QubiC: An Open-Source FPGA-Based Control and Measurement System for Superconducting Quantum Information Processors

YILUN XU[1] [iD], GANG HUANG[1] [iD], JAN BALEWSKI[1], RAVI NAIK[2], ALEXIS MORVAN[1], BRADLEY MITCHELL[2], KASRA NOWROUZI[1], DAVID I. SANTIAGO[1], AND IRFAN SIDDIQI[1,2]

[1] Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA
[2] University of California at Berkeley, Berkeley, CA 94720 USA

Corresponding author: Gang Huang (ghuang@lbl.gov).

ABSTRACT As quantum information processors grow in quantum bit (qubit) count and functionality, the control and measurement system becomes a limiting factor to large-scale extensibility. To tackle this challenge and keep pace with rapidly evolving classical control requirements, full control stack access is essential to system-level optimization. We design a modular field-programmable gate array (FPGA)-based system called QubiC to control and measure a superconducting quantum processing unit. The system includes room temperature electronics hardware, FPGA gateware, and engineering software. A prototype hardware module is assembled from several commercial off-the-shelf evaluation boards and in-house-developed circuit boards. Gateware and software are designed to implement basic qubit control and measurement protocols. System functionality and performance are demonstrated by performing qubit chip characterization, gate optimization, and randomized benchmarking sequences on a superconducting quantum processor operating at the Advanced Quantum Testbed at the Lawrence Berkeley National Laboratory. The single-qubit and two-qubit process fidelities are measured to be 0.9980 ± 0.0001 and 0.948 ± 0.004, respectively, by randomized benchmarking. With fast circuit sequence loading capability, the QubiC performs randomized compiling experiments efficiently and improves the feasibility of executing more complex algorithms.

INDEX TERMS Engineering software, field-programmable gate array (FPGA), gateware, hardware, noisy intermediate-scale quantum, qubit control.

I. INTRODUCTION

The quantum computer represents a paradigm shifting innovation for computing technology, spurring the development of new breakthroughs in science [1], [2]. Superconducting-circuit-based quantum bits (qubits) are a leading platform for quantum information science, with recent demonstrations of quantum advantage [3]. Qubit control hardware generates and routes complex sequences of radio frequency (RF) signals from room temperature electronics to the quantum processor at cryogenic temperature. As the size and complexity of the quantum system increases, the cost-efficient and compact generation of such signals becomes a bottleneck limiting system extensibility in the near-term noisy intermediate-

scale quantum computing era [4]. Thus, hardware elements in a next-generation design should be carefully matched to the needs of both current and potential future superconducting multiqubit processors, while maintaining the modularity available at the printed circuit board (PCB) design level. However, commercially available laboratory equipment such as arbitrary waveform generators (AWGs) and data acquisition cards are designed for general-purpose test and measurement applications and typically cannot keep pace both in terms of footprint and cost as quantum system complexity increases [5]. A customized quantum engineering solution rooted in extensible primitives is needed for the quantum computing community.

The field-programmable gate array (FPGA) architecture allows for customized solutions capable of growing and evolving with the field [6]. Several FPGA frameworks have been developed for quantum control and measurement [7]–[10]. Nevertheless, current FPGA-based control systems are not fully open to the broader quantum community, so it is hard to make a general toolbox for information scientific discovery [11]–[13]. Understanding the full electronics hardware and software stack of the qubit control systems is essential to the system-level optimization and extensibility [14]. Furthermore, compiling near-term quantum algorithms directly down to the native hardware descriptions via pulse-level control will improve the reliability of program execution [15]. Recent developments have also explored highly integrated system-on-chip (SoC) solution and moved it to the cryogenic stage [16]–[18]. As such efforts mature, they will benefit from a deeper understanding of the control requirements that emerge as system size increases and will, thus, rely on broad conceptual explorations rooted in more flexible hardware platforms.

Here, we develop and test the QubiC (qubit control) system—an open-source FPGA-based RF control system that integrates qubit pulse generation and quantum state measurement [19]. Leveraging state-of-the-art FPGA technology, QubiC provides fully parametric waveform generation and allows researchers to access all the control layers. This scalable and cost-effective system will be a potential open-source toolbox for the quantum community.

II. QUBIC SYSTEM

We design an FPGA-based qubit control system called QubiC to integrate the execution of gate-based algorithms with native quantum hardware implementation. QubiC is an open-source system, which can accommodate users and developers from different layers to enable efficient codesign. As the starting point, we develop and test a prototype system of a single control unit to demonstrate basic functionality and evaluate performance.

The room temperature electronics hardware, the FPGA gateware, and the engineering software are integrated in the QubiC prototype system so as to realize the desired RF pulses to control and measure qubits. This prototype system also provides a platform to explore and optimize real-time feedback control such as fast reset [20], [21] and error correction algorithms.

A. ELECTRONICS HARDWARE

The QubiC prototype hardware employs the heterodyne technique to generate and detect RF signals in a compact manner. It includes three basic building modules: the FPGA/analog-to-digital converter (ADC)/digital-to-analog converter (DAC) module to generate/detect the intermediate frequency (IF) signal, the RF mixing module to convert the signal frequency to/from the target frequency, and the local oscillator (LO) generation module to provide low-noise LO signals, as shown in Fig. 1. A series of commercial
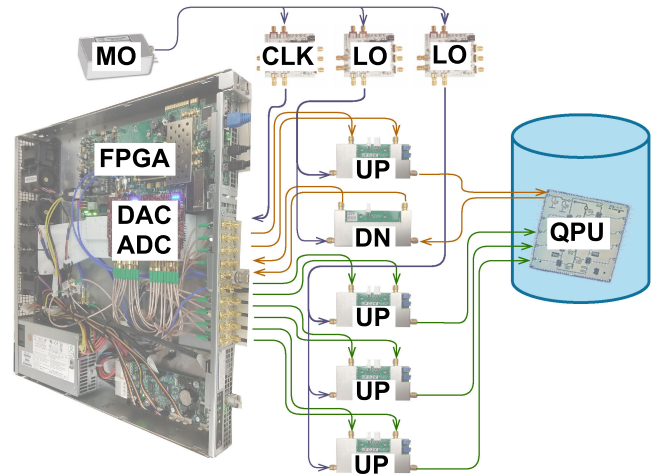

FIGURE 1. QubiC prototype hardware. MO: master oscillator, CLK: clock, LO: local oscillator, FPGA: field-programmable gate array, DAC: digital-to-analog converter, ADC: analog-to-digital converter, UP: upconverter, DN: downconverter, QPU: quantum processor unit. The yellow line indicates the measurement path, while the green line defines the qubit control path.

off-the-shelf (COTS) evaluation boards are selected and assembled with in-house-developed RF mixing modules to test functionality and benchmark performance. Using these COTS evaluation boards enabled us to rapidly develop a hardware platform to study and develop the qubit control logic, while leaving plenty of room for channel density and cost optimization in future iterations.

