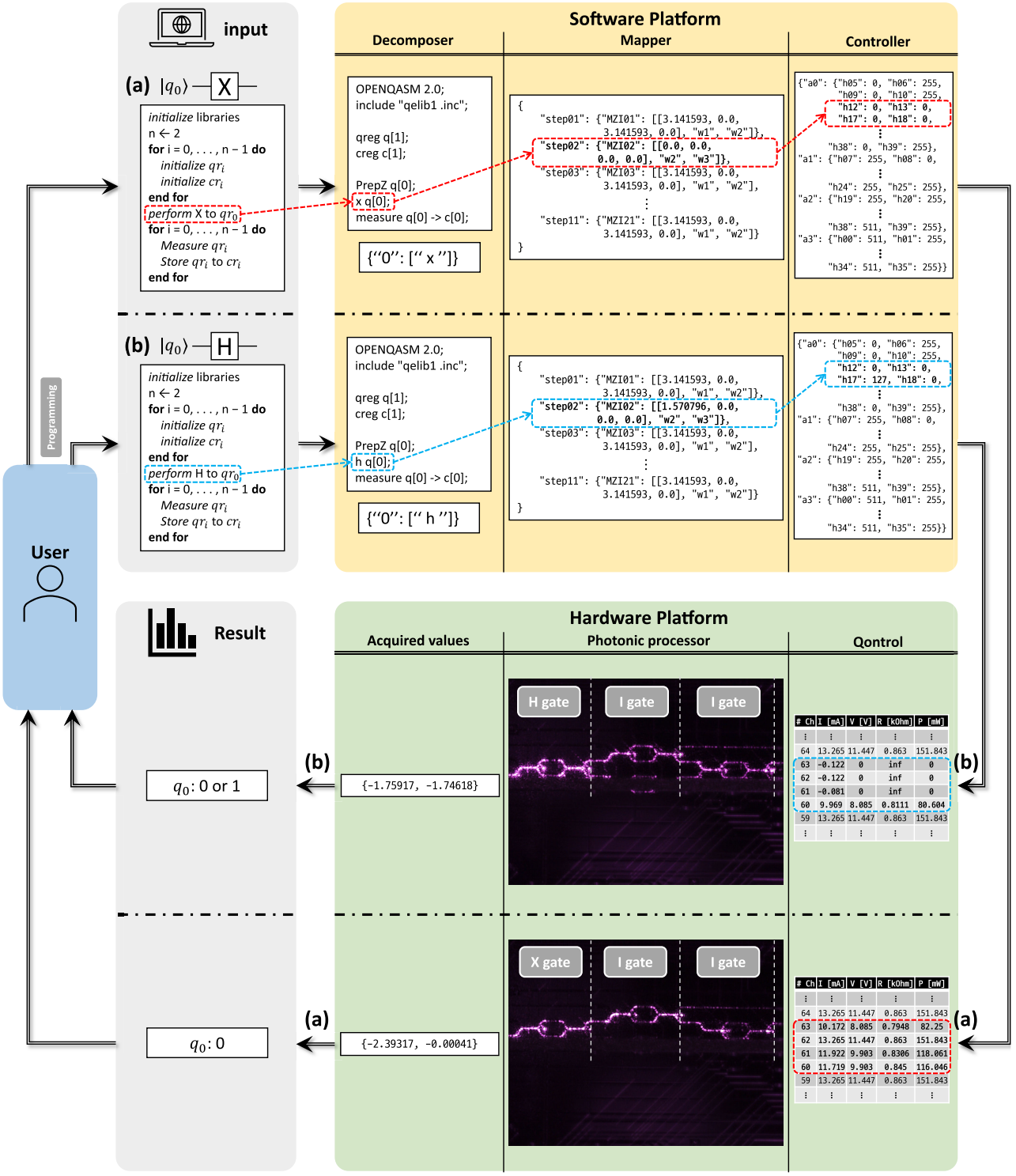

FIGURE 10. Schematic diagram of applying a one single-qubit gate one-time experiment and results. We report all results from the implemented LOQC platform with the main components as boxes and the data flow as arrows. The figure shows I/O result of each component for the case of applying (a) Pauli-X to q_0 and (b) Hadamard to q_0. Followed by the command of software platform, both of the gates are implemented on the second MZI shown in hardware platform. As progressing through each step, major changes are colored in red for (a) and blue for (b).

