Check for updates

QiCells: A Modular RFSoC-based Approach to Interface Superconducting Quantum Bits

RICHARD GEBAUER, NICK KARCHER, MEHMED GÜLER, and OLIVER SANDER, Institute for Data Processing and Electronics, Karlsruhe Institute of Technology, Germany

Quantum computers will be a revolutionary extension of the heterogeneous computing world. They consist of many quantum bits (qubits) and require a careful design of the interface between the classical computer architecture and the quantum processor. For example, even single nanosecond variations of the interaction may have an influence on the quantum state. Designing a tailored interface electronics is therefore a major challenge, both in terms of signal integrity with respect to single channels, as well as the scaling of the signal count.

We developed such an interface electronics, an RFSoC-based qubit control system called QiController. In this paper, we present the modular FPGA firmware design of our system. It features so-called digital unit cells, or QiCells. Each cell contains all the logic necessary to interact with a single superconducting qubit, including a custom-built RISC-V-based sequencer. Synchronization and data exchange between the cells is facilitated using a special star-point structure. Versatile routing and frequency-division multiplexing of generated signals between QiCells and converters are also supported. High-level programmability is provided using a custom Python-based description language and an associated compiler. We furthermore provide the resource utilization of our design and demonstrate its correct operation using an actual superconducting five qubit chip.

CCS Concepts: • **Hardware** → **Digital signal processing**; **Quantum computation**.

Additional Key Words and Phrases: Data acquisition, FPGA, Pulse generation, Quantum bits, Quantum-classical interface, Quantum computing, RFSoC, RISC-V

1 INTRODUCTION

Quantum computing promises many applications, ranging from quantum simulation [7] and implications for encryption [10], over drug research [22] and material science [4], to quantum machine learning [6] and other optimization problems [3]. At the same time, quantum computers are no general purpose machines and are best used as accelerators in a heterogeneous computing cluster. To build a useful quantum computer, many quantum bits (qubits) are required to form a quantum processor [28]. Such quantum processors are not able to execute programs autonomously like regular processors. They require external control, implemented in classical architectures and connected to them via a so called quantum-classical interface. This interface also handles the data exchange with the quantum processor. It thus plays a central role in the architecture of a quantum computer.

The requirements for such an interface strongly depend on the specific qubit realization. This work focuses on superconducting circuits like Transmon qubits [15], one of the most common realizations. To interface with them, microwave pulses with arbitrary shape and frequencies of a few gigahertz have to be generated and acquired.

Authors' address: Richard Gebauer, richard.gebauer@kit.edu; Nick Karcher; Mehmed Güler; Oliver Sander, oliver.sander@kit.edu, Institute for Data Processing and Electronics, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, Eggenstein-Leopoldshafen, Baden-Wurttemberg, Germany, 76344.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

1936-7406/2022/12-ART $15.00
https://doi.org/10.1145/3571820

Just Accepted

Timing of these pulses is crucial with nanosecond accuracy necessary to obtain reproducible results. While the precise requirements differ depending on setup and experiment, a sampling and timing accuracy of one nanosecond or less will mostly be sufficient. Qubit state measurements involve microwave signal recording and demodulation. Multi-qubit control pulses are either realized by flux pulses or also using separate microwave signals. Performing computational tasks requires executing well-defined sequences of single- and multi-qubit control pulses and state measurements. These pulses and measurements act as individual gate operations. Multiple such operations are concatenated to perform a quantum algorithm [16]. For the pulses, not only their duration and shape matter but also their exact frequency, amplitude and phase. The delays between multiple pulses can also have a significant impact on the results. The same applies for pulses on multiple qubits, which might need to be synchronized. For some operations, like quantum error correction, it is furthermore necessary to perform conditional pulses depending on the result of a previous state measurement [8, 26]. With single gate operations on the order of tens to hundreds of nanoseconds, response latencies should also be on the same order of magnitude.

Full-scale quantum processors, large enough to perform quantum error correction and solve relevant problems, do not yet exist. Furthermore, their design is still considered to be fundamental research in physics [2, 27]. While scaling up is one research direction, scientists also strive to improve the properties of single qubits, like minimizing the error rate [23, 29]. To perform experiments in both research contexts, we developed a versatile control electronics based on a heterogeneous radio-frequency system-on-chip (RFSoC) device that implements a quantum-classical interface tailored for superconducting qubits. Our platform fully satisfies the major requirements: control of pulse properties down to nanosecond precision, high-level programmability in quantum computing frameworks, modularity, scalability, and flexibility.

In this paper, we detail the design implemented within the programmable logic (PL) of this system. To provide scalable control and readout, the design is structured in a similar way as the quantum processor. A digital unit cell contains all necessary capabilities to interact with one qubit. This building block is then instantiated multiple times to provide individual control over up to 10 qubits with a single RFSoC. A custom cell coordinator ensures synchronization and communication between the cells to extend their usage to multi-qubit operations. A user-configurable cell signal router flexibly connects the cells to the available analog signal channels.

2 RELATED WORK

In the field of research of superconducting qubits, general-purpose laboratory equipment is widely used to generate and analyze microwave pulses. With increasing system complexity, utilizing these devices becomes unfeasible due to significant communication delays, bad scaling properties and high relative cost. Therefore, FPGA-based systems emerged being individually designed for certain experiments to meet the high data processing and latency demands, e.g. [1, 8, 20, 26].

Recently, commercial products have appeared on the market that specifically target superconducting qubits. Noteworthy products are the OPX of Quantum Machines [25], the Quantum Computing Control System (QCCS) of Zurich Instruments [30], and the Quantum Engineering Toolkit (QET) of Keysight [14]. All of these systems offer the sequencing, generation and detection of base-band microwave pulses on one or multiple FPGAs. Based on their data sheet [14], Keysight's QET uses different hardware modules for signal generation and digitization that are combined in a PXIe chassis. Zurich Instruments' QCCS also distributes the necessary capabilities over different devices suggesting higher latencies than if integrated on a single SoC. None of these two systems uses a comparable structure to the digital unit cell proposed in this paper. Signal generation and acquisition are distributed over physically separated devices. In contrast, Quantum Machines' OPX integrates them on a single device. They utilize "pulsers" that encapsulate the functionality for a single qubit, similar to the digital unit cell in this paper. A more detailed comparison is impossible as the internal technical details are not publicly available for these products.

Just Accepted

The same holds true for commercial quantum computing systems, as available from IBM [11] and Google [2]. Even though some information about their control electronics systems is disclosed in publications and on their websites, to the best of our knowledge, it is not detailed enough to perform a meaningful comparison.

We intend to close this gap in the scientific community by making our architecture publicly available. In our opinion, the interface between the quantum processor and the classical computing domain is and will be an essential part of quantum computers. Therefore, we believe it is essential to have a scientific discussion about this interface, as well as a framework which enables exploiting different techniques to find the optimum approach for qubit control and readout. This is only possible if the architecture of such a system is public knowledge and adaptations and verifications can be made by the community.

3 INTERFACING SUPERCONDUCTING QUANTUM BITS

Qubits are the building blocks of a quantum processor. Similar to a classical bit, they have two fundamental states, labeled $|0\rangle$ and $|1\rangle$. Yet, they can also stay in an arbitrary superposition $|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$. The state of the qubit can then be depicted as a point on the surface of a sphere, called Bloch sphere, with the states $|0\rangle$ and $|1\rangle$ being located at the north and south pole, respectively. In the typical frame, they are located on the Z axis while the X and Y axis span the plane of the equator. Accordingly, operations on the qubit are rotations around the surface of the sphere.

Superconducting qubits are microscopic, non-linear resonance circuits fabricated with similar methods as used in the semiconductor industry. They are made from superconducting material and exhibit quantum behavior when cooled below the transition temperature where the material completely drops its electrical resistance. This can be employed to engineer systems showing two fundamental quantum states used as computational subspace. Due to the low temperature requirement of typically tens of millikelvin, the chip with the quantum bits is located inside a cryostat.

To control a qubit's state, microwave pulses with frequencies of a few gigahertz and nanosecond time resolution are used. Each qubit has a dedicated transition frequency f_{01} corresponding to the energy difference of the two basis quantum states. When irradiated with a microwave pulse at this frequency, the state will oscillate around the Bloch sphere between $|0\rangle$ and $|1\rangle$, called Rabi oscillation. By adjusting either the duration or the amplitude of the pulse, the rotation angle around the Bloch sphere can be varied. Changing the phase of the microwave pulse will change the axis of rotation in the equatorial plane of the sphere. As the phase determines the frame of reference for the sphere, one can also adjust the global phase to perform a virtual Z rotation around the equator. As result, one can perform arbitrary rotations.

Depending on the actual superconducting circuits, additional current or flux pulses might be required to perform special single or two-qubit gates. Two-qubit gates might be realized by simultaneously applying special trapezoid DC pulses to both qubits on separate microwave channels [18]. In other quantum processor architectures, tunable couplers are integrated between the qubits that need to be switched on and off to realize a two-qubit gate [19]. As, in typical architectures for such processors, the qubits are arranged as a two-dimensional lattice, this leads to up to two tunable couplers that need to be controlled per qubit.