1) FPGA AND ADC/DAC MODULE

The Xilinx VC707 and the Abaco Systems FMC120 boards are chosen for their computational capability and precision analog performance. The VC707 evaluation board contains a Xilinx Virtex-7 FPGA [22]. Each FMC120 board has four channels of high-speed (1.25 GS/s) 16-bit DACs and four channels of high-speed (1 GS/s) 16-bit ADCs [23]. One VC707 and two FMC120 boards are assembled in a 1U rack mount computer chassis with a customized cooling manifold to meet the FMC120 board cooling requirements. An external 1-GHz clock is provided to the chassis to run the DACs/ADCs at 1 GS/s and to run the FPGA digital signal process (DSP) at 250 MHz. Compared with the latest RF SoC, this combination gives us more choices with respect to DAC/ADC selection, particularly when tailoring hardware for specific future applications.

2) RF MIXING MODULE

We develop compact RF up and down mixing modules that integrate the in-phase/quadrature (I/Q) mixer, IF/LO/RF power-level adjustments, and dc bias fine tuning on a 40 mm × 80 mm four-layer PCB board with electromagnetic interference shielding. The RF mixing module is designed to work with RF and LO frequency between 2.5 and 8.5 GHz. Typical image rejection and adjacent channel isolation are

TABLE 1 LO RMS Jitter

Modules	RMS Jitter (@6.52 GHz / @5.50 GHz)	
	Integration Bandwidth (1 Hz–10 MHz)	Integration Bandwidth (100 Hz–10 MHz)
Wenzel+LMX2595	1.0 ps / 0.9 ps	60 fs / 54 fs
Keysight N5183B	1.0 ps / 0.9 ps	31 fs / 41 fs

measured to be $\sim$27 dBc and $\sim$50 dB. The RF mixing module provides 5×10^{-4} (V_{pp}/V_{mean}) amplitude stability and 1×10^{-3} radian (pk-pk) phase stability [24].

3) LO GENERATION MODULE

The noise performance of the LO generation module is critical for high-fidelity qubit operation because imperfections will directly map on to the RF/IF control signal through the RF mixing module. The LO frequency also needs to be adjustable from experiment to experiment to accommodate different chip designs. Additionally, some qubit operations will require multiple different LO frequencies synchronized to each other. At the same time, the size and cost of LO generation should also be factored into design consideration.

We, thus, use multiple phase-locked loops (PLL) with a shared master oscillator (MO) to generate the LO and clock to meet these frequency and low-noise operation requirements. The Wenzel (501–16843) 100-MHz ultralow-noise crystal oscillator is used as the MO for the prototype system [25]. The Texas Instruments LMX2595 evaluation modules are used as a PLL given its 20-GHz wide frequency range and low-noise performance [26]. Specifically, we measure the RMS jitter of the "Wenzel+LMX2595" module at the LO frequency at the qubit readout resonator frequency (6.52 GHz) and qubit drive frequency (5.50 GHz). As shown in Table 1, the RMS jitters of the "Wenzel+LMX2595" module are 1.0 ps (1 Hz–10 MHz) and 60 fs (100 Hz–10 MHz), which are comparable with the widely used commercial Keysight signal generator [27]. The relationship between the LO phase noise metrics and the corresponding qubit dephasing spectral densities has been studied, and the formula to estimate the operational fidelity bounds on superconducting qubits was derived in [28]. Following the analysis and using the measured phase noise of the "Wenzel+LMX2595" configuration, we estimate that for the $\hat{X}$ gate shorter than 10 ms, the LO phase noise contributes $< 10^{-5}$ to the gate infidelity, which is much lower than the typical gate infidelity in superconducting qubits. This implies that the LO module phase noise is not the dominating factor to the gate fidelity. Moreover, the "Wenzel+LMX2595" module is much more compact in size and lower in cost.

4) MODULE SYNCHRONIZATION

To scale the system up, we design the module-to-module synchronization using three layers of protocols [29].

1) JESD204B subclass 1 [30] is used to synchronize multiple DAC chips via the external SYSREF signal.
2) Direct general-purpose input/output triggers are used to pass and fan out the event information, which requires low latency among modules.
3) Fiber-based synchronization using a protocol similar to the white rabbit [31] system is under development to lock the clock phases among the modules and also provide a high-speed data communication path.

B. FPGA GATEWARE

The gateware programmed on the FPGA is the lowest level to implement digital tuneup sequences and algorithmic protocols. Modularity is the key to designing reusable and maintainable FPGA gateware code. The QubiC FPGA gateware is written in the Verilog programming language, and it is separated into three modules: the board support package (BSP), the DSP, and the host interface (HOI).

1) The *BSP* is hardware specific and used to implement low-level hardware configuration and initialization.
2) The *DSP* implements basic qubit control and measurement functions, and it is designed to be independent of the low-level FPGA/ADC/DAC selection. The DSP runs in a single clock domain, and the HOI and BSP handle the clock domain crossing for the required registers and also buffer the transmitted and received data.
3) The *HOI* handles all the input and output to and from the host computer, which is the interface that hosts higher level software.

The QubiC gateware is capable of generating waveform from parameters, which leads to the benefit of pulse sequence reuse. The QubiC FPGA DSP is flexible and extensible. In particular, the QubiC system has the capability of the fast reset by implementing the qubit status classification and the feedback loop in the FPGA gateware with low latency [32]. In this section, we will focus on the gateware DSP module and the HOI associated with it.

1) GATEWARE DSP

The basic requirements of qubit control and measurement are to generate pulses at a specified carrier frequency with arbitrary amplitude modulation and to also synchronously detect the pulse after it goes through the readout resonator. The hardware module shifts frequencies between the RF and IF, while the gateware DSP module is responsible for the generation and processing of the data stream between the IF signal and the baseband signal. The QubiC gateware DSP block diagram is illustrated in Fig. 2.

The digital modulation/demodulation between the baseband envelope and the IF data stream is realized by a module called the processing element. Processing elements can be configured to execute digital up- or downconversions. When used as the upconverter, the module generates IF pulses in-phase (I_{IF}) and quadrature-phase (Q_{IF}) components from the gate-specific baseband envelope $I_{BB} + jQ_{BB}$, the initial
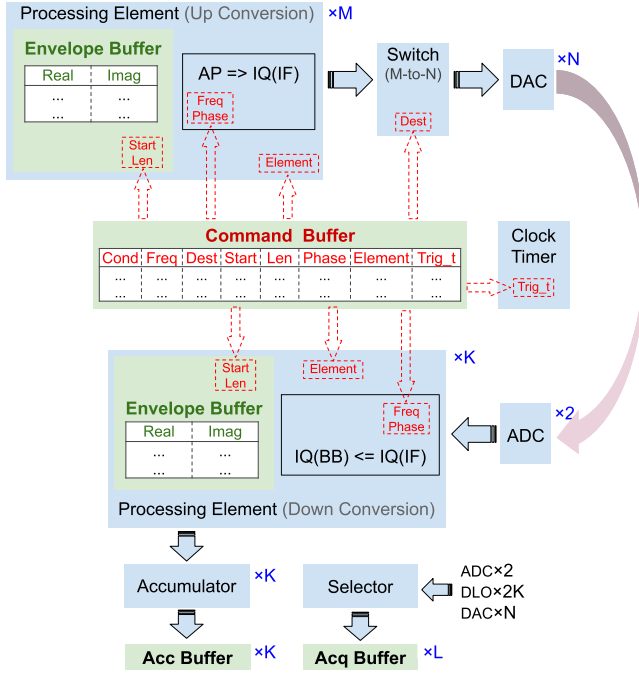
FIGURE 2. QubiC gateware DSP block diagram. The *processing elements* are employed as the up- or downconverters in the digital domain. *Real* and *Imag* denote the real and imaginary parts of the complex envelope value in the *envelope buffer*. *AP* represents the amplitude (the magnitude of the complex envelope) and the phase (computed by the carrier frequency and initial phase from the command). *IQ(IF)* and *IQ(BB)* are the in-phase and quadrature-phase components of IF and baseband signals, respectively. The fields in the *command buffer* are condition (fast reset flag), frequency, destination, start, length, phase, element, and trig_t from the most significant bit (MSB) to the least significant bit (LSB). The accumulated values that comprise the integration of the baseband I/Q series over the corresponding DLO are stored in the *acc buffer*. The *acq buffer* serves as a live oscilloscope for the ADC/DLO/DAC raw data. *M* upconversion *processing elements* are switched to *N* DACs with the dynamic mapping. *K* downconversion *processing elements* can be assigned to process *K*-qubit readout data simultaneously through the same ADC pair.