of software–hardware integration. The software verification process can show whether the software correctly works for all operations, and the software validation process can prove whether the software satisfies intended use [30], [31], [32]. In our case, the verification of this framework proceeds through displaying all outputs from each process, from user input to measurement result, including hardware execution. The validation process describes whether the results between

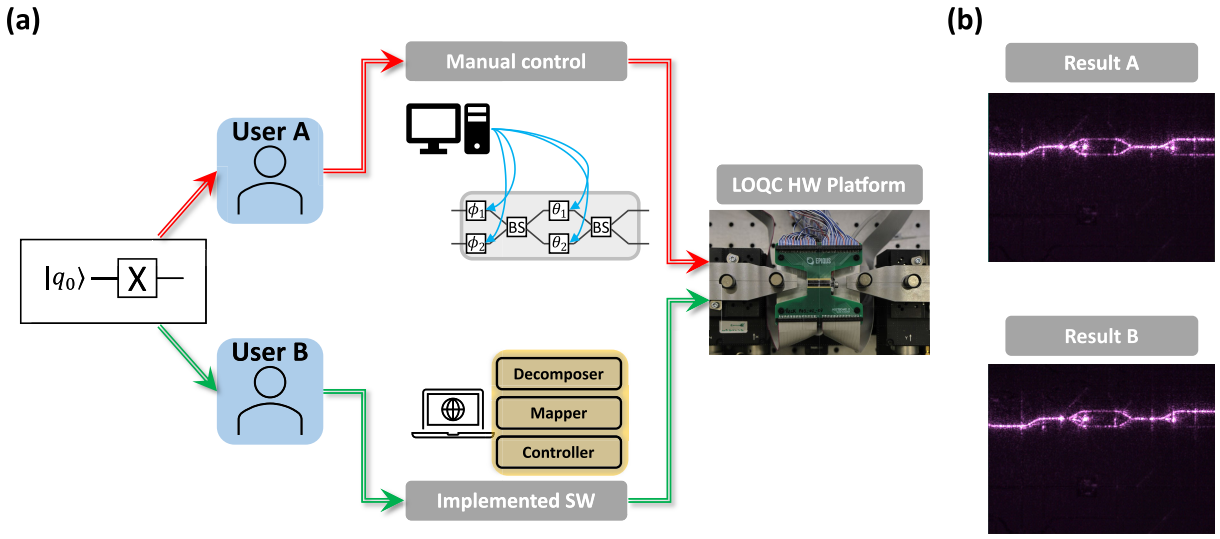

FIGURE 11. (a) Software validation process on LOQC. For validating the implemented software, we investigate the results of LOQC hardware applying Pauli-X gate on a same MZI reserved for testing by manipulating the photonic chip manually (red path) or using the implemented software platform (green path). Except for the two experimental methods, the experiment is conducted while keeping the hardware platform the same. (b) Captured photos of the light passing through the same MZI showing the results for User A and User B.

direct hardware manipulation and software application to hardware are identical. Through this process, it is ensured that the software has been implemented correctly, reflecting the properties of both quantum computing and linear optics. Note that the results obtained from software-related experiments identically follow the QC process described in Section III.

A. VERIFICATION THROUGH APPLYING A SINGLE-QUBIT GATE

We investigate all outputs of the overall process for two small-scale experiments, from implemented software platform in Section III to prepared hardware platform in Section II-B, and report them together with the schematic diagram illustrated in Fig. 10. In the first experiment, Pauli-X is applied on q_0, and in the second, Hadamard is applied to q_0. Prior to executing LOQC software, an algorithm for running a single qubit gate can be written in Algorithm 1, replacing the codes from lines 7 and 8 with "*perform* X to qr_0" if we apply Pauli-X. The LOQC software then receives the codes and the implemented modules, decomposer, mapper, and controller are executed in sequence, as explained in Section III. The distinction between the two experiments lie in the phase parameter value θ, ϕ, which is $(\theta_1, \theta_2, \phi_1, \phi_2) = (0, 0, 0, 0)$ for the X gate and $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi/2, 0, 0, 0)$ for the H gate. Next, since one quantum gate is only used in both of experiments, the mapper assigns the same location of a MZI, and after that the address information of the mapper, the controller on the software platform, and the electric power information on the hardware platform are identical. In other words, all conditions are identical, except for applying a quantum gate. In this case, it is only necessary to verify whether the hardware and measured results were observed as expected operations.