The readout of a qubit is commonly performed as dispersive readout where an additional microwave resonator is coupled to the qubit. Depending on the qubit state, the resonator will experience a slight shift in its resonance frequency f_r. By probing the resonator with a microwave pulse near this frequency, the qubit state will be encoded in the amplitude and phase response. Due to the low temperature requirement, the chip needs to be shielded as best as possible from the surrounding. Thus, the probe pulse will be dampened inside the cryostat to thermalize the noise present on the signal. After interacting with the resonator with a strength of only a few photons, it needs to be amplified again to be detectable by the room-temperature electronics. Special low-noise amplifiers are required to obtain a signal-to-noise ratio (SNR) good enough to extract the qubit state from a single

Just Accepted

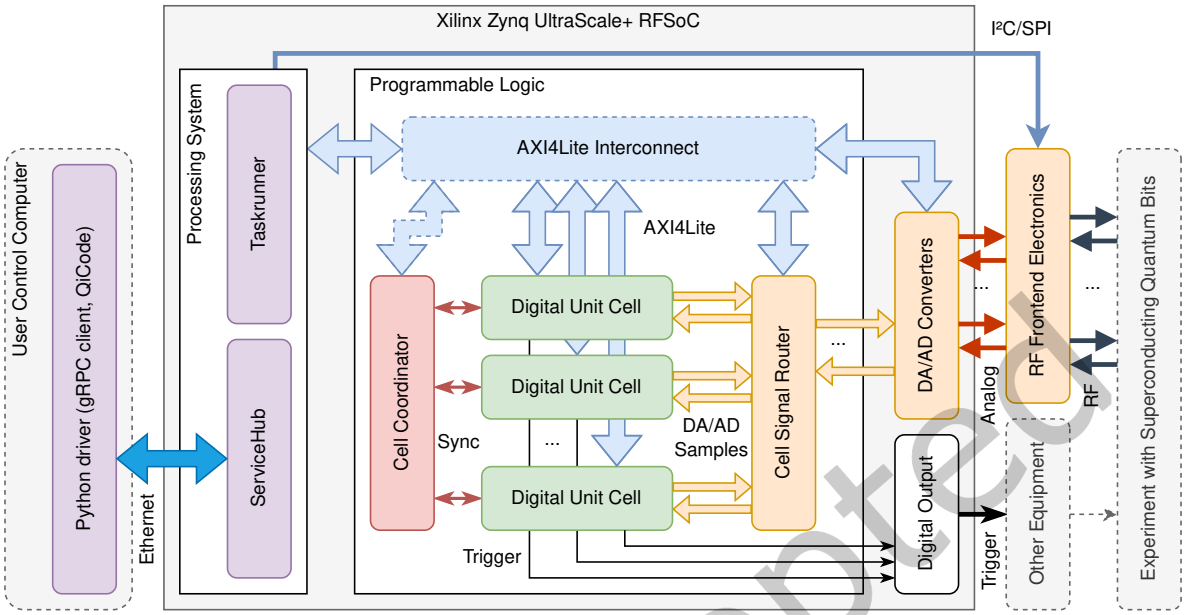

Fig. 1. Architecture of our heterogeneous electronics platform.

Just Accepted

measurement. If such an amplifier is not available, multiple repetitions can still be performed and the results averaged to increase the SNR. However, in this case, the qubit state from individual measurements cannot be determined.

More detailed information on superconducting qubits can e.g. be found in [16].

4 SYSTEM ARCHITECTURE

The system we developed aims to facilitate these control and readout mechanisms to provide a quantum-classical interface for superconducting qubits (see Fig. 1). It is based on the heterogeneous architecture of a Xilinx Zynq UltraScale+ RFSoC. The chip incorporates an FPGA, a quad-core ARM Cortex-A53 application processor (APU), a dual-core ARM Cortex-R5 real-time co-processor (RPU), as well as eight multi-gigasample AD and DA converters (ADCs/DACs). The converters operate at 4 GHz sampling frequency and handle the microwave generation and digitization in the complex base-band. The converter clock, from which all relevant other clocks are derived, can be locked onto an external, highly stable clock reference to prevent temperature-dependent frequency drifts. We utilize decimation and interpolation filters to obtain a per-channel data rate of 1 GSPS within the programmable logic (PL). The signals are digitally represented as in-phase and quadrature (I/Q) components, each with 16 bit resolution. Hence, two converter channels per signal input and output are required to handle both quadratures representing the complex-valued base-band. A separate radio-frequency (RF) frontend electronics with I/Q mixers and microwave sources translates these base-band signals from and to the frequency range of superconducting circuits, typically in the order of 4 to 10 GHz.

The APU hosts a Yocto-based Linux operating system which initializes the platform at boot time. Then, it starts the modular ServiceHub framework [13] to provide means for communication and external configuration. For each type of PL module, a ServiceHub plugin exists facilitating user access to these modules. The RF frontend electronics can also be controlled by a ServiceHub plugin via SPI and I^2C. The communication to the user is based on remote procedure calls (RPC) and utilizes the open-source framework gRPC [5]. Thus, the client can be written in any language that is supported by gRPC. As many physics laboratories use Python, we provide a Python client for our platform. It integrates with the open-source quantum measurement suite Qkit [24] and the open-source quantum development kit Qiskit [12]. The RPU hosts the Taskrunner framework [9] which provides convenient access to the real-time processor. It complements the PL with versatile, low-latency real-time control, data aggregation and evaluation features. Both processors communicate with the PL using a register-based AXI4Lite bus where the modules are mapped into the physical memory address range of the PS. Access is performed by simple memory read and write operations using the AXI HPM FPD interface.

The user design in the PL is written in VHDL and operates on a single clock domain of 250 MHz directly derived from the converter clock. This avoids clock domain crossings and guarantees deterministic timing and nanosecond accuracy, which is crucial for the control of superconducting qubits. The digital unit cells provide the main functionality within the PL. Each unit cell contains all the necessary modules to control and read out a single qubit, as well as to perform multi-qubit interactions, as presented in the following section. By implementing multiple such cells, multiple qubits can be controlled by a single system. To ensure synchronicity between the cells and facilitate inter-cell-communication, a special cell coordinator is connected to each of these cells which can start any subset of them simultaneously. It is also responsible to orchestrate data exchange between the cells and flexibly re-synchronize them during the execution. The digital microwave signals generated in the unit cells are routed via AXI-streams to the DACs. In the simplest case, each digital unit cell is connected to separate converter channels for readout and control pulses. To reduce the channel count, multiple signals can also be frequency-division multiplexed and combined (added up) onto one output in the cell signal router. Likewise, the returning digitized microwave signals from the ADCs can be distributed and split up onto the belonging cells. This configuration can be dynamically configured using the AXI4Lite register interface of the module. Besides the microwave pulses, also digital trigger signals can be generated. These can e.g. be used to trigger additional laboratory equipment for special experiments.

As this builds up to quite a complex system with a lot of dependencies, directly configuring all the modules is inconvenient and error-prone. Instead, we provide a high-level experiment description language based on Python, dubbed QiCode, that can be used to functionally describe the cell's control flow and output. Descriptions in QiCode will be automatically compiled into RISC-V instructions for the sequencers (see the following section) and the configuration parameters for the other modules within the different digital unit cells. More information concerning the high-level programmability is covered in Section 7. Furthermore, an appropriate task to fetch the data from the PL and transfer it to the user is loaded onto the Taskrunner. The remaining configuration set is loaded into the PL modules. Then, the user starts the execution of the task in the Taskrunner which, in turn, will simultaneously start the sequencers in the relevant digital unit cells using the cell coordinator. After all sequencers have finished their execution, the Taskrunner fetches the data from the data storages. This ensures that the limited memory space on the FPGA does not become a limiting factor. For special experiments with longer run-time or higher data rates, the Taskrunner can fetch data already during the execution to prevent an overflow within the FPGA memory. Depending on the configuration, the Taskrunner can also perform multiple repetitions and accumulate or average the resulting data. Finally, it transfers the data back to the user.

Just Accepted

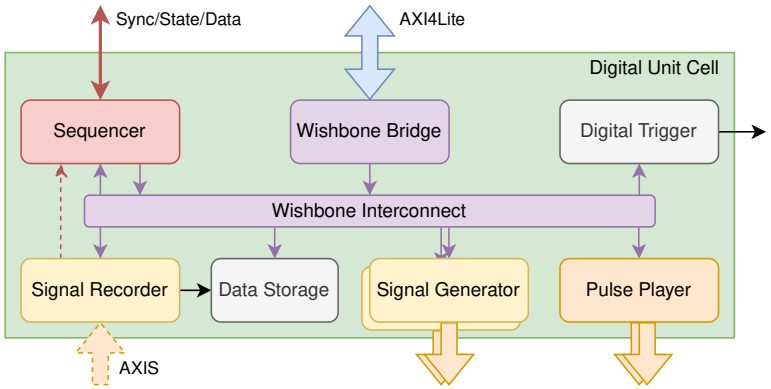

Fig. 2. Architecture of a single digital unit cell. Arrows of the WB bus indicate the direction from master to slave.

5 DIGITAL UNIT CELL

The core of our PL design is the digital unit cell, dubbed QiCell, which will be described in detail in this section. It generates and analyzes microwave pulses in the complex-valued base-band for a single qubit. The cells are also capable to generate additional flux pulses for multi-qubit coupling and to control peripheral devices for additional capabilities. We chose to implement the entire module on the PL because qubit control requires at least timing precision on the granularity of single clock cycles. Real-time capabilities of the RPU could be utilized but are impacted by latency and interferences on the AXI communication infrastructure of the PS.

5.1 Module Architecture

The architecture of the digital unit cell is depicted in Fig. 2. Most internal communication inside the cell is handled by a Wishbone (WB) bus [21]. WB was selected because of its simplicity and resource efficiency compared to a full featured AXI interface. We use a custom implementation to ensure deterministic timing, which is essential for our application. Because the same interface is used by the PS to configure the cell, we utilize an AXI4Lite to WB bridge to translate the register accesses for the internal bus. Two signal generators create the required pulses to control and read out the qubits utilizing an AXI-stream (AXIS) interface. A signal recorder takes the digitized signal from the ADCs and demodulates it to obtain the qubit state. A dedicated data storage can collect the resulting data from the signal recorder and the sequencer. A separate pulse player generates flux pulses required to realize multi-qubit interactions. A digital trigger block allows the user to generate digital signals to address and trigger external lab equipment. All modules are controlled and activated by the sequencer which orchestrates their execution in single-cycle steps (4 ns). The signal recorder directly reports all measured qubit states back to the sequencer which can then perform a fast conditional response. In all other cases, the modules communicate exclusively via the WB bus.