phase ϕ_0, and the digital local oscillator (DLO) frequency, as in

$$I_{IF} + jQ_{IF} = (I_{BB} + jQ_{BB})e^{\phi_0}e^{j\omega_{DLO}t}. \tag{1}$$

The carrier frequency is determined by the physical property of the qubits or the readout resonator. The carrier phase naturally rotates as time evolves; additionally, the engineering software can inject additional phase offsets to implement virtual Z gates [33]. The pulse envelope is defined or optimized through the qubit gate calibration process and stored in the FPGA memory point-by-point as complex numbers, each with in-phase and quadrature-phase terms. Moreover, this pulse envelope can be used repetitively by indexing the same address area, even if the carrier frequency or phase changes.

The calculated result then feeds into any DAC through an *m*-to-*n* switch. Signals from multiple processing elements can be sent to the same DAC and added together. For each pulse to be generated, the software needs to specify

the processing element, and the DAC to be used for the destination.

When the processing module is used in the downconversion mode, it receives the qubit signals from the ADC and recovers the baseband I and Q components, which is exactly the opposite operation, as defined by

$$I_{BB} + jQ_{BB} = (I_{IF} + jQ_{IF})e^{-j\omega_{DLO}t}. \tag{2}$$

Then, the baseband I/Q series are integrated by a vector accumulator, and the results are stored in an "accumulation buffer" (acc buffer) for the corresponding channel to determine the qubit final state. The raw ADC, DAC, and DLO values are available to the host computer through "acquisition buffers" (acq buffer), and the integrated I/Q data are available to the host computer through the acc buffer.

In the QubiC gateware, the processing element is flexible in the waveform generation. Each processing element has a waveform storage and a phase rotator, but only puts out one waveform at a time. Once stopped, it is available to replay another waveform with another configuration. Waveform envelopes drawn from that memory can be variable lengths in order to start and stop arbitrarily. Currently, the envelope buffer of each processing element is 1k deep and 32 bits wide. The upper and lower 16 bit words of each envelope point represent the real (I) and imaginary (Q) parts of the pulse envelope, respectively. The conversions between amplitude/phase and I/Q of a complex signal as in (1) and (2) are implemented with the Coordinate Rotation Digital Computer algorithm on FPGAs [34], [35].

2) HOST INTERFACE

A qubit algorithm typically consists of a series of qubit gates and measurements, which are eventually realized by the sequence of RF pulses. In general, a gate may consist of a series of pulses. Gate-level experiments adopt a precalibrated gate pulse, while pulse-level experiments need access the parameters for each pulse. The host computer interface needs to be designed to accommodate both types of experiments by way of parametric waveform generation.

In the current QubiC gateware, each pulse is defined by the pulse start time, the envelope information, and the carrier information. We define a 128-bit command as the interface between the lower level gateware on an FPGA and the higher level software on a host computer:

1) **Trig_t** (24 bits): pulse start time relative to the start of the whole sequence;
2) **Start** (12 bits): the start address of the envelope data in the envelope buffer;
3) **Length** (12 bits): the length of envelope buffer to play for this pulse;
4) **Frequency** (24 bits): the carrier frequency in 1 GHz represented by a 24-bit word. Each step is $1\,\text{GHz}/2^{24} \approx 60$ Hz;
5) **Phase** (14 bits): the carrier initial phase as the ϕ_0 shown in (1). Each step is $360°/2^{14} \approx 0.022°$;

6) **Element** (8 bits): processing element index;
7) **Destination** (2 bits): destination IQ pair index;
8) **Condition** (1 bit): flag for conditional gate, used for the fast reset gate (only available on the specified version of bit file);
9) **Reserved**: 31 bits.

The 128-bit command provides the parametric waveform generation from the lowest FPGA level. The QubiC reuses sequences by updating waveform parameters via commands without recompiling or resending the waveform data.

The commands defining quantum circuits are stored in a 64k deep 128-bit wide buffer on an FPGA. The envelope buffers are also written from the host computer and stored in each processing element. All commands repeat at a defined period until the acc buffer is full. The acc buffer can be cleared by the software after reading the data.

Writing to the command/envelope buffer to generate pulses and reading from the acc/acq buffer to collect responses are the lowest level software interface provided by the FPGA gateware. At this level, we have to adhere to the restrictions imposed by the gateware implementation, for example, the maximum pulse envelope length and command depth, the frequency and phase resolution, the longest sequence time, etc. The complexity of the restrictions imposed at this lower level makes it more difficult to use directly. However, the advantage is that this layer has full access to the FPGA directly and mainly serves developers who implement software directly on top of the QubiC gateware.

C. ENGINEERING SOFTWARE

The engineering software runs on the host computer, implementing the FPGA buffers/registers input/output and providing the software interface to higher level software or algorithms. QubiC engineering software consists of an application programming interface (API), a portfolio of quantum characterization verification and validation (QCVV) experimental scripts, and a graphical user interface (GUI).

1) APPLICATION PROGRAMMING INTERFACE

The Python-3-based API compiles the quantum processor gate pulse specification and the quantum circuit description to gateware commands and pulse envelopes.

The quantum processor gate pulse specification contains two parts: the frequencies (readout resonator and qubit drive) that represent the specific properties of each qubit and the gates that drive interactions between the control system and the qubits. Each gate is a series of pulses defined by its destination, timing, carrier, amplitude, and envelope. The quantum processor and gate specifications are captured in the JSON format file, which is generated or updated during the chip calibration.

The quantum circuit description can be written in our native function call or be imported from TrueQ [36]. The measurement result can also be sent back to TrueQ to allow for postprocessing of all the data. A similar interface to Qiskit/OpenQASM/Cirq [37]–[39] is under development.

The compiler is the bridge to translate the circuit description into the envelope and command buffers. There are two distinct types of compilers in the QubiC API. The first one is the OPTiMize gate (OPTM) compiler, which is used during the chip characterization and gate optimization. The OPTM compiler accommodates multiple different pulse envelopes, in length, shape, or other parameters. The envelope buffers from all the processing elements are collected together and allocated dynamically so that they can all be used in the specified qubits. In the OPTM compiler, the circuit description is written in our native function call. One can add a gate onto a sequence at a specific time and modify the gate parameters at the compiling time. The validation test is necessary in the OPTM compiler and so it takes longer to compile.

