2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-2027-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/HPCA53966.2022.00037

DigiQ: A Scalable Digital Controller for Quantum Computers Using SFQ Logic

Mohammad Reza Jokar[1], Richard Rines[1,3], Ghasem Pasandi[2,4], Haolin Cong[2], Adam Holmes[1†], Yunong Shi[5], Massoud Pedram[2], and Frederic T. Chong[1,3*]

[1]Department of Computer Science, University of Chicago
[2]Department of Electrical and Computer Engineering, University of Southern California (USC)
[3]Super.tech, [4]NVIDIA, [5]Amazon Braket

Email: jokar@uchicago.edu, richrines@alum.mit.edu, {pasandi,haolinco}@usc.edu, adholmes@uchicago.edu, shiyunon@amazon.com, pedram@usc.edu, chong@cs.uchicago.edu

***Abstract*—The control of cryogenic qubits in today's superconducting quantum computer prototypes presents significant scalability challenges due to the massive costs of generating/routing the analog control signals that need to be sent from a classical controller at room temperature to the quantum chip inside the dilution refrigerator. Thus, researchers in industry and academia have focused on designing in-fridge classical controllers in order to mitigate these challenges. Due to the maturity of CMOS logic, many industrial efforts (Microsoft, Intel) have focused on Cryo-CMOS as a near-term solution to design in-fridge classical controllers. Meanwhile, Superconducting Single Flux Quantum (SFQ) is an alternative, less mature classical logic family proposed for large-scale in-fridge controllers. SFQ logic has the potential to maximize scalability thanks to its ultra-high speed and very low power consumption. However, architecture design for SFQ logic poses challenges due to its unconventional pulse-driven nature and lack of dense memory and logic. Thus, research at the architecture level is essential to guide architects to design SFQ-based classical controllers for large-scale quantum machines.**

In this paper, we present *DigiQ*, the first system-level design of a Noisy Intermediate Scale Quantum (NISQ)-friendly SFQ-based classical controller. We perform a design space exploration of SFQ-based controllers and co-design the quantum gate decompositions and SFQ-based implementation of those decompositions to find an optimal SFQ-friendly design point that trades area and power for latency and control while ensuring good quantum algorithmic performance. Our co-design results in a single instruction, multiple data (SIMD) controller architecture, which has high scalability, but imposes new challenges on the calibration of control pulses. We present software-level solutions to address these challenges, which if unaddressed would degrade quantum circuit fidelity given the imperfections of qubit hardware.

To validate and characterize *DigiQ*, we first implement it using hardware description languages and synthesize it using state-of-the-art/validated SFQ synthesis tools. Our synthesis results show that *DigiQ* can operate within the tight power and area budget of dilution refrigerators at >42,000-qubit scales. Second, we confirm the effectiveness of *DigiQ* in running quantum algorithms by modeling the execution time and fidelity of a variety of NISQ applications. We hope that the promising results of this paper motivate experimentalists to further explore SFQ-based quantum controllers to realize large-scale quantum machines with maximized scalability.

***Keywords*-SFQ-based quantum gate; Quantum optimal control; Scalable quantum computer; Cryogenic electronic;**

†Adam Holmes is now at HRL Laboratories, LLC.

*Disclosure: Fred Chong is Chief Scientist at Super.tech and an advisor to Quantum Circuits, Inc.

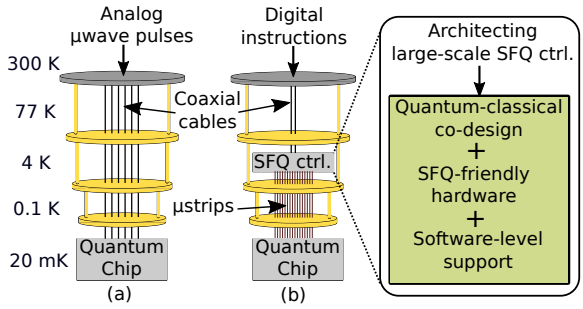


Figure 1: (a) Today's controller design: controller at room temperature, (b) *DigiQ*: controller close to quantum chip.