5.2 Communication Infrastructure

The Wishbone bus inside the digital unit cell features a 16 bit address width and a 32 bit data width. A custom WB interconnect allows for two masters and up to seven connected slaves. Both sequencer and WB bridge are connected as masters. In case of a conflicting access, the sequencer always takes priority in order to keep deterministic timing during executions. The WB bridge also performs an address translation from byte-based addressing as used by the AXI4Lite bus to register-based addressing used by WB. All slave modules have a special

Just Accepted

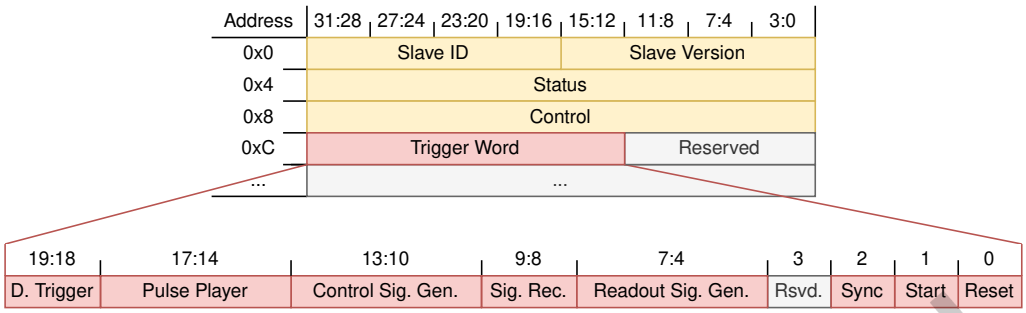

Fig. 3. Common start of the register interfaces of all modules inside the digital unit cell. The address offset is given in bytes.

WB register interface implemented that guarantees a deterministic response time of 2 cycles without stalling. With the interconnect, a read operation takes exactly 4 cycles from the sequencer to return a result. Access from the WB bridge will take one extra cycle and might be stalled if the sequencer is currently accessing the bus. With the deterministic latency in mind, we modified the interconnect to allow a single pipelined register access each cycle on the bus even if a previous operation originating from one of the masters has not finished yet. This way, we can ensure that the sequencer can always issue trigger commands on the WB bus with deterministic access latency.

The multiplexing between the modules happens according to the highest three address bits. While `000` up to `110` represent the according connected module, 111 acts as broadcast modifier. In this case, the bus operation will be forwarded to all connected slave modules at the same time. We utilize this feature to provide means to trigger all the connected modules with a common trigger word at the same time without utilizing a separate trigger infrastructure. The register interfaces of all slave modules are therefore starting in a similar way with an info, a status and a control register (see Fig. 3). Afterwards, a broadcast register follows with a 20 bit trigger word field that can be strobed by a write access. The remaining registers can be freely and independently used depending on the demand of the modules. Special trigger commands are shared between all modules to reset them, mark the start of an execution, and to synchronize the NCOs inside the two signal generators and the signal recorder.

5.3 Sequencer

The sequencer is the core of the digital unit cell. It controls all connected slave modules and can e.g. schedule pulses or start a recording. The user can define a sequence of operations in 4 ns steps using the RISC-V instruction set architecture (ISA). We chose the RISC-V ISA as it is state of the art, easily extensible, very flexible, hardware efficient, provides a rich ecosystem, and is well established in the scientific community. From the modular instruction sets of the RISC-V ISA, we implemented most of the base integer and multiplication set, as well as a custom special-purpose set for the sequencing. In total, 36 instructions are available for the sequencer as well as 32 registers. The following operations are part of the special-purpose set:

`TRIG`: Writes the given trigger word to the broadcast register of all connected WB slave modules.
`WAIT-IMM`: Delays the execution by the given number of clock cycles.
`WAIT-REG`: Delays the execution by the number of clock cycles given in the defined register.
`WAIT-REG-TRIG`: Same as `WAIT-REG` but reduces the wait time given in the register by one cycle. This can e.g. be used after a `TRIG` command to wait a register-defined time but include in it the duration of the previous command.

Just Accepted

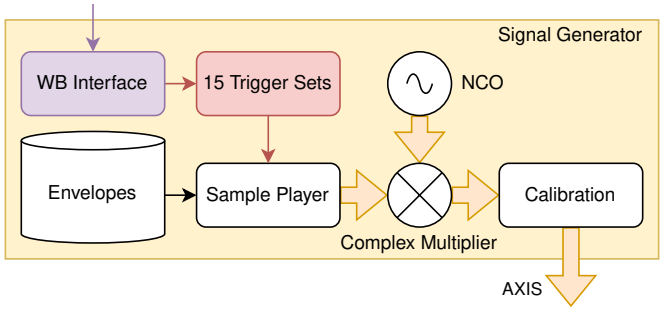

Fig. 4. Structure of the signal generator inside the digital unit cell.

`SYNC-STATE`: Waits for a new qubit state result before continuing with the program execution. The state can originate from the same or another cell and will be transferred via the cell coordinator.
`SYNC-START`: Ends the execution of the sequencer and returns to an idle state waiting for a new start command.
`CELL-SYNC`: Initiates a barrier synchronization of the cell with other given cells via the cell coordinator.
`CELL-DATA-SEND`: Sends the data of the given register to one or multiple other cells via the cell coordinator.
`CELL-DATA-RECV`: Receives data from a specified other cell via the cell coordinator and stores it in the given register.

Most instructions are optimized to execute in only one cycle for highest performance. Only multiplication takes 6 cycles in order to relax the timing requirements for the 32 bit times 32 bit multiplication. Instructions entailing a jump in the program counter (branch instructions if the comparison yields true, as well as the unconditional `JAL` jump operation) take 3 cycles. The different wait operations take as long as specified in the command. The sync and data exchange commands might wait an undetermined time on external input. Currently, up to 1024 instructions can be stored inside a single BRAM. Typical experiments require tens of instructions to be executed. It is therefore enough for nearly all imaginable experiments but can also be easily extended by enlarging the BRAM, if necessary.

The sequencer has both a WB master and slave interface. As for every other module, the slave interface is used to configure and control the sequencer. The master interface can reconfigure the connected slave modules or fetch data from them. The according load and store operations will take 8 cycles as they wait for the response of the WB bus. 4 cycles account for the deterministic latency of the bus and 4 for processing the operation in the sequencer, applying the output to the bus and processing the return signals. Trigger commands are applied as pipelined block write operations as it is essential that they are issued each cycle without stalling. The sequencer does not wait for the bus return when applying trigger commands. Therefore, after finishing the block write at the sequencer, the bus might still be processing while the sequencer already executes the next operation. If this is a normal load or store operation, it will be delayed as the sequencer first has to wait until the bus is not busy anymore. This could be further optimized by always applying pipelined block write operations and exploiting the deterministic latency of our modified WB bus.

5.4 Signal Generator

Each digital unit cell has two signal generators, one for readout pulses and one for control pulses. It contains 15 trigger sets that can be selected by a 4 bit trigger command within the trigger word. Each trigger set represents a certain pulse that can be played by the signal generator. The following properties can be individually set for each trigger set:

Just Accepted

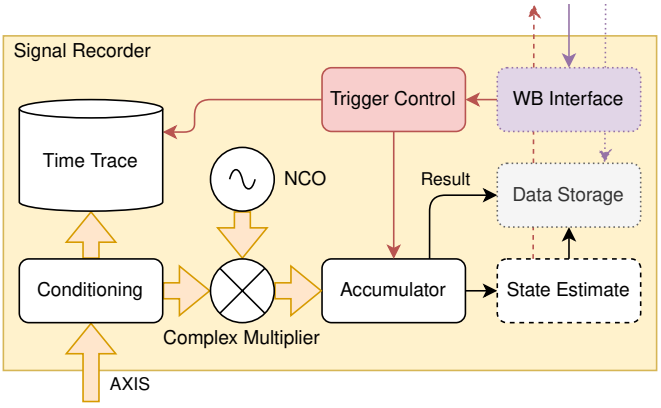

Fig. 5. Structure of the signal recorder inside the digital unit cell.

- The duration of the pulse in cycles.
- The phase offset of the pulse relative to a global phase reference inside the module.
- A scaling factor to change the amplitude of the pulse.
- Address offsets for I and Q envelopes inside the envelope memory. These can also point to the same address.
- Option to neglect the Q envelope, saving half the memory if the envelope values are real.
- Option to hold the last envelope value until another trigger is received. This is used for continuous wave operation and variable length pulse shapes like a trapezoid.
- Option to persist the phase offset in the global phase reference, leading to a virtual Z rotation.

Additionally, the module has a common frequency reference which can be configured. It is implemented as a numerically controlled oscillator (NCO). Furthermore, an output calibration is possible to adjust the I and Q output amplitude.

The structure of the signal generator is presented in Fig. 4. All configuration as well as the trigger is fed to the module via its WB interface. When a trigger arrives and selects one of the 15 trigger sets (trigger value 0 is reserved for no operation), the configuration of this trigger set is loaded into the module and the execution is started. The sample player fetches the according I and Q envelope sample values from the envelope memory. It is 8 kB large and can therefore store 4096 real-valued samples (16 bit each) corresponding to up to about 4 µs of pulse data. As most pulses are in the order of tens to hundreds of nanoseconds, this is enough for most applications. The pulse will then be output by the sample player and fed into a complex multiplier. There, the envelope will be multiplied by the oscillating complex quadrature signal of the NCO to obtain the digital pulse in the base-band. Afterwards, the I and Q quadratures can be independently calibrated and the signal leaves the module as AXI-stream.