The results of the hardware platform, as depicted in Fig. 10, reveal the expected operation behaviors. Initially, referring to the prepared hardware system depicted in Fig. 5, the Qontrol electronic device sets voltage values, which are then directly injected into the target PSs in a MZI. After injecting an electric signal into the photonic hardware, the path of light penetrating the waveguides is shown, and the desired path is identified for each applied quantum gate. When the light source begins to propagate in the lower waveguide, the path is detected as inverting to the upper waveguide for Pauli-X and superposing to both waveguides for Hadamard. This result indicates that the quantum hardware is operated as intended, executing the instructions from the software platform and passing through each module. Subsequently, the measurement data from the experiments is collected in the form of voltage values, analogous to the input data to the PIC, which is performed on the photodiodes assembled in each waveguide at the end of the experiment. Consequently, by tracing and analyzing the measurement data, the result provides the user with the bit data detected in each qubit. In this experiment, the X-gate converges to 0 and the H-gate converges to 0 and 1 at the same ratio.

B. VALIDATION COMPARING WITH DIRECT MANIPULATION

In the validation process, it is necessary to investigate whether the software is identical to a user-directed manipulation. Prior to deployment of the software platform, the user-directed manipulation of a MZI is conducted by adjusting phase shifters manually. The current hardware platform permits a computer code to process the voltage applied to phase shifters via a daughterboard connected directly to the PIC (for further details, see Section II-B and Fig. 11). The custom code allows free modification of the voltage input to the PS as the same way as the verification process that the Qontrol sets the voltage values. The optical path can be altered

instantaneously in accordance with the content of the written code.

We compare the Pauli-X experiment by manual operation and by software platform for the same MZI. In order to implement the gate, the phase shifters are tuned to $(\theta_1, \theta_2, \phi_1, \phi_2) = (\pi, \pi, \pi, \pi)$, as this setting is visually represented as Pauli-X according to (1), analogous to the default state which is represented by $(\theta_1, \theta_2, \phi_1, \phi_2) = (0, 0, 0, 0)$. For manual operation, custom code which contains the Pauli-X gate implementation is applied to the hardware platform. The process using the software platform sets up the Pauli-X gate at the quantum circuit level, and goes through all software modules described in Section III. As illustrated in Fig 11, the experimental results were almost similar to what was expected for the same operation. Although some unintended light is observed at the output port of the lower waveguide in the direct manipulation experiment, the two results are almost identical. Therefore, the implemented software can be considered to operate correctly compared to direct manipulation.

V. CONCLUSION

We have presented the implementation of the software platform followed by the suggested software design for LOQC [17] and reported the results of applying them. The software components, decomposer, mapper, and controller, are demonstrated in compatible data form, which reflects the photonic properties explained in Section II-B. To ensure that the implemented software operates as a quantum computer perfectly, experiments were conducted with a prepared hardware platform and analyzed from the perspective of software verification and validation. Although the PIC can only support a small number of qubits and a limited number of gates and has the potential to exhibit incomplete behaviors, the software functions correctly by proceeding with verification in Section IV-A and validation in Section IV-B.

This paper presents three future work items for improving the software platform. Firstly, the platform applied in this work has been tested on the small-scale supported PIC, and it should be conducted on the larger-scale supported PIC. Also, the software platform could be considered to support testing special quantum algorithms such as the Quantum Approximate Optimization Algorithm or Variational Quantum Algorithms for solving quantum problems [45], [46], [47]. Secondly, to operate effective QC and to evaluate the performance of a photonic device as opposed to the method of software verification and validation, it is necessary to apply QC theory-based verification methods. For small-scale photonic devices, quantum state tomography or quantum process tomography can be included in the software and controlled by the software platform [48], [49], [50], [51]. This technique can provide the implication of the ability to analyze the performance of a PIC, by comparing a quantum fidelity between ideal and experimental cases. Thirdly, given that the software platform is composed of three modules (decomposer, mapper, and controller) with a simple algorithm, it is necessary to improve the module to apply generalized and sophisticated algorithms to support complex quantum circuits.