I. Introduction

Superconducting quantum computing is one of the promising technologies for building a quantum computer [1], [2], with many prototypes having been manufactured in the recent years [3]–[6]. However, today's prototypes rely on sending separate analog microwave control pulses for each qubit through coaxial cables from a classical controller at room temperature to the quantum chip inside the dilution refrigerator (see Fig. 1(a)). This design is simple and straightforward, however presents severe scalability challenges due to the massive costs of generating/routing the analog microwave signals, and significant heat dissipation at millikelvin temperatures due to using a large number of high bandwidth coaxial cables [7]–[9]. Thus, it is essential to build compact controllers as close as possible to quantum chips in order to generate and route the control signals locally and address the scalability problem of today's systems (see Fig. 1(b) for an example of such a controller).

978-1-6654-2027-3/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCA53966.2022.00037

Cryo-CMOS is an excellent near-term solution to increase the scalability of today's quantum machines, and controller prototypes based on Cryo-CMOS have been manufactured [10] which can scale to >800 qubits (see Sec. III). Meanwhile, Superconducting Single Flux Quantum (SFQ) logic [11], [12] is an emerging classical logic family and is recognized as a promising solution to maximize the scalability of in-fridge controllers due to its unique characteristics such as ultra-high speed and very low power [7]–[9], [13]. However, a key remaining step is designing an SFQ-friendly controller architecture that operates within the tight power and area budget of dilution refrigerators at large scales, while ensuring good quantum algorithmic performance.

Prior work has demonstrated quantum gates based on SFQ logic [8], [9], [13], and has outlined a vision for an SFQ-based controller for fault-tolerant quantum computing that relies on repeated streaming of quantum instructions that are stored in SFQ registers [7]. These studies done by physicists are essential in order to show the feasibility of performing SFQ-based quantum gates, and envision the potentials of SFQ logic in controlling large-scale quantum chips. However, the following architectural shortcomings remain to be addressed: (1) prior work does not consider scenarios where we are no longer primarily repeating the same quantum instructions. Thus, they are especially not suitable for running Noisy Intermediate Scale Quantum (NISQ) algorithms; (2) there has not been a detailed synthesis of a complete SFQ-based controller architecture in order to get an accurate estimate of power and area, and determine the scale we can achieve with such controllers given the power and area budget of dilution refrigerators; (3) there has not been an analysis of the implications of SFQ-based quantum controllers on quantum algorithmic performance. This analysis is necessary to assess the cost and effectiveness of SFQ-based controller designs; (4) there has not been an analysis on the robustness of SFQ-based controllers to the qubit imperfections in NISQ systems.

In this paper, we address the above architecture-level shortcomings and present *DigiQ*, the first system-level design of a scalable NISQ-friendly SFQ-based classical controller. Inspired by the SuperNPU paper [14], which demonstrates architecture design for SFQ-based neural processing units, our paper guides architects to design SFQ-based controller architectures for large-scale quantum computers.

Architecture design for SFQ logic is different from that of CMOS logic, due to its unconventional pulse-driven nature and lack of dense memory/logic. In addition, implementation of quantum gates using SFQ pulses is different from that of microwave pulses. Thus, novel SFQ-friendly controller architecture designs are essential. We perform a design space exploration of SFQ-based controllers and co-design the quantum gate decompositions and SFQ-pulse implementation of those decompositions to ensure that our design both works within the tight power and area budget of dilution refrigerators at large scales and provides good algorithmic performance.

Quantum gate parallelism is essential to preserve good quantum algorithmic performance in many NISQ applications [15]. Our quantum-classical co-design demonstrates that due to the lack of dense memory/logic in SFQ and tight power and area budget of dilution refrigerators, we cannot afford to simultaneously send tailored quantum gates to many qubits at large scales. The implementation of quantum algorithms with significant gate parallelism therefore requires sharing SFQ-based quantum instructions among multiple qubits (i.e., single instruction, multiple data (SIMD)). Getting good algorithmic performance on a SIMD NISQ architecture is especially challenging because of qubit variations and frequency drift, which typically require gates to be uniquely calibrated for each qubit. We propose novel software-level solutions to address these challenges and preserve SIMD parallelism.