5.5 Signal Recorder

The signal recorder obtains the digitized microwave signals from the converters and performs a digital down-conversion (DDC). Its structure is depicted in Fig. 5. Due to mixer and other imperfections in the analog setup, the raw I and Q data from the converters might not be balanced or have a 90° phase relation. To correct for this, a matrix multiplication will be performed on the raw input data:

$$\begin{pmatrix} I_{\text{out}} \\ Q_{\text{out}} \end{pmatrix} = M_{\text{cond}} \left[\begin{pmatrix} I_{\text{in}} \\ Q_{\text{in}} \end{pmatrix} - \begin{pmatrix} I_{\text{offset}} \\ Q_{\text{offset}} \end{pmatrix} \right] \tag{1}$$

Just Accepted

Besides the 2×2 matrix M_{cond} to correct for amplitude and phase distortions, a DC offset can also be subtracted. The corrected raw time trace is stored inside a BRAM. It can later be used for debugging purposes or to visualize the raw input that was demodulated in the following. The signal will then be down-converted by a complex multiplier where it is multiplied with a reference oscillation having the negative frequency of the base-band carrier. It thereby shifts the frequency of the signal carrier to DC. Afterwards, a low-pass filter and decimation are necessary to average the resulting I and Q components which are later used to determine the amplitude and phase response. In our case, we implement a boxcar integrator by using a simple accumulator to add up the samples over an adjustable time window. Alternatively, e.g. an FIR filter and decimation could be used for a better low-pass characteristic. At the same time, the accumulation yields a smaller latency to obtain a result.

While conditioning and complex multiplication are performed continuously, the boxcar integration as well as the storage of the raw time trace are only activated when the signal recorder receives a trigger signal. As the readout pulse experiences an electrical delay to the quantum chip and back, a trigger offset can be defined. Only after this offset time has passed, the trigger will be executed by the module. This way, the sequencer can trigger the readout signal generator and recorder at the same time and does not have to account for the electrical delay itself. Once the recording duration has passed, the accumulated result value is passed to the data storage, and used to estimate the qubit state.

For this estimation, the result is transformed into a binary information of `0` or `1` corresponding to the two possible qubit states that can be measured. Via the cell coordinator, this state result will be directly returned to the sequencer which can be programmed to wait for this value and store it in a register using the `SYNC-STATE` operation. The state will also be passed to the data storage where it can be aggregated and saved for later retrieval. As the data storage is tightly linked to the signal recorder, it is also shown in Fig. 5. For simple experiments, the signal recorder also provides an averaging functionality where obtained results will be summed up until the module is reset externally. This is especially helpful if a single measurement should be performed and repeated many times to obtain an averaged I and Q result value.

Different operation modes of the signal recorder can be distinguished, based on the received 2 bit trigger value:

`SINGLE`: Performs a single measurement.

`ONESHOT`: Performs a single measurement but does not forward it to the data storage. A typical use-case are two consecutive measurements where the first one is only used internally and will result in a state estimation on which the sequencer will react. The second one is then to obtain a measurement result of the experiment.

`CONTINUOUS`: Continuously performs consecutive measurements and returns the values to the storage module. This mode can be used to obtain a seamless stream of demodulated results without the need of the sequencer to trigger each single measurement. The continuous mode will continue until another `CONTINUOUS` trigger is received. Together with the data storage and the Taskrunner, continuous operation over long periods is possible, e.g. to observe state changes of the qubit over time (so-called quantum jumps).

5.6 Data Storage

After demodulating the measurement results, the data needs to be persisted for later retrieval by the user. The data storage handles this in a configurable and flexible way. Its structure is shown in Fig. 6. The module contains four separated dual-port BRAMs to store values. These can be filled individually and in parallel. Thereby, result values can be partitioned in a user-defined way, e.g. to store both qubit states and I/Q results, or to store additional information from the sequencer in a separate BRAM. These memories provide an interface to consecutively append 32 bit values to the memory until it is full. It furthermore contains an option to use the memory as a circular buffer and wrap the address instead of rising an overflow flag if the memory is full. The second port of the BRAM is mapped into the WB interface for direct read and write access from sequencer and PS.

Just Accepted

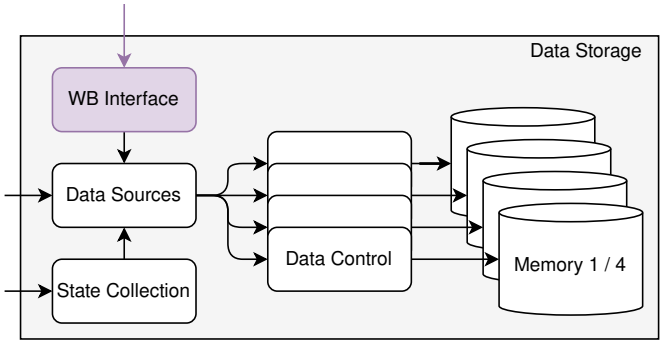

Fig. 6. Structure of the data storage inside the digital unit cell.

The signal recorder passes the single results and the estimated qubit states to the data storage. There, the qubit states will be concatenated to obtain 32 bit words containing multiple of them. Depending on the estimation routine, one can either store 32 states if only one bit is used, or 10 states if also higher states are accounted for (3 bit information per state). Summarizing, the following data can be selected and stored inside the individual memories:

- Single I and Q result values
- Single estimated qubit states
- Concatenated qubit states (either 10 or 32 per register)
- Input data from the WB interface

The last one is a special register in the WB interface to which the sequencer can write to append values to the memory blocks. This way, the sequencer can also perform calculations and store them or persist some additional values. The sequencer can also use the second port of the BRAMs mapped into the WB interface to have a memory extension, e.g. for arrays. Each memory block has a dedicated data control that decides which data source will be assigned to it. It also realizes the append and circular buffer logic as described above and provides status signals for the user, like empty, full, and overflow flags, as well as the current data size.

5.7 Pulse Player

To support multi-qubit operations, each digital unit cell contains an additional pulse player module. This is essentially an arbitrary waveform generator that can output pre-defined pulses on separate DAC channels. In contrast to the signal generator module, these pulses are not modulated using an NCO signal but directly synthesized as written to the internal memory. This saves resources and is enough for most applications where, typically, either trapezoidal flux pulses are required or the phase relation between consecutive microwave pulses is not relevant.

The structure of the pulse player module is presented in Fig. 7. It is separated in two channels that are both addressed using one common WB interface. The 4 bit trigger command is therefore split up in two 2 bit trigger values for three different trigger sets each (as, again, zero is reserved for no operation). The remaining configuration of the module and the structure of the two channels is similar to the signal generator as presented in Section 5.4.

Just Accepted

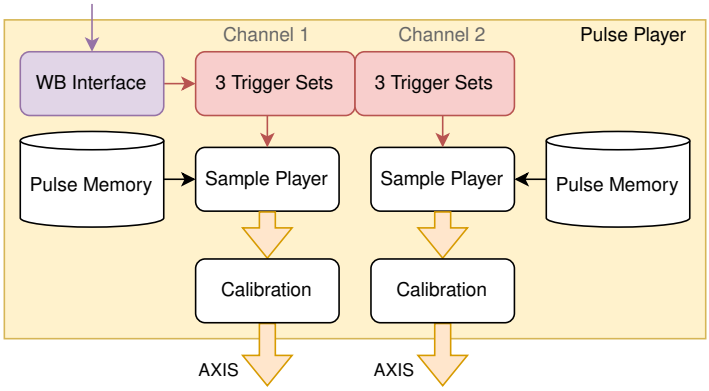

Fig. 7. Structure of the pulse player inside the digital unit cell.

5.8 Digital Trigger

While the system covers most aspects to control and readout superconducting qubits, for some experiments it might be necessary to digitally trigger external measurement equipment for additional functionality. The digital trigger provides 4 trigger sets which can individually define which of the 8 available digital outputs should be activated and how many cycles they should stay asserted. A special option can be used for continuous activation. Each output can be individually inverted and a trigger offset specified. This is especially important to synchronize the action of external devices with the operation of the system.

6 MULTI-CELL INTERACTION

While digital unit cells are sufficient to interact with individual qubits, experiments and quantum algorithms require the accurate orchestration of operations on multiple qubits. This section therefore describes how the digital unit cells are connected to facilitate synchronization and data exchange in a deterministic way.

6.1 Cell Coordinator

The cell coordinator is a star-point structure that has a connection to each digital unit cell individually. It is dimensioned for up to 16 digital unit cells. For different functionalities, different signals exist, as can be seen in Fig. 8. The supported functionalities are described briefly in the following paragraphs. By exploiting a star-point architecture, the placement of the qubits on the chip and the interactions between each other can be flexibly mapped onto the available unit cells without special constraints.

Busy Aggregation Logic. Each digital unit cell reports its busy status to the cell coordinator, i.e. if it is currently performing an operation or not. The cell coordinator aggregates these to a single logic value indicating if any of the cells is busy. It also exposes all individual busy signals via the AXI4Lite register interface, so the user can easily query the status of all cells at once.