Another compiler is the RUN Circuit (RUNC) compiler, which is used to run a quantum circuit with calibrated gates on all the qubits simultaneously. Since only a few predefined gates are applied for each qubit, the envelope memory can be preallocated statically to reduce the compiling time. In the RUNC compiler, the circuit description can be imported from other languages, which need to be translated to the QubiC format. The validation tests during compiling are skipped to execute long circuits efficiently in the RUNC compiler.

The OPTM and RUNC compilers coexisting in the QubiC software bring in the flexibility and efficiency for distinct control flows. The OPTM compiler targets a comprehensive chip calibration via tweaking multiple parameters with high accuracy in a broad range, while the RUNC compiler specifies a dedicated structure to perform the advanced quantum algorithms in an efficient manner. Both the OPTM and RUNC are three-step compilers, which are illustrated in the Appendix.

2) QCVV EXPERIMENT SUPPORT

A calibrated quantum information processor is the necessary prerequisite to perform QCVV experiments. In the current QubiC engineering software, a series of scripts are developed to characterize the quantum information processor and optimize the gates to support the superconducting QCVV experiments. This type of characterization and optimization routine can also be adapted for other architectures at hand. Quantum processor characterization begins with time alignment, which measures the latency for the readout signal and aligns it with the DLO. The single-tone experiment involves measuring all the readout resonator frequencies on the readout bus [11]. We also conduct a punch-out experiment, where the drive power is increased significantly to suppress the Josephson nonlinearity of the qubit, yielding a simple spectroscopic signature of viable qubits, and a coarse adjust of the readout amplitude needed to achieve linear operation [40]. The qubit drive frequency can be extracted from a two-tone spectroscopy experiment or the chevron pattern
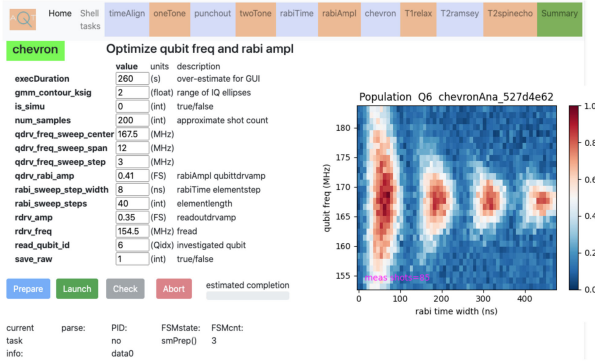
FIGURE 3. Screenshot of the GUI while executing the chevron task.

obtained from coherent oscillations. We drive Rabi oscillations with different pulse lengths or amplitudes and derive a Gaussian mixture model for qubit state discrimination. A readout correction is employed to statistically calibrate out the effect of classical readout bit-flip errors.

A bounded minimization method [41] is used to adjust the readout amplitude and frequency together with the qubit drive frequency to maximize the Rabi oscillation contrast. To increase the gate repeatability, the gate pulse length is set to be a multiple of the DAC sampling rate and the FPGA clock rate. The qubit drive amplitude is optimized for each gate by stacking multiple identical gates at the target pulse width to maximize the recovery probability [42], [43]. With the calibrated single-qubit gate, we can measure the qubit coherence time including the qubit relaxation time (T_1) and dephasing time (T_2). The residual oscillation frequency measured by the Ramsey experiment can help us finely tune the qubit drive frequency. The single-qubit gate quality can be validated by the AllXY experiment [44] and by a conventional randomized benchmarking (RB) sequence [45].

Two-qubit gate calibration starts with optimizing the cross resonance (CR) pulse to reach full entanglement [46]. The single-qubit gate parameters around the CR pulse to construct a CNOT gate are identified by a curve fitting of a full XY-plane measurement of a degenerated state followed by a CR pulse [47]. With the circuits from TrueQ, we can execute a two-qubit RB measurement to obtain the two-qubit process fidelity.

3) GRAPHICAL USER INTERFACE

The QubiC GUI is being developed to streamline and simplify the complex sequences of hardware calibration tasks.

The GUI client, which is implemented in Bootstrap and JavaScript, can be used to remotely control the quantum hardware connected by the internet.

The operator can change the qubit bias point in real time, initialize/terminate measurements, and display the results. Fig. 3 shows an example of the user-facing GUI webpage for generating a chevron pattern for coherent oscillations.

The GUI server, which is implemented in the Python web framework Flask, runs on a Linux machine located in the
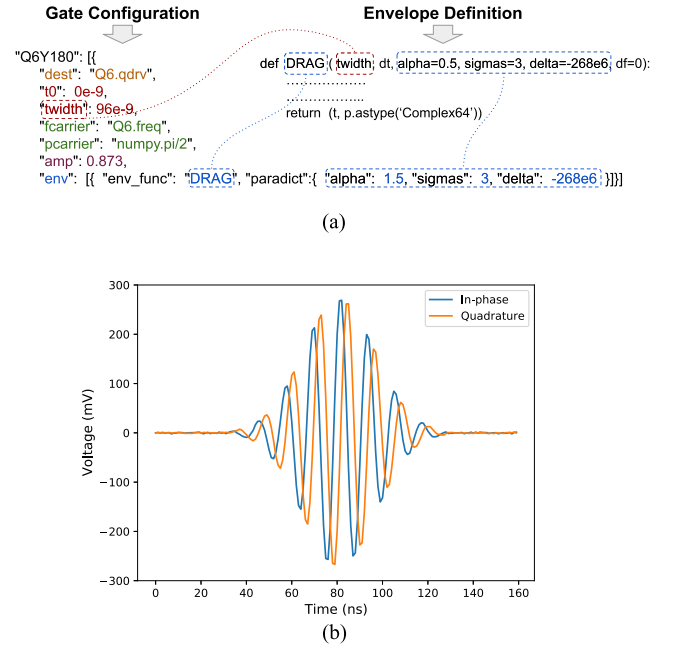

(a)

(b)

FIGURE 4. Parametric waveform generation. (a) Gate configuration and envelope definition. The destination parameter is the destination qubit or readout to be controlled, which is named as dest in the gate configuration. The timing parameter consists of the pulse starting time (t0) and the pulsewidth (twidth). The carrier frequency (fcarrier) and the initial phase (pcarrier) values are encoded in the carrier parameter. The pulse amplitude (amp) is a fraction of DAC full scale. The envelope function (env) can be any Python function (commonly used or customized) returning a NumPy array. (b) Y180 gate with DRAG envelope measured by ADCs. The ADC digitizes the IF signal at a sampling rate of 1 GS/s.

proximity of the QubiC FPGA chassis. For any measurement requested by the client, the configuration and the measured raw IQ pairs are saved in local files in the YAML and HDF5 data formats, respectively. The postprocessing is done on the server side, and the plots are made available for the remote client. Any file saved by the GUI server is time-stamped, which allows for the retrieval of the history or tracking of the time stability of the hardware calibration constants. The finite-state machine governs the internal logic of the server, and the experiments are executed asynchronously allowing for an early termination by the client.