To validate and characterize *DigiQ*, we first work out the details of quantum program execution flow, starting from our compiler at room temperature and ending with sending the control signals to qubits. Then, we implement our complete design in hardware description languages, and synthesize it using state-of-the-art/validated SFQ synthesis tools [16]–[19] to measure power, area, and delay values. Our synthesis results are obtained post-layout based on accurate extraction of the gate layouts and passive transmission lines and subsequently simulated using well-established tools for superconductive electronic applications [16], [20], [21], and are thus highly accurate and reliable. Finally, we show the effectiveness of *DigiQ* in running quantum algorithms by compiling a variety of NISQ algorithms for our system, and modeling the resulting execution time and fidelity.

We position ourselves as addressing the physical challenges in scaling up the NISQ machines. Our work is complementary to Perfect Intermediate Scale Quantum computing (PISQ) [22] approach, which suggests developing new quantum algorithms assuming perfect qubits and evaluating them with classical simulators. PISQ is a great approach since it separates the development of quantum algorithms and applications from hardware and architectural research and allows researchers to focus on the development of new quantum algorithms in various scientific fields.

To summarize, our key contributions are as follows:

- **Architecting SFQ-based controllers:** We guide architects to design large-scale SFQ-based controllers by taking into consideration the opportunities (efficient on-chip broadcast and ultra-fast clock) and challenges (lack of dense memory/logic) of SFQ.
- **Quantum-classical co-design:** By co-designing quantum gate decompositions and SFQ-based implementation of those decompositions, we present *DigiQ*, an SFQ-friendly SIMD controller architecture.
- **Addressing the SIMD calibration challenges:** We

present software solutions to address the quantum gate calibration challenges of SIMD hardware to make it robust to imperfections in qubit hardware.
- **Characterizing controller hardware:** We implement *DigiQ* in hardware description languages and show its feasibility in large scales in terms of power, area, and delay using state-of-the-art/validated SFQ synthesis tools.
- **Characterizing algorithmic performance:** We confirm the effectiveness of *DigiQ* in running quantum algorithms by compiling a variety of NISQ applications for *DigiQ* and modeling their execution time and fidelity.

The rest of the paper is organized as follows. Sec. II provides a background on quantum controllers, SFQ logic, and discusses the opportunities and challenges of SFQ-based controllers. We present related work in Sec. III. Sec. IV demonstrates the details of *DigiQ*, our scalable digital SFQ-based quantum controller architecture. Sec. V discusses the quantum gate calibration challenges of *DigiQ*, and presents software-level solutions to address them. Sec. VI shows our methodology and thorough evaluation of *DigiQ*. Finally, Sec. VII concludes the paper and discusses the future work.

II. Background and Motivation

Here we provide a background on quantum computing followed by a discussion of quantum gates/controllers in existing systems and their limitations. We then present a background on SFQ logic and discuss the opportunities and challenges of SFQ-based controllers.

A. Quantum computing

A quantum algorithm specifies a series of transformations on quantum systems called qubits, which are analogous information carriers to classical bits. The state of a single qubit can be represented as a linear combination of two states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where the amplitudes $\alpha, \beta \in \mathbb{C}$ satisfy $|\alpha|^2 + |\beta|^2 = 1$. It is useful to visualize the state of a single qubit as a vector on the unit Bloch sphere as shown in Fig. 2(a), where $\alpha = \cos\theta/2$ and $\beta = e^{i\phi}\sin\theta/2$. A multi-qubit state may be written as $|\psi\rangle = \sum_i \alpha_i\,|i\rangle$, where $\sum_i |\alpha_i|^2 = 1$ and $|i\rangle = |i_{n-1}\rangle \ldots |i_1\rangle\,|i_0\rangle$ are the computational basis states of the n-qubit quantum system.

Quantum gates are unitary operators which modify the state of the qubit system. Any single-qubit gate can be represented as a rotation on the unit Bloch sphere (see Fig. 2(a)). Rotations around any two axes can be combined to perform arbitrary single-qubit gates. Combined with a two-qubit *entangling* gate (i.e., a gate which cannot be decomposed into one-qubit gates), this is sufficient for universal quantum computation [23].