Start Trigger Logic. A priori, the sequencers within the digital unit cells operate independent of each other. However, it is important that one has the ability to synchronously start all or a subset of them at the beginning of an execution. Therefore, for each of the cells, the cell coordinator has a single logic signal with which the sequencer of this cell can be triggered. Using the AXI4Lite register interface, the user can strobe these signals synchronously by writing the appropriate bit mask into a register which is mapped to these signals. As the

Just Accepted

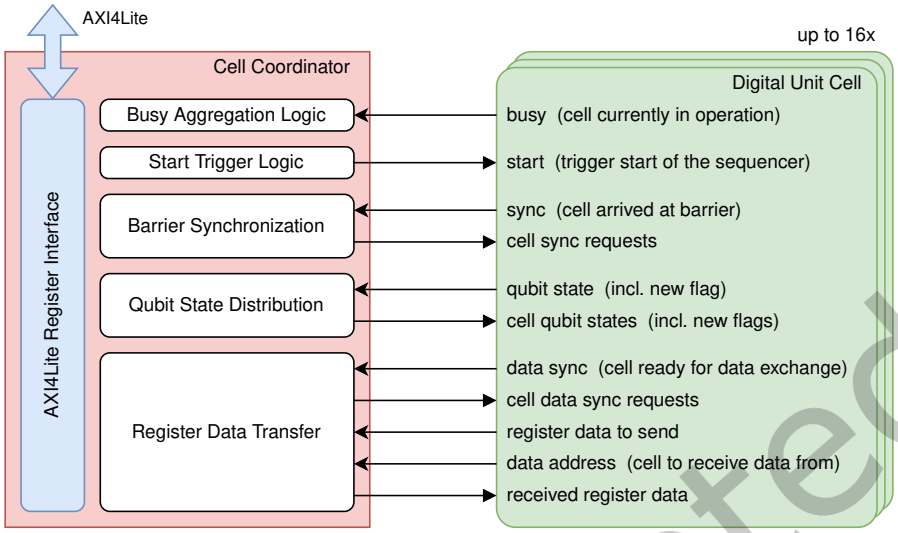

Fig. 8. Structure of the cell coordinator and its interface to the digital unit cells.

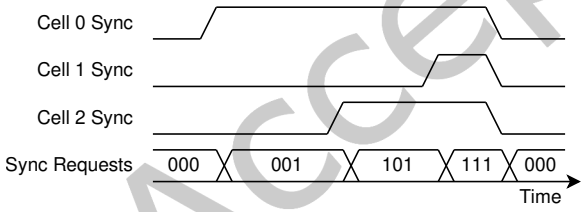

Fig. 9. Signals involved in a barrier synchronization via the cell coordinator using three cells.

connection to all the cells is identical and the cells themselves are also identical, this will lead to a synchronous start of all triggered cells.

Barrier Synchronization. Due to different operations, e.g. waiting for an external impulse or having a conditional branch depending on a measured qubit state, the initial synchronization between the digital unit cells can be lost. To re-synchronize an arbitrary subset of the cells, the sequencers and the cell coordinator implement a synchronization mechanism using a barrier. The mechanism is implemented by a sync flag with which the sequencers can tell that they have arrived at the synchronization barrier. The cell coordinator bundles these signals into a logic vector and distributes it to all connected digital unit cells. There, the sequencers wait until all relevant cells, including themselves, have arrived at the barrier. As the distribution happens via the cell coordinator, all cells will receive this state at the same time and can then continue synchronously. The concept is also exemplary visualized in Fig. 9 with three cells. Using the `CELL-SYNC` instruction, a synchronization is initialized for a cell. The instruction also holds a bit mask for all cells indicating if this cell is contributing to the synchronization barrier or not. This makes it possible to only synchronize a subset of the cells while others can continue autonomously. By issuing the same instruction word to all contributing sequencers, these will be synchronized. Once the last cell arrives at the synchronization barrier, a latency of three cycles is required before

Just Accepted

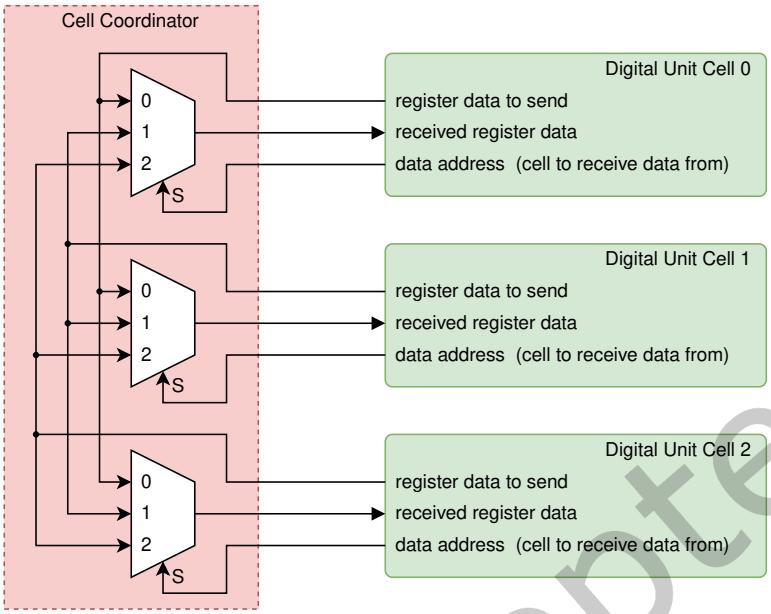

Fig. 10. Schematic implementation of the register data transfer via the cell coordinator depicted for three cells.

the next command after the barrier is executed. This is due to the processing of the synchronization instruction in the last sequencer and the synchronous propagation of its sync command through the cell coordinator.

Qubit State Distribution. The FPGA-based signal generation and processing makes it possible to respond to a previously measured qubit state with a latency of only a few hundreds of nanoseconds. To be able to not only react on the state recorded by the same cell, the measured qubit states need to be distributed among all cells. To facilitate this in a low-latent and elegant way, the cell coordinator has a special qubit state distribution logic. Each digital unit cell directly reports new states to the cell coordinator. There, they are aggregated in a logic vector containing the last states of all cells, i.e. qubits. This vector is then distributed back among all cells. Additionally, the same happens with a strobed flag indicating that a qubit state has been updated. The sequencer has a special instruction, `SYNC-STATE`, to wait for a new qubit state to arrive in this logic vector. In the instruction, one specifies both the cell index from which the qubit state will arrive and a target register in the sequencer where the state value will be placed. Afterwards, one can e.g. perform conditional branching based on this register value.

Register Data Transfer. For more complex use-cases, it can be necessary to share the value of a sequencer register in one cell with the sequencer of one or multiple other cells. This is especially true if this value is based on external influences, like measured data that cannot be pre-calculated. For this purpose, a register data transfer protocol has been established to exchange data in a deterministic and versatile fashion. The protocol is designed in a way that the participating cells can independently arrive at their respective send/receive instruction and will be synchronized during the process. The whole process is controlled by the sequencers of the participating cells. The cell coordinator acts as a a passive structure facilitating the exchange. It contains a multiplexer for each cell where the cell can specify from which source cell it wants to receive data during the exchange. The implementation in the cell coordinator is sketched in Fig. 10. For the sender cell, the `CELL-DATA-SEND` instruction is used to perform the exchange. It specifies the source register from where to take the data and all cells that

Just Accepted

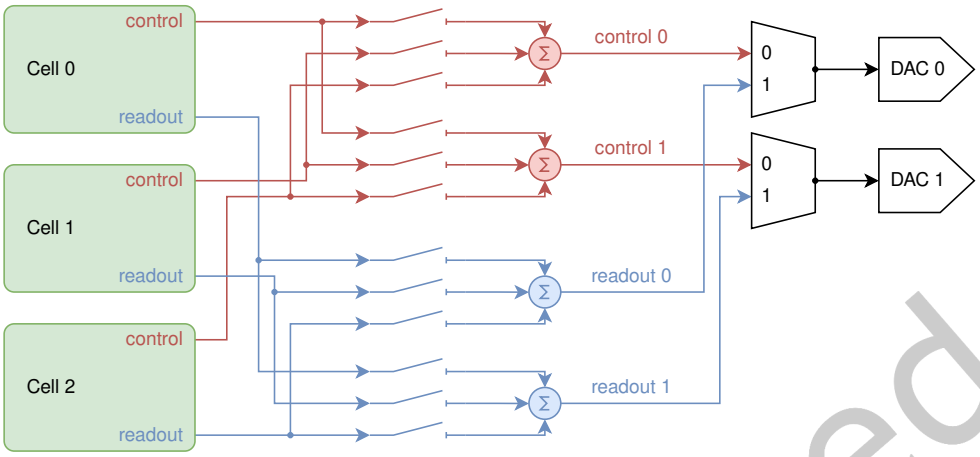

Fig. 11. Schematic structure of the flexible cell signal router depicted for three cells and two DAC channels. It only shows control and readout signals and neglects the pulse player output channels.

participate in the data exchange. The sender cell will then immediately apply the data of the register to its output towards the cell coordinator and signal that it is ready for the data transfer using the data sync flag (compare Fig. 8). Then it waits until all other cells are also ready for the data transfer, similar to the synchronization barrier. The receiving cell has the separate instruction `CELL-DATA-RECV`. It specifies the target register to store the data, as well as the cell from which to receive it. Additionally, all cells that participate in the data exchange are given. The index of the cell to receive the data from will be directly applied to the cell coordinator to switch the multiplexer input. The data sync flag will also be set indicating that the receiving cell is ready for the transfer.

Once the receiving cell finds that all participating cells, including itself, are ready for the transfer it copies the received data into the specified register and continues with the next instruction. Due to the barrier mechanism, the sender cell will reach this point at the same time and also continue with the next instruction, clearing the data output signal towards the cell coordinator. As the sync flag needs the same time to propagate through the cell coordinator as the data to send, it is guaranteed that the receiving cell stores the correct data from the sender cell. From the point when the last participating cell arrives at its data transfer instruction, the whole process is completed in three clock cycles. Due to the decentralized protocol, transfers with one sender and multiple receivers can also be realized, as well as multiple parallel data transfer operations, as long as the participating cells are disjoint.

6.2 Cell Signal Router

To build a versatile control system for superconducting qubits, different quantum processor architectures have to be supported. The interface mainly differs in the number of external microwave lines which need to be addressed and connected. Some architectures require two microwave lines per qubit, others combine all or some readout signals on one line using frequency-division multiplexing, and yet other chips do the same not only for the readout, but also for the control pulses. In order to support all these different architectures, we implemented a flexible cell signal router which can combine signals and distribute them between the digital unit cells and the converter channels. The configuration of the module can be adapted at run-time using an AXI4Lite register interface.