III. BENCH TEST

Putting the electronics hardware, FPGA gateware, and engineering software together, we used the QubiC system to generate the desired RF pulses to control and measure the qubits. Specifically, we performed bench testing to demonstrate fully parametric waveform generation using the destination, timing, carrier, amplitude, and envelope parameters.

A simple example is shown in Fig. 4(a), where we aim to generate a Y180 gate on Q6 (which is a $180°$ single-qubit gate around the Y-axis of the Bloch sphere on a qubit labeled Q6) starting at some time (defined in the circuit) after the circuit starts. The in-phase and quadrature components are sent to the IQ channels for the Q6 qubit drive "Q6.qdrv"

(dest). The pulse starts immediately (t0) after the gate starts and lasts 96 ns (twidth). The carrier frequency is specified as "Q6.freq" (fcarrier), while the initial phase is "numpy.pi/2" (pcarrier). The overall amplitude of the pulse is 0.873 of the DAC full scale. The amplitude modulation is applied with a "DRAG" (derivative removal by adiabatic gate [48]) envelope (env) with appropriate parameters. This simple mapping from the gate configuration to the envelope definition function enables us to use any user provided Python function as the pulse envelope to construct a gate.

The aforementioned Y180 gate pulse is generated by DAC pairs as IF I/Q signals. The IF signals are then upconverted in the up mixing module to an RF signal. In order to measure the Y180 gate pulse, the RF signal is directly connected back to the RF down mixing module, followed by the ADC digitization. As shown in Fig. 4(b), we can successfully generate the parameterized RF pulse, which is essential to the pulse-level experiments.

IV. TEST WITH QUANTUM PROCESSOR

A. RANDOMIZED BENCHMARKING

Leveraging standard qubit characterization and gate optimization protocols [47], the QubiC can automatically find and tune a two-qubit system. In order to validate the performance of the QubiC system on quantum hardware, we perform streamlined RB [49] on two of the qubits in an 8-qubit quantum processor [50]. As shown in Fig. 5, the single-qubit process fidelity [51], [52] is measured to be 0.9980 ± 0.0001, while the two-qubit process fidelity is measured to be 0.948 ± 0.004. The RB results demonstrate that the QubiC system can control and measure qubits efficiently, and it should be capable of delivering high-fidelity gates on state-of-the-art processors.

B. RANDOMIZED COMPILING

Randomized compiling (RC) is newly developed technique in quantum computing to tailor and mitigate noise [53], [54]. The RC protocol efficiently compiles an algorithm circuit in a number of different, but equivalent variations. Instead of executing just the original "bare" circuit, the protocol requires execution of all the variations of the circuit. On the conventional control hardware side, it may be challenging to create and upload the various pulse sequences without significant overhead. Here, we show that QubiC can execute this general, yet sophisticated compilation protocol on a quantum computer in an efficient manner.

On the host computer, the circuits are randomly compiled ("Compile") and transpiled to a device's native gateset ("Transpile") [36], [55]. Afterward, QubiC transfers the native gate sequences to QubiC command sets ("Transfer") and then generates the sequences ("SeqGen"). The QubiC software prepares the hardware and transports the command and the memory to the QubiC hardware ("Run"). The qubit response is acquired from the FPGA ("Acquire") and processed by QubiC software to discriminate the state ("Process").
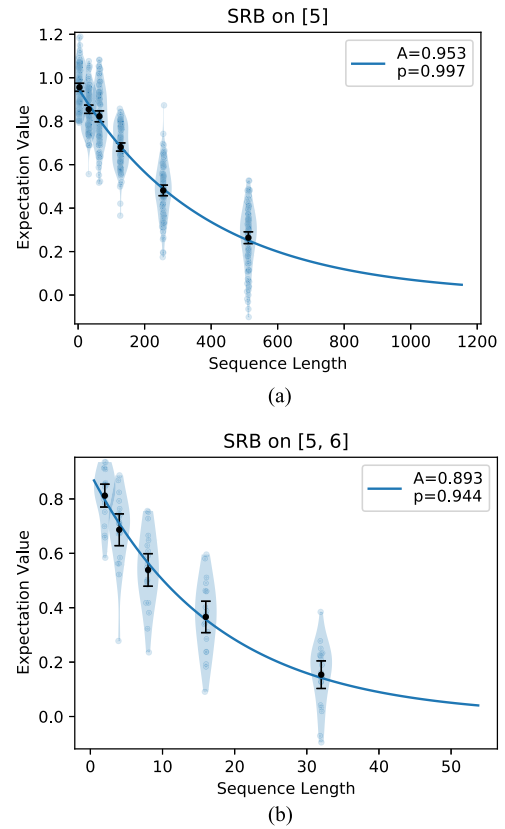

FIGURE 5. Streamlined RB results. Each point corresponds to a measurement result of a random quantum circuit. The curve was fitted with an exponential decay function Ap^m. The sequence length is expressed in terms of Clifford gates. (a) Single-qubit RB. 100 points per length, 1k shots per point. (b) Two-qubit RB. 20 points per length, 2k shots per point.

To evaluate RC efficacy, we compare the measured probability $P(x)$ with the ideal probability $P_{ideal}(x)$ for the given circuit to calculate the total variational distance (TVD), which is defined as

$$\text{TVD} = \frac{1}{2} \sum_{x \in X} |P(x) - P_{ideal}(x)| \qquad (3)$$

where x is the bit string. The TVD parameterizes the probability of measuring an incorrect solution.

We perform the two-qubit RC on the quantum processor and measure the TVD distributions of bare circuits and RC circuits, as shown in Fig. 6. Compared to the bare circuits, lower TVD values are observed from the RC process, which means that the output distributions are more accurate.

Furthermore, QubiC can execute the RC protocol relatively quickly, as shown in Fig. 7. The total execution time grows linearly with the increase of the number of RC circuits or the number of two-qubit gates. The "Compile" and the "Transpile" account for the majority of the execution time, which are limited by the host computer performance. The major part of the QubiC execution time is the "Run," while command writing is the most time-consuming task in the "Run." One can observe that the "Run" time increases
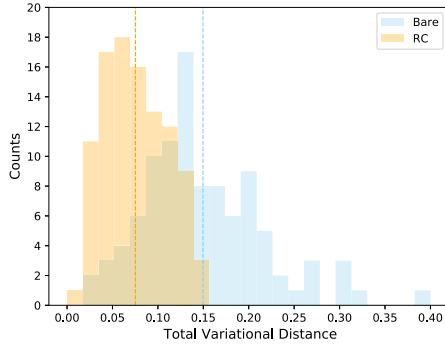
FIGURE 6. TVD distributions obtained from two-qubit RC. 100 different random bare circuits were generated for sufficient statistics, with an average of 20 RC circuits for each bare circuit. The circuit depth is 5 (two-qubit gates). Each measurement is the average of 1024 shots. The TVD values are 0.150 ± 0.067 for bare circuits and 0.075 ± 0.034 for RC circuits.
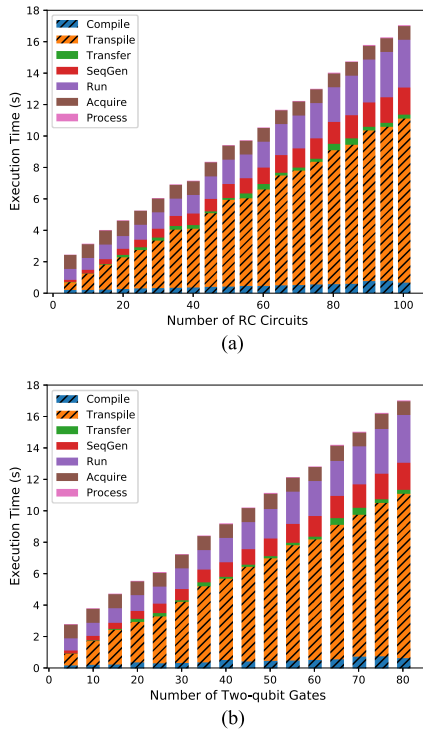