ACKNOWLEDGMENT

The authors would like to thank Tiziano Stedile (Fondazione Bruno Kessler) for the PCB board preparation and MNF Laboratory for supporting fabrication of the photonic chip.

REFERENCES

[1] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn, "Linear optical quantum computing with photonic qubits," *Rev. Modern Phys.*, vol. 79, no. 1, pp. 135–174, Jan. 2007.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed., Cambridge, U.K.: Cambridge Univ. Press, 2010.

[3] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte der Physik*, vol. 48, nos. 9–11, pp. 771–783, Sep. 2000.

[4] J. L. O'Brien, "Optical quantum computing," *Science*, vol. 318, no. 5856, pp. 1567–1570, Dec. 2007.

[5] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Phys. Rev. Lett.*, vol. 73, no. 1, pp. 58–61, Jul. 1994.

[6] N. J. Cerf, C. Adami, and P. G. Kwiat, "Optical simulation of quantum logic," *Phys. Rev. A, Gen. Phys.*, vol. 57, no. 3, pp. R1477–R1480, 1998.

[7] W. Bogaerts, D. Pérez, J. Capmany, D. A. B. Miller, J. Poon, D. Englund, F. Morichetti, and A. Melloni, "Programmable photonic circuits," *J. Phys., Conf. Ser.*, vol. 586, no. 7828, pp. 207–216, Oct. 2022.

[8] 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.

[9] J. Carolan, C. Harrold, C. Sparrow, E. Martín-López, N. J. Russell, J. W. Silverstone, P. J. Shadbolt, N. Matsuda, M. Oguma, M. Itoh, G. D. Marshall, M. G. Thompson, J. C. F. Matthews, T. Hashimoto, J. L. O'Brien, and A. Laing, "Universal linear optics," *Science*, vol. 349, pp. 711–716, Aug. 2015.

[10] N. C. Harris, D. Bunandar, M. Pant, G. R. Steinbrecher, J. Mower, M. Prabhu, T. Baehr-Jones, M. Hochberg, and D. Englund, "Large-scale quantum photonic circuits in silicon," *Nanophotonics*, vol. 5, no. 3, pp. 456–468, Aug. 2016.

[11] N. C. Harris, J. Carolan, D. Bunandar, M. Prabhu, M. Hochberg, T. Baehr-Jones, M. L. Fanto, A. M. Smith, C. C. Tison, P. M. Alsing, and D. Englund, "Linear programmable nanophotonic processors," *Optica*, vol. 5, no. 12, p. 1623, 2018.

[12] Y. Kwon, A. Baldazzi, L. Pavesi, and B.-S. Choi, "Quantum circuit mapping for universal and scalable computing in MZI-based integrated photonics," *Opt. Exp.*, vol. 32, no. 7, p. 12852, 2024.

[13] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov, "A layered software architecture for quantum computing design tools," *Computer*, vol. 39, no. 1, pp. 74–83, Jan. 2006.

[14] National Academies of Sciences, Engineering, and Medicine, "National academies of sciences, engineering, and medicine," in *Quantum Computing: Progress and Prospects*, E. Grumbling and M. Horowitz, Eds., Washington, DC, USA: National Academy Press, 2019.

[15] Y. Hwang, T. Kim, C. Baek, and B.-S. Choi, "Integrated analysis of performance and resources in large-scale quantum computing," *Phys. Rev. Appl.*, vol. 13, no. 5, May 2020, Art. no. 054033.

[16] L. Riesebos, B. Bondurant, J. Whitlow, J. Kim, M. Kuzyk, T. Chen, S. Phiri, Y. Wang, C. Fang, A. V. Horn, J. Kim, and K. R. Brown, "Modular software for real-time quantum control systems," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2022, pp. 545–555.

[17] Y. Kwon and B.-S. Choi, "A software platform for programmable linear optical quantum computer," *IEEE Access*, vol. 11, pp. 112682–112692, 2023.

[18] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.

[19] C. Taballione, T. A. W. Wolterink, J. Lugani, A. Eckstein, B. A. Bell, R. Grootjans, I. Visscher, D. Geskus, C. G. H. Roeloffzen, J. J. Renema, I. A. Walmsley, P. W. H. Pinkse, and K.-J. Boller, "8 × 8 reconfigurable quantum photonic processor based on silicon nitride waveguides," *Opt. Exp.*, vol. 27, no. 19, p. 26842, 2019.