Just Accepted

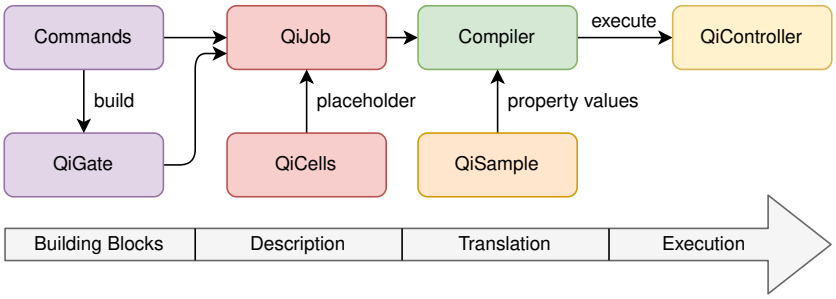

Fig. 12. Abstract concept of the QiCode experiment description language.

The structure of the cell signal router is sketched in Fig. 11. Each DAC channel, comprising two DAC outputs for I/Q operation, has two separate adder structures for readout and control signals where all signals from the cells are added together. For each adder structure the inputs from the cells can be individually muted. This way, signals can be arbitrarily multiplexed and routed to the DAC channels. Control and readout signals are kept separate to save resources and as they normally do not need to be multiplexed together. If this is a requirement, the signals can still be combined after the analog conversion. This is even beneficial as readout and control signals are usually located at different frequencies, exceeding the base-band bandwidth, and thus need to be up-converted using separate I/Q mixers before being combined. For each DAC channel, one can select if it should be used for control or readout signals. Further options of the multiplexer are the signals from the pulse players of the cells. Due to their scope of application being mainly intended for flux pulses, these are not multiplexed. On the other direction, one can decide for each digital unit cell from which ADC channel the returning signals should be forwarded.

7 HIGH-LEVEL PROGRAMMABILITY

The developed system, also called QiController, is designed as laboratory equipment specifically for the demands of superconducting qubits. To facilitate its usage, different layers of abstraction to control and configure the system exist. The ServiceHub [13] provides a first layer by encapsulating register interface accesses inside gRPC remote procedure calls. A Python client adds another layer to provide low-level configuration access in a Python-style class by wrapping the remote procedure calls. However, to provide easy and intuitive access to the capabilities of the system, another layer of abstraction is added. It is realized by a Python-based, high-level experiment description language called QiCode. It enables users to functionally and intuitively describe the control flow and output of the presented QiCells with nanosecond precision. QiCode was designed from the ground up with the needs and requirements of the superconducting qubit research community in mind, specifically targeting quantum computing experiments and algorithms. An overview of its structure and concept is shown in Fig. 12.

In QiCode, experiment descriptions are called `QiJob`. These consist of one or multiple base commands, like `Wait`, `Play`, `If/Else`, `Recording`, `Sync`, and so on. Commands can also be combined to reusable gates, so-called `QiGates`, and defined as decorated Python method. To make QiJob descriptions reusable for different qubit chips and quantum processors, all qubit related properties can be defined virtually in a `QiCells` object. Values that are often defined as placeholders here are all sorts of qubit-related characteristics, like individual frequencies and pulse lengths for different operations. The placeholders will be only filled later during the compilation stage as provided by a separate `QiSample` object. This sample object holds the actual physics parameters to interact with the connected superconducting qubit chip. These are at least the ones previously referred to using the `QiCells` object inside the `QiJob`. The sample can also be exported to and imported from JSON, so different setups can be

Just Accepted

easily persisted and loaded. An example experiment description illustrating the high-level programmability of the system is provided in Listing 1. It describes the measurement of the energy relaxation time T_1 of five qubits simultaneously and also includes the compilation and execution step.

Listing 1. Experiment description of T_1 experiment which is performed on 5 qubits simultaneously.

```
# Define some gates to reuse for the experiment descriptions
@QiGate
def PiPulse(cell: QiCell, phase: float=0.0):
    """Output manipulation pulse to rotate qubit state around Bloch sphere by pi."""
    Play(cell, QiPulse(length=cell["pi"], phase=phase, frequency=cell["f_01"]))
    # cell["pi"] is a placeholder for the given qubit's pi pulse duration
@QiGate
def Measurement(cell: QiCell, save_to):
    """Output readout pulse and trigger signal recorder to process result."""
    PlayReadout(cell, QiPulse(cell["rec_pulse"], frequency=cell["f_res"]))
    Recording(cell, cell["rec_length"], cell["rec_offset"], save_to=save_to)
@QiGate
def Thermalize(cell: QiCell):
    """Wait 5x the qubit T1 time so it can return to the (thermal) ground state."""
    Wait(cell, 5 * cell["T1"])

with QiJob() as multi_t1: # Get a reference for the job description
    cells = QiCells(5) # This experiment is designed for 5 qubits
    length = QiTimeVariable() # Create a variable the sequencers will track
    with ForRange(length, 0, 8e-6, 400e-9): # Vary length from 0 to 8µs in 0.4µs steps
        for q in cells: # Exploit Python's interpreter to repeat code for all cells
            PiPulse(q) # Excite qubit into state |1>
            Wait(q, length) # Wait variable time for qubit energy to decay
            Measurement(q, save_to="result") # Measure the obtained state
            Thermalize(q) # Reset qubit into thermal equilibrium before repeating

qic = QiController('ip-address') # Connecting to the system (calibration not shown)
sample = QiSample(5) # Sample with the qubit properties (loading values not shown)

multi_t1.run(qic, sample, averages=1000) # Compile + run the QiJob 1000x with sample
# The measured results can now be accessed via:
multi_t1.cells[i].data("result") # i defines from which qubit/cell the result is
```

Just Accepted

QiCode is designed in a way that a control flow graph of the Python-based experiment description is directly created when the Python interpreter steps through the QiJob definition. For calculations, the Python interpreter will create an abstract syntax tree as representation as all operators are overloaded. These abstract representations are stored inside the `QiJob` object until our custom, Python-based compiler is invoked. During compilation, first, the placeholder values from the `QiCells` object will be replaced by the values from the `QiSample` object. For the remaining compilation, the visitor pattern is used to translate the control flow graph step by step into the

Table 1. Resource utilization on a Xilinx XCZU28DR RFSoC for 10 cells and 8 DACs and ADCs. Categories are configurable logic blocks (CLB), block RAMs (BRAM), and digital signal processing slices (DSP).

Entity	CLB	BRAM	DSP
Available resources	53 160	1080	4272
Full design	44 074 (82.9 %)	770 (71.3 %)	1080 (25.2 %)
Single cell	3596 (6.76 %)	77 (7.12 %)	108 (2.53 %)
Sequencer	1081 (2.04 %)	1 (0.09 %)	4 (0.09 %)
Signal generator	531 (1.00 %)	22 (2.04 %)	20 (0.47 %)
Signal recorder	733 (1.38 %)	22 (2.04 %)	48 (1.12 %)
Data storage	193 (0.36 %)	4 (0.37 %)	0
Pulse Player	651 (1.22 %)	6 (0.56 %)	16 (0.37 %)
Digital trigger	220 (0.42 %)	0	0
WB infrastructure	396 (0.74 %)	0	0
Cell coordinator	242 (0.46 %)	0	0
Cell signal router	16 273 (30.6 %)	0	0

configuration values of the PL modules and assembler instructions of the sequencers within the digital unit cells. This information is then loaded onto the system, and the execution is triggered. Afterwards, the result data is collected and transferred back to the Python client where it can be accessed using the `QiJob` object.

8 RESULTS & PERFORMANCE

We benchmarked our design using a Xilinx ZCU111 evaluation board with a custom-built analog frontend. The resource utilization of the complete design is provided in the following section. Each PL module is thoroughly unit tested during development and within a continuous integration workflow. We verified the correct operation of the complete design using the platform in a loop-back configuration and with an oscilloscope. Afterwards, we also performed experiments with actual superconducting qubits to show that operation in the field is working as expected. One exemplary experiment is presented below.

8.1 Resource Utilization

The resource utilization of the design is given in Table 1 for 10 digital unit cells and 8 DACs and ADCs. It is currently limited by the amount of available configurable logic blocks, especially from the cell signal router. The required amount grows substantially with every added digital unit cell due to the additional logic in the cell signal router which also complicates the timing of the design. A substantial reduction of the required resources could be achieved by implementing the cell signal router using partial reconfiguration. In this case, one would not have to cover all possible routing scenarios with one fixed PL design. Another resource limitation are the BRAMs, mainly due to the required resources for the NCOs (1.48 % per NCO) inside the signal recorders and generators. However, special Ultra RAM blocks inside the RFSoC could be used in addition to normal BRAM blocks. For most experiments where 8 DAC outputs are sufficient, 10 unit cells are more than enough though. Yet, when utilizing another RFSoC with more channels, optimizations should be considered to allow for more unit cells on the system.