FIGURE 7. Execution time of two-qubit RC on QubiC. (a) measures the execution time with different number of RC circuits when the circuit depth is 80 (two-qubit gates). (b) measures execution time with different number of two-qubit gates when 100 RC circuits are averaged for each bare circuit. Each measurement is the average of 1k shots. The bar with slash inside is operated on the host computer, which is the main limiting factor in the execution time.

linearly with the number of RC circuits and the number of two-qubit gates, respectively, since the command depth also grows linearly. The speed of the "Run" portion is currently limited by the constraint that the load command only accepts 16k commands and takes ~55 ms to execute. This can be made more efficient in future versions of the software. For a 600-μs relaxation time, we can load ~100 circuits at a time. Currently, QubiC takes <17 s for 80 cycles and 100 RC circuits with 1024 shots, which is efficient and promising.
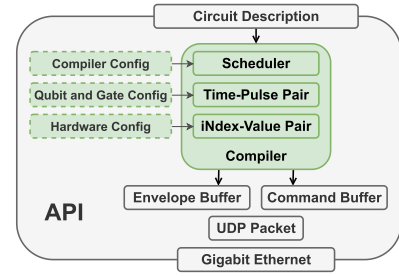

FIGURE 8. Compiling procedure.

V. CONCLUSION

We develop an open-source FPGA-based QubiC system, which bridges the quantum algorithm and the quantum hardware implementation layers of the computing stack. The QubiC system includes the room temperature electronics hardware, the gateware, and the engineering software running on it to implement the qubit control pulses and also to perform quantum state readout. QubiC allows researchers to access the full electronics hardware, gateware, and software stack, which will enable the execution of a broader class of computation experiments while also facilitating the implementation codesign at each level of the stack in next-generation systems. The QubiC prototype system functionality and performance were demonstrated by measuring single-qubit and two-qubit process fidelities. We obtained RB values of 0.9980 ± 0.0001 and 0.948 ± 0.004, respectively, on an 8-qubit superconducting quantum processor operating at the Advanced Quantum Testbed at the Lawrence Berkeley National Laboratory. Moreover, we demonstrated that QubiC has the capability to run advanced quantum algorithms such as RC with promising execution speed. QubiC can accommodate the full spectrum of users and developers and will be a potential open-source toolbox for the quantum community.

APPENDIX
COMPILING PROCEDURE

The compiler is the bridge in the software API to transfer the circuit description into the envelope and command buffers. The compiling procedure is shown in Fig. 8. The three steps of the compilation are as follows.

1) *Scheduler:* The first step is scheduling, which takes in the circuit description together with the compiler configuration to generate the gate sequences with relative timing information. For the circuits imported from other language, which does not include the timing information, such as OpenQASM, this step is necessary. However, if the circuit is written using native function calls, this step can be skipped by assigning an explicit start time to each gate.

2) *Time–pulse (TP) pair:* The second step is to combine the gate sequences with the gate configuration to generate a list of pulses described as TP pairs. The qubit chip and gate configuration are generated from the

qubit characterization and gate optimization. The gate configuration contains information about the pulses used in the gate, including the carrier, the envelope, and the relative timing. The TP description contains the parameterized gate information, which can then be implemented using different electronics hardware setups. The conventional AWG contains idle periods of zero or constant output. Rather than inefficiently storing repeated values in the waveform memory, we instead employ the TP pairs to store the time and pulse envelope value in the waveform memory. At this level, the TP pair can still be human readable, so the user who is familiar with the pulse should be able to understand the TP description and also check the relative pulse timing.

3) *iNdex–value (NV) pair:* The third step includes the hardware configuration and generates the NV pair as the buffer value for the lower level. This is a hardware-dependent layer. The hardware configuration contains the sampling rate, the physical wiring diagram between the DACs and the qubits, and the gateware register maps. This only needs to be updated when the quantum chip or the chip wiring changes. Separating this step and configuration gives us the opportunity to run the same pulse on different hardware for comparison.

The outputs of the three-step compiler are the envelope buffer, the command buffer, and the simulator to visualize the pulse from the buffer values. The commands normally sent to the FPGA are redirected to the on-host analysis and visualization package, when switching to the simulation mode, which allows verification of correctness of encoded gates. The final output from the API is the user datagram protocol (UDP) packet, which will be sent out over Gigabit Ethernet to the FPGA hardware.

REFERENCES

[1] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018, doi: 10.22331/q-2018-08-06-79.

[2] F. Arute *et al.*, "Hartree-Fock on a superconducting qubit quantum computer," *Science*, vol. 369, no. 6507, pp. 1084–1089, 2020, doi: 10.1126/science.abb9811.

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

[4] A. Vainsencher *et al.*, "Superconducting qubit control electronics—Part 1/2: System overview and control hardware," *Bull. Amer. Phys. Soc.*, vol. 64, no. 2, 2019, Art. no. V26.00005. [Online]. Available: https://meetings.aps.org/Meeting/MAR19/Session/V26.5

[5] C. Ryan, B. R. Johnson, D. Ristè, B. Donovan, and T. A. Ohki, "Hardware for dynamic quantum computing," *Rev. Sci. Instrum.*, vol. 88, no. 10, 2017, Art. no. 104703, doi: 10.1063/1.5006525.

[6] Y. Salathé *et al.*, "Low-latency digital signal processing for feedback and feedforward in quantum computing and communication," *Phys. Rev. Appl.*, vol. 9, no. 3, 2018, Art. no. 034011, doi: 10.1103/PhysRevApplied.9.034011.

[7] D. Riste *et al.*, "Deterministic entanglement of superconducting qubits by parity measurement and feedback," *Nature*, vol. 502, no. 7471, pp. 350–354, 2013, doi: org/10.1038/nature12513.

[8] L. Steffen *et al.*, "Deterministic quantum teleportation with feed-forward in a solid state system," *Nature*, vol. 500, no. 7462, pp. 319–322, 2013, doi: 10.1038/nature12422.

[9] C. Bultink *et al.*, "Active resonator reset in the nonlinear dispersive regime of circuit QED," *Phys. Rev. Appl.*, vol. 6, no. 3, 2016, Art. no. 034008, doi: 10.1103/PhysRevApplied.6.034008.

[10] R. Gebauer *et al.*, "State preparation of a fluxonium qubit with feedback from a custom FPGA-based platform," *AIP Conf. Proc.*, vol. 2241, 2020, Art. no. 020015, doi: 10.1063/5.0011721.