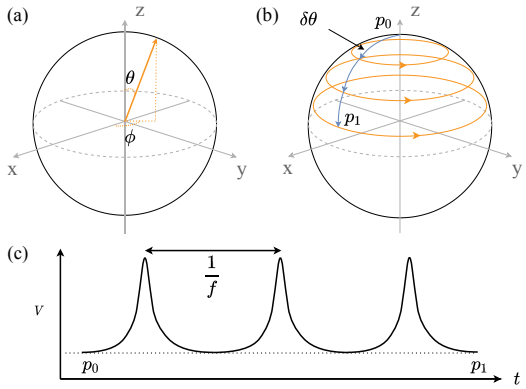


Figure 2: (a) Bloch sphere representation of a qubit; (b) SFQ driven trajectory. The blue trajectory is driven by the periodic SFQ pulse train shown in (c), and the orange trajectory is driven by the qubit free evolution; (c) SFQ pulse train in the time domain. f is the qubit oscillation frequency.

B. Superconducting qubit controllers and their limitations

Superconducting qubits are nonlinear LC circuits built with Josephson junctions (JJ) that operate at near absolute zero temperature and oscillate at microwave frequencies. Qubits are defined using the quantized ground ($|0\rangle$) and first excited ($|1\rangle$) states of the oscillator, where the qubit's oscillation frequency is defined as the energy difference between the two levels. *Transmons* are a simple form of superconducting qubit comprising a JJ and a shunt capacitor, designed to reduce the qubit's sensitivity to electrical charge noise [24], and is the qubit technology in many state-of-the-art systems [3]. *Flux-tunable* transmons allow tuning the qubit oscillation frequency, and are implemented by replacing the transmon's JJ with a pair of parallel junctions separated by a small distance. The qubit frequency can then be shifted by driving an external magnetic flux (using a small electrical current, $\sim$1 mA) through the enclosed loop [25]. The relationship between frequency and flux can be fine tuned by varying the parameters of each JJ individually, creating what's known as an *asymmetric* transmon [26].

Superconducting quantum computer prototypes perform single-qubit gates by driving transitions between the oscillator's $|0\rangle$ and $|1\rangle$ states using microwave pulses. Controllable multi-qubit operations require a means of selectively interacting qubits through some coupling architecture. Two-qubit gates such as CZ can be implemented using flux-tunable transmons by changing the frequency of the qubits temporarily such that their quantum states are on resonance [25].

Superconducting quantum computers have received significant attention due to their convenient qubit design and configurability [25], leading to rapid advances in their size, coherence time, and gate fidelity [3]–[5]. However, they also pose significant challenges which must be addressed for superconducting quantum computing to be scalable.

Superconducting qubits need cooling to very low temperatures (∼20 mK) which is expensive and complicates control hardware. Further, unlike technologies such as trapped ion (in which qubit uniformity is guaranteed by nature [27]), the characteristics of superconducting qubits are subject to variation and drifts over time.

The controller design in today's prototypes relies on separate cables from room temperature to control individual qubits [3]. This approach has severe scalability issues. First, there is a massive electronics cost for generating and routing the analog control pulses from room temperature including the cost of arbitrary waveform generators (AWGs), microwave sources, IQ mixers (which modulate the in-phase and out-of-phase components of the drive signal [25]), and amplifiers and attenuators that are used for thermal property matching at each stage of the fridge [7], [8], [25]. Second, the cooling power at millikelvin stage of the fridge is very limited ($<$10 μW [28]) and a large number of high bandwidth coaxial cables create a big heat load problem at this stage. Thus, alternative quantum controller architectures are needed to realize scalable and robust quantum machines.