Just Accepted

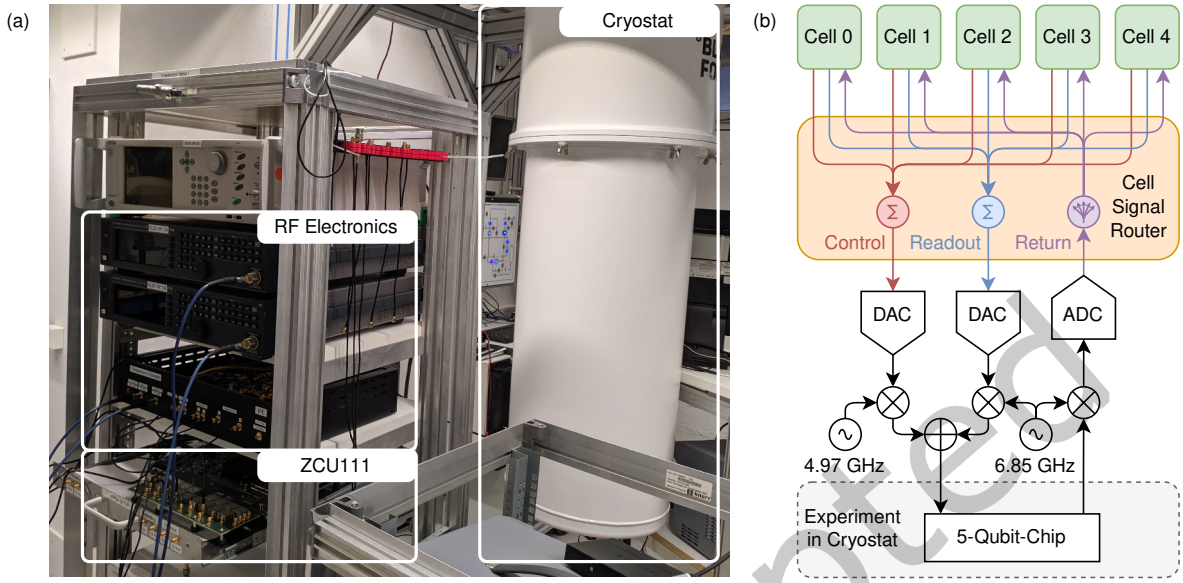

Fig. 13. (a) Photograph of the experiment setup including the cryostat containing the five qubit chip at roughly 15 mK. (b) Simplified sketch of the signal routing between digital unit cells and the five qubit chip.

Just Accepted

8.2 Functional Verification Utilizing a Five Qubit Chip

To test the system in the field, we used a well-characterized qubit chip with five superconducting Transmon qubits [17]. The qubits are not coupled to each other but via separate readout resonators to a common microwave line. Both control and readout pulses are fed into the chip on a single input line using frequency-division multiplexing. The complex-valued base-band frequencies for these signals range from −260 MHz to 230 MHz by using in-phase and quadrature components. Typical control pulses are around 50 ns long. From the 10 available unit cells of our design, we utilize five, one for each qubit. In the RF electronics, we use two I/Q mixers with separate local oscillator microwave sources, one to up-convert all control pulses, and one for all readout pulses. For this, the readout signals of the five digital unit cells are digitally combined in the cell signal router. The same applies to the control signals. After up-conversion, both RF signals are combined onto a single microwave line. By using a complex-valued base-band, the RF signals can be located flexibly on both sides of the local oscillator. A photograph and sketch to illustrate the experiment setup is given in Fig. 13.

As a first step, all five qubits need to be characterized and the necessary parameters to perform experiments determined. The first parameter is the frequency of the readout resonator. With it, one can then read out the qubit state. Our system can be used like a vector network analyzer to output a continuous tone with the readout signal generator and demodulate the response in the signal recorder. By changing the frequency of the internal NCO in both signal generator and recorder, a frequency sweep can be performed. As each qubit has its own unit cell, this can even be done simultaneously for all five qubits, as shown in Fig. 14. Here, each cell performs the signal generation and processing for one resonator sweep. The output signals are then flexibly combined within the cell signal router and forwarded to one DAC output. Likewise, the returning signal from the ADC is distributed to the respective cells.

After the characterization of frequencies, signal amplitudes and pulse lengths, different experiments can be performed. One type of experiment is to determine the coherence time T_2 of the qubits. This is the characteristic

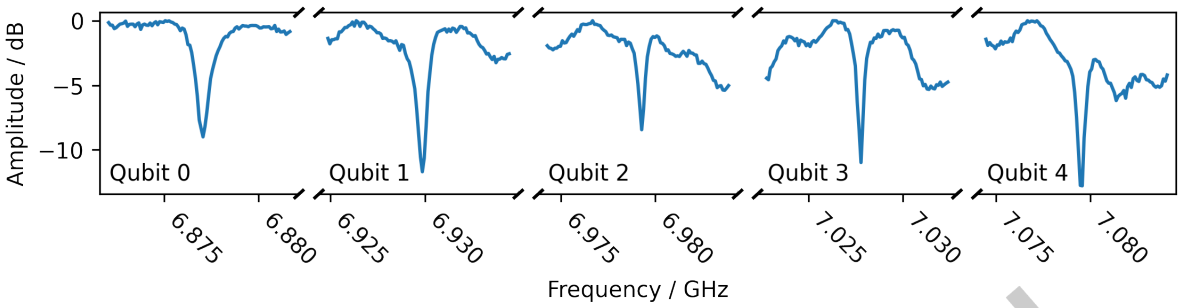

Fig. 14. Simultaneous VNA-like measurements to find the frequency of the readout resonators for all five qubits. For each frequency section, the amplitude has been separetely normalized to the largest value.

timescale after which quantum information inside the qubit is lost due to external influences that disturb the quantum state. It can be measured using a Ramsey pulse sequence [16]. With the qubit starting in state $|0\rangle$, one performs a $\pi/2$ rotation around the X axis to end up on the equator of the Bloch sphere. There, one waits a variable delay before performing another $\pi/2$ rotation. When the control pulses are slightly detuned from the actual qubit frequency, by 5 MHz in our case, the state will oscillate around the bloch sphere during the waiting time. Then, the second $\pi/2$ rotation will not bring the qubit into the $|1\rangle$ state, but depending on the duration of the rotation to another state. This results in an oscillation between the $|0\rangle$ and $|1\rangle$ state with respect to the delay. In Fig. 15, the measurement result for the simultaneous measurement on all five qubits is shown. The state of the qubit is encoded in the phase response of the corresponding readout resonator due to the dispersive readout. One can see an exponential decay of the envelopes which is due to the finite coherence time as the information of the state is lost when waiting too long between the pulses. The decay constant is equivalent to the coherence time T_2. The values are extracted from damped sine fits and also given in the figure. They are within the expected range obtained from previous separate measurements with the same chip and different electronics.

We also performed a set of other standard experiments that all yield results comparable to previous, consecutive measurements. These measurements have been conducted without our system, indicating that our RFSoC-based approach can be used as drop-in replacement to state-of-the-art electronics without accuracy losses. Furthermore, the presented unit cell approach features intuitive parallel control and readout of all five qubits, thereby significantly reducing execution time compared to the previous measurements by more than 80 %.

9 CONCLUSION

We presented the modular FPGA firmware of our RFSoC-based qubit control electronics. Our design is based on a digital unit cell that contains all necessary logic and capabilities to interact with a single qubit. All components of the cell, including the RISC-V-based sequencer, implement the required deterministic latency and support high-level programmability. Our custom Wishbone bus implementation contains additional features such as broadcasts for versatile synchronous triggering. Synchronization and data exchange between the cells is facilitated by a star-point structure dubbed cell coordinator. Versatile routing and frequency-division multiplexing of generated signals is achieved within the cell signal router which can be freely configured using its register interface. Using a Xilinx XCZU28DR RFSoC, we can implement 10 digital unit cells in the PL. For most applications, this will be more unit cells than the amount of physical qubits which can be addressed with the eight DAC outputs the chip offers.

Just Accepted

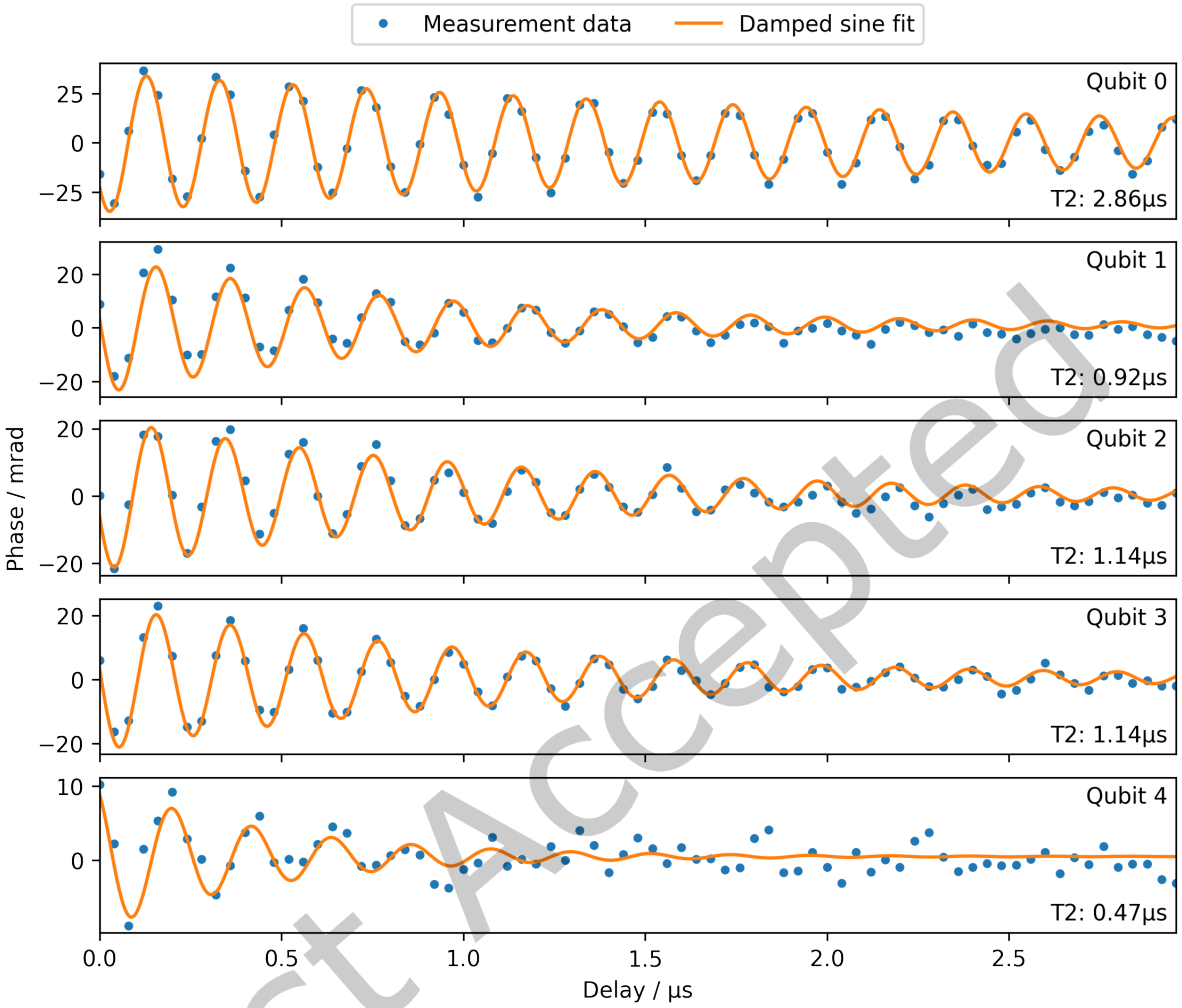

Fig. 15. Simultaneous measurements to extract the decoherence time T_2 with five qubits using a Ramsey pulse sequence. 100 000 repetitions have been performed for averaging.