[11] Y. Chen *et al.*, "Multiplexed dispersive readout of superconducting phase qubits," *Appl. Phys. Lett.*, vol. 101, no. 18, 2012, Art. no. 182601, doi: 10.1063/1.4764940.

[12] N. Ofek *et al.*, "Extending the lifetime of a quantum bit with error correction in superconducting circuits," *Nature*, vol. 536, no. 7617, pp. 441–445, 2016, doi: 10.1038/nature18949.

[13] X. Fu *et al.*, "An experimental microarchitecture for a superconducting quantum processor," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2017, pp. 813–825, doi: 10.1145/3123939.3123952.

[14] K. Bertels *et al.*, "Quantum computer architecture toward full-stack quantum accelerators," *IEEE Trans. Quantum Eng.*, vol. 1, pp. 1–17, 2020, doi: 10.1109/TQE.2020.2981074.

[15] P. Gokhale, A. Javadi-Abhari, N. Earnest, Y. Shi, and F. T. Chong, "Optimized quantum compilation for near-term algorithms with openpulse," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2020, pp. 186–200, doi: 10.1109/MICRO50266.2020.00027.

[16] J. Bardin *et al.*, "A 28 nm bulk-CMOS 4-to-8 GHz <2 mW cryogenic pulse modulator for scalable quantum computing," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2019, pp. 456–458, doi: 10.1109/ISSCC.2019.8662480.

[17] B. Patra *et al.*, "A scalable cryo-CMOS 2-to-20 GHz digitally intensive controller for 4 frequency multiplexed spin qubits/transmons in 22 nm FinFET technology for quantum computers," in *Proc. IEEE Int. Solid- State Circuits Conf.*, 2020, pp. 304–306, doi: 10.1109/ISSCC19947.2020.9063109.

[18] S. J. Pauka *et al.*, "A cryogenic CMOS chip for generating control signals for multiple qubits," *Nat. Electron.*, vol. 4, no. 1, pp. 64–70, 2021, doi: 10.1038/s41928-020-00528-y.

[19] Git repository of the QubiC system. Accessed: Oct 14, 2021. [Online]. Available: https://gitlab.com/LBL-QubiC

[20] R. Vijay *et al.*, "Stabilizing Rabi oscillations in a superconducting qubit using quantum feedback," *Nature*, vol. 490, no. 7418, pp. 77–80, 2012, doi: 10.1038/nature11505.

[21] D. Riste, C. C. Bultink, K. W. Lehnert, and L. Dicarlo, "Feedback control of a solid-state qubit using high-fidelity projective measurement," *Phys. Rev. Lett.*, vol. 109, no. 24, 2012, Art. no. 240502, doi: 10.1103/PhysRevLett.109.240502.

[22] VC707, Xilinx Inc., San Jose, CA, USA. Accessed: Dec. 21, 2020. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html

[23] FMC120, Abaco Systems, Inc., Huntsville, AL, US A. Accessed: Dec. 21, 2020. [Online]. Available: https://www.abaco.com/products/fmc120-fpga-mezzanine-card

[24] Y. Xu, G. Huang, D. I. Santiago, and I. Siddiqi, "RF mixing modules for superconducting qubit room temperature control systems," 2020, *arXiv:2101.00066*.

[25] *ULN P/N:501-16843*, Wenzel Associates, Inc., Austin, TX, USA. Accessed: Dec. 21, 2020. [Online]. Available: https://wenzel.com/support/legacy-parts/

[26] *LMX2595 EVM*, Texas Instruments, Inc., Dallas, TX, USA. Accessed: Dec. 21, 2020. [Online]. Available: https://www.ti.com/lit/ug/snau219b/snau219b.pdf

[27] *MXG N5183B*, Keysight Technologies, Santa Rosa, CA, USA. Accessed: Apr. 27, 2021. [Online]. Available: https://www.keysight.com/us/en/assets/7018-04096/data-sheets/5991-3131.pdf

[28] H. Ball, W. D. Oliver, and M. J. Biercuk, "The role of master clock stability in quantum information processing," *npj Quantum Inf.*, vol. 2, no. 1, 2016, Art. no. 16033, doi: 10.1038/npjqi.2016.33.

[29] G. Huang *et al.*, "Clock synchronization and data exchange between FPGA modules for superconducting qubit control," *Bull. Amer. Phys. Soc.*, 2021, Art. no. R33.00007. [Online]. Available: https://meetings.aps.org/Meeting/MAR21/Session/R33.7

[30] *JESD204B*, Texas Instruments, Inc., Dallas, TX, USA. Accessed: Apr. 27, 2021. [Online]. Available: https://www.ti.com/lit/ml/slap161/slap161.pdf

[31] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over Ethernet," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun.*, 2009, pp. 1–5, doi: 10.1109/ISPCS.2009.5340196.

[32] G. Huang *et al.*, "Qubit fast reset with QubiC," *Bull. Amer. Phys. Soc.*, vol. 65, no. 1, 2020, Art. no. C71.00233. [Online]. Available: https://meetings.aps.org/Meeting/MAR20/Session/C71.233

[33] D. C. McKay *et al.*, "Efficient Z gates for quantum computing," *Phys. Rev. A*, vol. 96, no. 2, 2017, Art. no. 022330, doi: 10.1103/PhysRevA.96.022330.

[34] J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959, doi: 10.1109/TEC.1959.5222693.

[35] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, USA, 1998, pp. 191–200, doi: 10.1145/275107.275139.

[36] *True-Q*, Quantum Benchmark, Inc., Kitchener, ON, Canada. Accessed: Dec. 21, 2020. [Online]. Available: https://trueq.quantumbenchmark.com/

[37] *Qiskit*. IBM, Armonk, NY, USA. Accessed: Dec. 21, 2020. [Online]. Available: https://qiskit.org/

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

[39] *Cirq*, Google LLC, Menlo Park, CA, USA. Accessed: Dec. 21, 2020. [Online]. Available: https://github.com/quantumlib/Cirq

[40] M. D. Reed *et al.*, "High-fidelity readout in circuit quantum electrodynamics using the Jaynes-Cummings nonlinearity," *Phys. Rev. Lett.*, vol. 105, no. 17, 2010, Art. no. 173601, doi: 10.1103/PhysRevLett.105.173601.

[41] R. P. Brent, *Algorithms for Minimization Without Derivatives*. North Chelmsford, MA, USA: Courier Corporation, 2013.

[42] S. Sheldon *et al.*, "Characterizing errors on qubit operations via iterative randomized benchmarking," *Phys. Rev. A*, vol. 93, no. 1, 2016, Art. no. 012301, doi: 10.1103/PhysRevA.93.012301.

[43] Y. Xu *et al.*, "Automatic single qubit characterization with QubiC," *Bull. Amer. Phys. Soc.*, vol. 65, no. 1, 2020, Art. no. R68.00010. [Online]. Available: https://meetings.aps.org/Meeting/MAR20/Session/R68.10

[44] M. D. Reed, "Entanglement and quantum error correction with superconducting qubits," Ph.D. dissertation, Yale Univ., New Haven, CT, USA, 2013.

[45] E. Knill *et al.*, "Randomized benchmarking of quantum gates," *Phys. Rev. A*, vol. 77, no. 1, 2008, Art. no. 012307, doi: 10.1103/PhysRevA.77.012307.