[20] F. Hoch, S. Piacentini, T. Giordani, Z.-N. Tian, M. Iuliano, C. Esposito, A. Camillini, G. Carvacho, F. Ceccarelli, N. Spagnolo, A. Crespi, F. Sciarrino, and R. Osellame, "Reconfigurable continuously-coupled 3D photonic circuit for boson sampling experiments," *NPJ & Quantum Inf.*, vol. 8, no. 1, pp. 1–7, May 2022.

[21] M. Dong, M. Zimmermann, D. Heim, H. Choi, G. Clark, A. J. Leenheer, K. J. Palm, A. Witte, D. Dominguez, G. Gilbert, M. Eichenfield, and D. Englund, "Programmable photonic integrated meshes for modular generation of optical entanglement links," *NPJ & Quantum Inf.*, vol. 9, no. 1, pp. 1–8, Apr. 2023.

[22] J. Bütow, J. S. Eismann, V. Sharma, D. Brandmüller, and P. Banzer, "Generating free-space structured light with programmable integrated photonics," *Nature Photon.*, vol. 18, no. 3, pp. 243–249, Mar. 2024.

[23] Y. Yang, R. J. Chapman, A. Youssry, B. Haylock, F. Lenzini, M. Lobino, and A. Peruzzo, "Programmable quantum circuits in a large-scale photonic waveguide array," 2024, *arXiv:2405.13654*.

[24] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, "Strawberry fields: A software platform for photonic quantum computing," *Quantum*, vol. 3, p. 129, Mar. 2019.

[25] F. Zilk, K. Staudacher, T. Guggemos, K. Fürlinger, D. Kranzlmüller, and P. Walther, "A compiler for universal photonic quantum computers," in *Proc. IEEE/ACM 3rd Int. Workshop Quantum Comput. Softw. (QCS)*, Nov. 2022, pp. 57–67.

[26] H. Zhang, A. Wu, Y. Wang, G. Li, H. Shapourian, A. Shabani, and Y. Ding, "OneQ: A compilation framework for photonic one-way quantum computation," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, vol. 4, Jun. 2023, pp. 1–14.

[27] Z. Kolarovszki, T. Rybotycki, P. Rakyta, Á. Kaposi, B. Poór, S. Jóczik, D. T. R. Nagy, H. Varga, K. H. El-Safty, G. Morse, M. Oszmaniec, T. Kozsik, and Z. Zimborás, "Piquasso: A photonic quantum computer simulation software platform," 2024, *arXiv:2403.04006*.

[28] N. Heurtel, A. Fyrillas, G. D. Gliniasty, R. Le Bihan, S. Malherbe, M. Pailhas, E. Bertasi, B. Bourdoncle, P.-E. Emeriau, R. Mezher, L. Music, N. Belabas, B. Valiron, P. Senellart, S. Mansfield, and J. Senellart, "Perceval: A software platform for discrete variable photonic quantum computing," *Quantum*, vol. 7, p. 931, Feb. 2023.

[29] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walsmley, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, p. 1460, 2016.

[30] W. L. Oberkampf, T. G. Trucano, and C. Hirsch, "Verification, validation, and predictive capability in computational engineering and physics," *Appl. Mech. Rev.*, vol. 57, no. 5, pp. 345–384, Sep. 2004.

[31] W. L. Oberkampf and T. G. Trucano, "Verification and validation benchmarks," *Nucl. Eng. Design*, vol. 238, no. 3, pp. 716–743, Mar. 2008.

[32] M. Lewis, S. Soudjani, and P. Zuliani, "Formal verification of quantum programs: Theory, tools, and challenges," *ACM Trans. Quantum Comput.*, vol. 5, no. 1, pp. 1–35, Mar. 2024.

[33] L. Gemma, M. Bernard, and D. Brunelli, "An optical tool to optimize the output of a photonic integrated chip architecture," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 3, pp. 694–702, Sep. 2022.

[34] M. Sanna, A. Baldazzi, G. Piccoli, S. Azzini, M. Ghulinyan, and L. Pavesi, "SiN integrated photonic components in the visible to near-infrared spectral region," *Opt. Exp.*, vol. 32, no. 6, p. 9081, 2024.