C. Opportunities and challenges of SFQ quantum controllers

Superconducting Single Flux Quantum (SFQ) is a magnetic-pulse based device and a single quantum of magnetic flux, $\Phi_0 = h/2e$ = 2.07 mV·ps, is used for logic bit representation. In this representation, presence of a pulse has the meaning of "logic-1", while absence of a pulse is considered as a "logic-0". Operation of SFQ logic is based on overdamped JJs [11], [12], [29]–[31]. There are two types of D-Flip-Flops (DFFs) in this technology: Destructive Read Out (DRO) DFF and Non-Destructive Read Out (NDRO) DFFs. In the first type the stored data will be erased after one read operation and the DFF will be reset back to its zero state. However, in the second type, multiple read operations can be done without erasing the content of DFF. SFQ devices with switching delay of 1 ps and switching energy of 10^{-19} J are considered great candidates to provide high speed solutions post-silicon and post-CMOS [14]. Moreover, these Niobium-based devices are extremely low power and despite their cryo-cooling overhead they still consume significantly less power compared to the state-of-the-art silicon-based devices [32]. These unique properties make SFQ an attractive logic technology to implement classical controllers with maximized scalability for quantum computers.

SFQ can be utilized not only to implement classical controller logic, but also to perform quantum gates locally. Prior work demonstrated that SFQ pulse trains can be utilized to perform single-qubit gates [8], [13], [33]. For example, an intuitive approach to do rotations along the y-axis, $Ry(\theta)$, is to apply one SFQ pulse every qubit oscillation period as shown in Fig. 2. The time integral of an SFQ pulse is equal to the superconducting flux quantum and determines the energy deposited in the qubit. The result of this energy deposition is a small rotation of $\delta\theta$ around the y-axis. We can perform a rotation along the y-axis by keep applying one SFQ pulse every qubit period. An arbitrary single-qubit gate can be represented by a bitstream (denoted as SFQ bitstream); the gate is applied by applying the SFQ bitstream to the qubit, one bit at a time: if the bit is 1 (0), we apply (don't apply) an SFQ pulse to the qubit at the corresponding SFQ chip cycle.

Despite its high potentials, SFQ imposes unique constraints on the controller design. First, limited JJ density in existing technology (100X-10,000X lower density than CMOS [34]) leads to lack of dense memory/logic in SFQ. Second, SFQ design is different from that of CMOS due to unconventional pulse-driven nature of SFQ logic. SFQ logic gates receive clock signals and they are evaluated by arrival and consumption of clock pulses. Therefore, balancing the circuit path by inserting extra DFFs is essential to ensure that inputs are consumed at correct clock cycles [17]. The extra DFFs further increase the logic area (and power), and might incur significant costs in complex logic designs. Thus, we need to take these constraints into consideration and (1) use limited SFQ storage; (2) keep the logic simple.

III. Related Work

In this section we discuss prior research on in-fridge quantum gates and controllers based on various technologies, and also SFQ-based accelerators.

A. Cryo-CMOS based quantum controllers

Due to maturity of CMOS technology, Cryo-CMOS is an attractive technology to do computation at the 4 K stage of the fridge. In [10], single-qubit gate operation using a Cryo-CMOS controller prototype is demonstrated, which is done by generating microwave control signals inside the fridge using the pulse information that is stored in on-chip SRAMs. The prototype presented in [10] can scale to $>$800 qubits given 12 mW/qubit power consumption reported in the paper and 10 W power budget [7]. In comparison, SFQ logic can potentially maximize the scalability of quantum controllers ($>$42,000 qubits in our SIMD design) due to its very low power consumption. Note that SIMD can potentially increase the scalability of today's Cryo-CMOS prototypes as well, which we see as important future work (see Sec. VII).

B. SFQ-based quantum gates and controllers

In [8], coherent control of a qubit using SFQ pulse trains is demonstrated using a DC/SFQ converter that is fabricated on the qubit chip; the experimental results in the paper show the feasibility of performing quantum gates using SFQ. In [9], the authors propose a method to find SFQ bitstreams for qubits with different frequencies using one single SFQ clock. A genetic algorithm is used in [13] as an approach to find efficient SFQ bitstreams to reduce leakage errors (i.e.,

Table I: Design space for SFQ-based single-qubit gate controllers.