[46] S. Sheldon *et al.*, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," *Phys. Rev. A*, vol. 93, no. 6, 2016, Art. no. 060302, doi: 10.1103/PhysRevA.93.060302.

[47] Y. Xu *et al.*, "Automatic qubit characterization and gate optimization with QubiC," 2021, *arXiv:2104.10866*.

[48] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, "Simple pulses for elimination of leakage in weakly nonlinear qubits," *Phys. Rev. Lett.*, vol. 103, no. 11, 2009, Art. no. 110501, doi: 10.1103/PhysRevLett.103.110501.

[49] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and robust randomized benchmarking of quantum processes," *Phys. Rev. Lett.*, vol. 106, no. 18, 2011, Art. no. 180504, doi: 10.1103/PhysRevLett.106.180504.

[50] M. S. Blok *et al.*, "Quantum information scrambling on a superconducting qutrit processor," *Phys. Rev. X*, vol. 11, no. 2, 2021, Art. no. 021010, doi: 10.1103/PhysRevX.11.021010.

[51] C. Dankert, R. Cleve, J. Emerson, and E. Livine, "Exact and approximate unitary 2-designs and their application to fidelity estimation," *Phys. Rev. A*, vol. 80, no. 1, 2009, Art. no. 012304, doi: 10.1103/PhysRevA.80.012304.

[52] J. Emerson *et al.*, "Scalable noise estimation with random unitary operators," *J. Opt. B-Quantum Semiclassical Opt.*, vol. 7, no. 10, 2005, Art. no. S347, doi: 10.1088/1464-4266/7/10/021.

[53] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Phys. Rev. A*, vol. 94, no. 5, 2016, Art. no. 052325, doi: 10.1103/PhysRevA.94.052325.

[54] M. Ware, G. Ribeill, D. Riste, C. A. Ryan, B. Johnson, and M. P. da Silva, "Experimental demonstration of Pauli-frame randomization on a superconducting qubit," 2018, *arXiv:1803.01818*.

[55] A. Hashim *et al.*, "Randomized compiling for scalable quantum computing on a noisy superconducting quantum processor," 2020, *arXiv:2010.00215*.

Yilun Xu received the B.S. and Ph.D. degrees in engineering physics from Tsinghua University, Beijing, China, in 2013 and 2018, respectively.

Since 2018, he has been with Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His research interests include radio frequency control systems for applied physics facilities (including quantum computing, laser, and particle accelerator), and field-programmable-gate-array-based digital signal processing and feedback control.
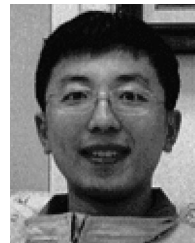
Gang Huang received the B.S. degree in engineering physics and the Ph.D. degree in accelerator technology and application from Tsinghua University, Beijing, China, in 1997 and 2002, respectively.

Since 2005, he has been with the Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His research interests include field-programmable-gate-array-based radio frequency control for quantum computing, low-level radio frequency control system for particle accelerators, femtosecond timing and synchronization, and ultrafast beam diagnostic.

Jan Balewski received the Ph.D. degree in physics from Jagiellonian University, Kraków, Poland, in 1992.

He was a Researcher with the Laboratory for Nuclear Science, Massachusetts Institute of Technology, Cambridge, MA, USA, where he worked on the DarkLight experiment at the Jefferson Lab and on polarized protons experiment STAR at the Relativistic Heavy Ion Collider. In 2016, he joined the Data Science Engagement Group, National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, to work on machine learning and recently on a variety of quantum computing projects.

Ravi Naik received the Ph.D. degree in multimode circuit quantum electrodynamics from the University of Chicago, Chicago, IL, USA, under the supervision of Prof. David Schuster, in 2018.

He is currently a Postdoctoral Researcher with the University of California at Berkeley, Berkeley, CA, USA, leading the measurement efforts at the Quantum Nanoelectronics Laboratory with Prof. Irfan Siddiqi. His current research interests include the implementation, characterization, and optimization of two-qubit gates in multiqubit superconducting circuit processors, as well as studying the methods and effectiveness of error mitigation strategies on near-term quantum algorithms.

Alexis Morvan received the B.Sc. degree in physics from the École Normale Supérieure de Lyon, Lyon, France, in 2012, the M.S. degree in quantum physics from École Normale Supérieure, Paris, France, in 2015, and the Ph.D. degree in physics from the University of Paris-Saclay, Gif-sur-Yvette, France, in 2019.

He is currently a Postdoctoral Researcher with the Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His research interests include the characterization and optimization of superconducting circuit processors for quantum computing.

Bradley Mitchell received the B.S. degree in engineering physics from the University of Colorado Boulder, Boulder, CO, USA, in 2013, and the M.Sc. degree in physics from ETH Zürich, Zürich, Switzerland, in 2016. He is currently working toward the Ph.D. degree with the University of California at Berkeley, Berkeley, CA, USA.

Kasra Nowrouzi received the B.S. degree in electrical engineering and computer sciences and the Ph.D. degree in applied science and technology from the University of California at Berkeley, Berkeley, CA, USA, in 2009 and 2019, respectively.

His research involved the development of an ultrastable high-resolution ptychographic X-ray microscope with the Advanced Light Source Synchrotron Facility, Lawrence Berkeley National Laboratory, Berkeley, where he is currently a Postdoctoral Researcher working on development and deployment of full-stack hardware for superconducting quantum computing experiments.

David I. Santiago received the B.A. degree in physics from the University of Puerto Rico, San Juan, Puerto Rico, in 1993 and the Ph.D. degree in physics from Stanford University, Stanford, CA, USA, in 2000.

His experience with Defense Advanced Research Projects Agency and Intelligence Advanced Research Projects Activity program managers as well as experience in the private field with Northrop Grumman include shaping, managing, and executing programs in microtechnology, condensed matter physics, and quantum computation. His hands-on experience with physical modeling and qubit device theory involves modeling qubit gates, qubit readout and characterization, and noise in quantum computing. In 2019, he joined the Advanced Quantum Testbed Program, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, as the Science and Technology Lead.

Irfan Siddiqi received the A.B. degree in chemistry and physics from Harvard University, Cambridge, MA, USA, in 1997, and the Ph.D. degree in applied physics from Yale University, New Haven, CT, USA, in 2002.

He is currently a Professor of physics with the University of California at Berkeley (UC Berkeley), Berkeley, CA, USA, and a Faculty Scientist with the Lawrence Berkeley National Laboratory, Berkeley. He is the Director of the Quantum Nanoelectronics Laboratory, UC Berkeley, the Advanced Quantum Testbed Program, Lawrence Berkeley National Laboratory, and the Quantum Systems Accelerator Program. His research interests include single-shot readout of superconducting quantum bits, quantum feedback, observation of single-quantum trajectories, and near-quantum limited microwave frequency amplification.

Dr. Siddiqi was a recipient of the American Physical Society George E. Valley, Jr., Prize in 2006, the UC Berkeley Distinguished Teaching Award in 2016, and the Joseph F. Keithley Award for Advances in Measurement Science in 2021. He is a Fellow of the American Physical Society.