[35] K. H. R. Mojaver, B. Zhao, E. Leung, S. M. R. Safaee, and O. Liboiron-Ladouceur, "Addressing the programming challenges of practical interferometric mesh based optical processors," *Opt. Exp.*, vol. 31, no. 15, pp. 23851–23866, 2023.

[36] J. Wang, S. Paesani, Y. Ding, R. Santagati, P. Skrzypczyk, A. Salavrakos, J. Tura, R. Augusiak, L. Mančinska, D. Bacco, D. Bonneau, J. W. Silverstone, Q. Gong, A. Acín, K. Rottwitt, L. K. Oxenløwe, J. L. O'Brien, A. Laing, and M. G. Thompson, "Multidimensional quantum entanglement with large-scale integrated optics," *Science*, vol. 360, no. 6386, pp. 285–291, Apr. 2018.

[37] F. Acerbi, M. Bernard, B. Goll, A. Gola, H. Zimmermann, G. Pucker, and M. Ghulinyan, "Monolithically integrated SiON photonic circuit and silicon single-photon detectors for NIR-range operation," *J. Lightw. Technol.*, vol. 42, no. 8, pp. 2831–2841, Apr. 15, 2024.

[38] M. Bernard, F. Acerbi, G. Paternoster, G. Piccoli, L. Gemma, D. Brunelli, A. Gola, G. Pucker, L. Pancheri, and M. Ghulinyan, "Top-down convergence of near-infrared photonics with silicon substrate-integrated electronics," *Optica*, vol. 8, no. 11, p. 1363, 2021.

[39] *We Configured the Channel With 12 Qontrol Q8b Driver Modules Mounted on the BP12 Board*. [Online]. Available: https://qontrol.co.uk/

[40] J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning, "Demonstration of an all-optical quantum controlled-NOT gate," *Nature*, vol. 426, no. 6964, pp. 264–267, Nov. 2003.

[41] J.-M. Lee, W.-J. Lee, M.-S. Kim, S. Cho, J. J. Ju, G. Navickaite, and J. Fernandez, "Controlled-NOT operation of SiN-photonic circuit using photon pairs from silicon-photonic circuit," *Opt. Commun.*, vol. 509, Apr. 2022, Art. no. 127863.

[42] T. Kim and B.-S. Choi, "Efficient decomposition methods for controlled-R_n using a single ancillary qubit," *Sci. Rep.*, vol. 8, no. 1, p. 5445, Apr. 2018.

[43] P. Rakyta and Z. Zimborás, "Efficient quantum gate decomposition via adaptive circuit compression," 2022, *arXiv:2203.04426*.

[44] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," 2017, *arXiv:1707.03429*.

[45] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Commun.*, vol. 5, no. 1, p. 4213, Jul. 2014.

[46] X. Qiang, X. Zhou, J. Wang, C. M. Wilkes, T. Loke, S. O'Gara, L. Kling, G. D. Marshall, R. Santagati, T. C. Ralph, J. B. Wang, J. L. O'Brien, M. G. Thompson, and J. C. F. Matthews, "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing," *Nature Photon.*, vol. 12, no. 9, pp. 534–539, Sep. 2018.

[47] V. Cimini, M. Valeri, S. Piacentini, F. Ceccarelli, G. Corrielli, R. Osellame, N. Spagnolo, and F. Sciarrino, "Variational quantum algorithm for experimental photonic multiparameter estimation," *NPJ & Quantum Inf.*, vol. 10, no. 1, pp. 1–9, Feb. 2024.

[48] D. F. V. James, P. G. Kwiat, W. J. Munro, and A. G. White, "Measurement of qubits," *Phys. Rev. A, Gen. Phys.*, vol. 64, no. 5, Nov. 2001, Art. no. 052312.

[49] A. G. White, A. Gilchrist, G. J. Pryde, J. L. O'Brien, M. J. Bremner, and N. K. Langford, "Measuring two-qubit gates," *J. Opt. Soc. Amer. B, Opt. Phys.*, vol. 24, no. 2, p. 172, 2007.

[50] J. L. O'Brien, G. J. Pryde, A. Gilchrist, D. F. V. James, N. K. Langford, T. C. Ralph, and A. G. White, "Quantum process tomography of a controlled-NOT gate," *Phys. Rev. Lett.*, vol. 93, no. 8, Aug. 2004, Art. no. 080502.