Just Accepted

With our system, all relevant single- and multi-qubit operations can be easily performed with high precision. This applies both for research with superconducting qubits, as well as for implementations as interface between a quantum processor and the classical computing domain. We performed various experiments using a superconducting five qubit chip to verify correct operation of the system. Future work will focus on performing more complex experiments, continued optimization of the design, and multi-system synchronization for further scaling.

ACKNOWLEDGMENTS

Funding was provided by the Helmholtz Association. The authors acknowledge the financial support by the German Federal Ministry of Education and Research in the framework of PtQube (FKZ:13N15015). Richard Gebauer acknowledges support by the State Graduate Sponsorship Program (LGF). Nick Karcher acknowledges support by the Karlsruhe School of Elementary Particle and Astroparticle Physics (KSETA). We are grateful for the experimental support and infrastructure of the Institute of Physics at Karlsruhe Institute of Technology (KIT). We acknowledge Qkit [24] for providing a convenient measurement software framework for applications in quantum computing with superconducting quantum bits.

REFERENCES

[1] Christian Kraglund Andersen, Ants Remm, Stefania Lazar, Sebastian Krinner, Johannes Heinsoo, Jean-Claude Besse, Mihai Gabureac, Andreas Wallraff, and Christopher Eichler. 2019. Entanglement stabilization using ancilla-based parity detection and real-time feedback in superconducting circuits. *npj Quantum Information* 5, 1 (2019), 1–7. https://doi.org/10.1038/s41534-019-0185-4

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (01 Oct. 2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[3] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A. G. Fowler, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and John M. Martinis. 2016. Digitized adiabatic quantum computing with a superconducting circuit. *Nature* 534, 7606 (01 June 2016), 222–226. https://doi.org/10.1038/nature17658

[4] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. 2020. Quantum Algorithms for Quantum Chemistry and Quantum Materials Science. *Chemical Reviews* 120, 22 (25 Nov. 2020), 12685–12717. https://doi.org/10.1021/acs.chemrev.9b00829

[5] Cloud Native Computing Foundation. 2020. *gRPC - A high-performance, open source universal RPC framework.* https://grpc.io/

[6] Vedran Dunjko and Hans J Briegel. 2018. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics* 81, 7 (June 2018), 074001. https://doi.org/10.1088/1361-6633/aab406

[7] R. P. Feynman. 1982. Simulating Physics with Computers. *International Journal of Theoretical Physics* 21 (June 1982), 467–488.

[8] Richard Gebauer, Nick Karcher, Daria Gusenkova, Martin Spiecker, Lukas Grünhaupt, Ivan Takmakov, Patrick Winkel, Luca Planat, Nicolas Roch, Wolfgang Wernsdorfer, Alexey V. Ustinov, Marc Weber, Martin Weides, Ioan M. Pop, Oliver Sander, Aleksey Fedorov, and Alexey Rubtsov. 2020. State preparation of a fluxonium qubit with feedback from a custom FPGA-based platform. *AIP Conference Proceedings* 2241, 1 (2020), 020015. https://doi.org/10.1063/5.0011721 arXiv:https://aip.scitation.org/doi/pdf/10.1063/5.0011721

[9] Richard Gebauer, Nick Karcher, Jonas Hurst, Marc Weber, and Oliver Sander. 2021. Taskrunner: A Flexible Framework Optimized for Low Latency Quantum Computing Experiments. In *2021 IEEE 34th International System-on-Chip Conference (SOCC)*. IEEE, 123–128. https://doi.org/10.1109/SOCC52499.2021.9739306

[10] Edward Gerjuoy. 2005. Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers. *American Journal of Physics* 73, 6 (2005), 521–540. https://doi.org/10.1119/1.1891170 arXiv:https://doi.org/10.1119/1.1891170

[11] IBM. 2021. *IBM Quantum.* https://quantum-computing.ibm.com/

[12] IBM. 2022. *Qiskit - Open-Source Quantum Development.* https://qiskit.org/

[13] N. Karcher, R. Gebauer, R. Bauknecht, R. Illichmann, and O. Sander. 2021. Versatile Configuration and Control Framework for Real Time Data Acquisition Systems. *IEEE Transactions on Nuclear Science* 68, 8 (2021), 1899–1906. https://doi.org/10.1109/TNS.2021.3084355

[14] Keysight. 2019. *Quantum Engineering Toolkit (QET) - Data Sheet.* https://www.keysight.com/us/en/assets/7018-06423/data-sheets/5992-3503.pdf

[15] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. 2007. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A* 76 (Oct. 2007), 042319. Issue 4.

Just Accepted

[16] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. 2019. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews* 6, 2 (2019), 021318. https://doi.org/10.1063/1.5089550 arXiv:https://doi.org/10.1063/1.5089550

[17] M. Kristen, A. Schneider, A. Stehli, T. Wolz, S. Danilin, H. S. Ku, J. Long, X. Wu, R. Lake, D. P. Pappas, A. V. Ustinov, and M. Weides. 2020. Amplitude and frequency sensing of microwave fields with a superconducting transmon qudit. *npj Quantum Information* 6, 1 (25 Jun 2020), 57. https://doi.org/10.1038/s41534-020-00287-w

[18] J. Majer, J. M. Chow, J. M. Gambetta, Jens Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. 2007. Coupling superconducting qubits via a cavity bus. *Nature* 449, 7161 (01 Sep 2007), 443–447. https://doi.org/10.1038/nature06184

[19] David C. McKay, Stefan Filipp, Antonio Mezzacapo, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. 2016. Universal Gate for Fixed-Frequency Qubits via a Tunable Bus. *Phys. Rev. Applied* 6 (Dec 2016), 064007. Issue 6. https://doi.org/10.1103/PhysRevApplied.6.064007

[20] Nissim Ofek, Andrei Petrenko, Reinier Heeres, Philip Reinhold, Zaki Leghtas, Brian Vlastakis, Yehan Liu, Luigi Frunzio, SM Girvin, Liang Jiang, et al. 2016. Extending the lifetime of a quantum bit with error correction in superconducting circuits. *Nature* 536, 7617 (2016), 441–445.

[21] OpenCores 2010. *Wishbone B4 - WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores.* OpenCores. https://cdn.opencores.org/downloads/wbspec_b4.pdf

[22] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alán Aspuru-Guzik. 2012. Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific Reports* 2, 1 (13 Aug. 2012), 571. https://doi.org/10.1038/srep00571

[23] Alexander P. M. Place, Lila V. H. Rodgers, Pranav Mundada, Basil M. Smitham, Mattias Fitzpatrick, Zhaoqi Leng, Anjali Premkumar, Jacob Bryon, Andrei Vrajitoarea, Sara Sussman, Guangming Cheng, Trisha Madhavan, Harshvardhan K. Babla, Xuan Hoang Le, Youqi Gang, Berthold Jäck, András Gyenis, Nan Yao, Robert J. Cava, Nathalie P. de Leon, and Andrew A. Houck. 2021. New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds. *Nature Communications* 12, 1 (19 Mar 2021), 1779. https://doi.org/10.1038/s41467-021-22030-5

[24] Qkitgroup. 2020. *Qkit - a quantum measurement suite in python.* https://github.com/qkitgroup/qkit

[25] Quantum Machines. 2019. *The Quantum Orchestration Platform.* https://www.quantum-machines.co/platform/

[26] D. Ristè, C. C. Bultink, K. W. Lehnert, and L. DiCarlo. 2012. Feedback Control of a Solid-State Qubit Using High-Fidelity Projective Measurement. *Phys. Rev. Lett.* 109 (Dec 2012), 240502. Issue 24. https://doi.org/10.1103/PhysRevLett.109.240502

[27] D. Rosenberg, D. Kim, R. Das, D. Yost, S. Gustavsson, D. Hover, P. Krantz, A. Melville, L. Racz, G. O. Samach, S. J. Weber, F. Yan, J. L. Yoder, A. J. Kerman, and W. D. Oliver. 2017. 3D integrated superconducting qubits. *npj Quantum Information* 3, 1 (09 Oct 2017), 42. https://doi.org/10.1038/s41534-017-0044-0

[28] B. Schumacher. 1995. Quantum coding. *Phys. Rev. A* 51 (April 1995), 2738–2747. Issue 4.

[29] Aaron Somoroff, Quentin Ficheux, Raymond A. Mencia, Haonan Xiong, Roman V. Kuzmin, and Vladimir E. Manucharyan. 2021. Millisecond coherence in a superconducting qubit. arXiv:2103.08578 [quant-ph]

[30] Zurich Instruments. 2019. *Quantum Computing Control System.* https://www.zhinst.com/others/quantum-computing-control-system-qccs

Just Accepted