[51] L.-T. Feng, M. Zhang, X. Xiong, D. Liu, Y.-J. Cheng, F.-M. Jing, X.-Z. Qi, Y. Chen, D.-Y. He, G.-P. Guo, G.-C. Guo, D.-X. Dai, and X.-F. Ren, "Transverse mode-encoded quantum gate on a silicon photonic chip," *Phys. Rev. Lett.*, vol. 128, no. 6, Feb. 2022, Art. no. 060501.

YONG KWON received the B.S. and M.S. degrees in physics from Pukyong National University (PKNU), Busan, Republic of Korea, in 2022 and 2024, respectively.

He was a Scientific Researcher with the Basic Science Research Institute, PKNU, until September 2024. From 2020 to 2022, he started research on quantum information with the Statistical Physics Laboratory, PKNU. During that period, he worked on mutual information and concurrence between two qubit system. Then, he studied quantum computing with the Quantum Computational Science Laboratory, PKNU, and worked in a software platform for small-scale photonic devices. His research interest includes photonic quantum computing from the quantum circuit level to the physical implementation level.

MARTINO BERNARD received the B.S., M.S., and Ph.D. degrees from the University of Trento, Italy, in 2011, 2013, and 2017, respectively. He is currently a Researcher with the Centre for Sensors and Devices, Fondazione Bruno Kessler, Italy. He is the author or co-author of more than 30 articles. His research interests include photonic integrated circuits, with a particular focus on ring resonators and photonic-electronic integration for the quantum technologies. His current research, conducted in the integrated and quantum optics unit, focuses on the integration of photonic circuits with control electronics and detectors.

LEONARDO LIMONGI is currently pursuing the Ph.D. degree with the University of Trento. He is also a part of the Integrated and Quantum Optics Group, Centre for Sensors and Devices, Fondazione Bruno Kessler, Trento. His academic background is in physics and he has been actively involved in quantum information, quantum computation, and quantum optics, since 2020. His current research interest includes photonic integrated platforms in lithium niobate-on-insulator. Specifically, he is dedicated to integrating the laboratory capabilities of generating, manipulating, and detecting quantum states of light on a single chip. He is also deeply passionate about the realization and execution of quantum protocols and the mathematical principles that underpin these processes.

GIOELE PICCOLI received the B.S. degree from the University of Padova, in 2015, and the M.S. degree from the University of Trento, in 2018, where he is currently pursuing the Ph.D. degree in physics. He is a Researcher with the Sensors and Devices Center, Fondazione Bruno Kessler. His research interests include development of integrated photonic platforms based on CMOS-compatible materials and the realization of devices for sensing and quantum applications.

MHER GHULINYAN received the M.S. and Ph.D. degrees in physics from Yerevan State University, Armenia, in 1995 and 1999, respectively. His early-career research with the University of Trento, from 2002 to 2006, focused on optical cavities and superlattices leading to first-time demonstrations of Bloch oscillations, Zener tunneling, and Anderson localization of light. Since 2006, he has been with FBK, in the field of photonics and microresonator optics, with a focus on monolithically integrated classical and quantum photonics. He is the author of several books and book-chapters, more than 130 publications, and several patents. He is a Senior Member of Optica (former OSA).

BYUNG-SOO CHOI received the B.S. degree in computer engineering from Chungnam National University, South Korea, in 1996, and the M.S. and Ph.D. degrees in information and communication from Gwangju Institute of Science and Technology (GIST), South Korea, in 1998 and 2004, respectively. From 2004 to 2006, he was a Postdoctoral Researcher involved in quantum algorithms with the University of York, U.K. From 2011 to 2013, he was a Research Scientist involved in fault-tolerant quantum computation with Duke University, USA. From 2013 to 2015, he was a Researcher involved in quantum-dot qubit device with The University of Tokyo, Japan. From 2015 to 2022, he was a Principal Researcher with the Electronics and Telecommunications Research Institute (ETRI), involved in quantum computing research and development. Since 2022, he has been an Assistant Professor with the Department of Scientific Computing, Pukyong National University. His research interest includes all areas of quantum computing from device implementation to quantum algorithms.