	SFQ_MIMD_naive	SFQ_MIMD_decomp	SFQ_SIMD_decomp (DigiQ)	
			DigiQ_min	DigiQ_opt
Scalability	Limited by power, area, and bandwidth	Limited by power and area	High scalability	
Quantum program execution	No gate serialization	No gate serialization	Long decompositions	Potential serialization
Pulse calibration	Hardware	Hardware	Software	

unwanted energy transfer to states other than $|0\rangle$ and $|1\rangle$) and gate time. Reinforcement learning is another approach that has been utilized to find efficient SFQ bitstreams to perform quantum gates [35]. In [7], the authors outline a vision to perform fault tolerant quantum computing that relies on repeated streaming of stored SFQ bitstreams, and present a simple estimation of power by adding the power of SFQ registers. In contrast to these works, *DigiQ* is the first system-level design of a scalable NISQ-friendly SFQ-based controller.

C. SFQ-based accelerators

SFQ has been utilized to design hardware accelerators thanks to its ultra-high speed. In [36], the authors propose an SFQ-based error decoder to accelerate quantum error correction. In [14], the authors present an ultra-fast SFQ-based neural processing unit. In contrast to these works, our focus is on designing a scalable SFQ-based controller rather than accelerating a task.

IV. DigiQ quantum controller

In this section we provide guidelines for designing scalable SFQ-based controller architectures for universal quantum computation, and present *DigiQ*, our novel SFQ-based controller architecture.

A. SFQ-based universal quantum computation

1) Design space for single-qubit gate controllers: A naive design to do arbitrary single-qubit gates is to allocate separate SFQ registers for SFQ bitstreams per qubit (similar to [7]) and update them as needed. This design, denoted as *SFQ_MIMD_naive*, is similar in spirit to today's microwave-based designs, but relies on digital communication from room temperature rather than analog communication. *SFQ_MIMD_naive* is straightforward and provides quantum gate parallelism (i.e., multiple instruction, multiple data (MIMD) paradigm). However, it requires high communication bandwidth from room temperature in scenarios where we are not primarily repeating the same quantum gates, and thus need to update a large number of SFQ registers on-the-fly during the quantum program execution. In addition, *SFQ_MIMD_naive* is expensive in terms of power and area (5.01 mW/qubit and 13.9 mm^2/qubit just for 300-bit SFQ registers based on our results obtained using detailed SFQ synthesis tools and cell libraries). Thus, the scalability of *SFQ_MIMD_naive*, is limited by power/area/bandwidth.

In order to reduce the required bandwidth from room temperature, we can store a universal single-qubit gate set per qubit on the SFQ chip, and send the quantum gate decomposition information from the room temperature. In the simplest case where the single-qubit gate set includes two gates, we need to send only 1 bit per qubit at any given time from room temperature to select/apply one of the two gates. This design, denoted as *SFQ_MIMD_decomp*, reduces the bandwidth requirement significantly compared to *SFQ_MIMD_naive*. However, it further increases the power and area as it allocates more than one SFQ register per qubit. Thus, the scalability of *SFQ_MIMD_decomp* is still limited by power/area.

Finally, we demonstrate our scalable design. Tight power, area, and bandwidth budget of dilution refrigerators, and lack of dense memory in SFQ (see Sec. II) leads us to a design where we share SFQ bitstreams among a group of qubits (i.e., SIMD). Grouping is static and done at the design time, such that qubits in a group have the same nominal oscillation frequency; we show that we can compensate for frequency drift in software in Sec. V. Note that sharing the bitstreams can be done efficiently in SFQ by broadcasting the bitstreams (one bit per SFQ cycle) using splitter gates. This design, denoted as *SFQ_SIMD_decomp*, is a potential candidate to realize a controller with high scalability thanks to its reduced power, area, and bandwidth requirements compared to the other two designs. We therefore base *DigiQ* on this design. Table I summarizes the investigated design space.

2) Single-qubit gate decomposition: Out of the plethora of proposed quantum gate decomposition protocols for single-qubit gates, we must choose the ones that can be implemented efficiently using SFQ. We prefer a decomposition protocol that relies on storing/processing a limited number of SFQ bitstreams (see Sec. II). This is important because (1) there is a lack of dense memory in SFQ, thus we can afford to store only a limited number of SFQ bitstreams; (2) we need to keep the SFQ logic simple, thus we can afford to process only a limited number of SFQ bitstreams at any given time. We consider two SFQ-friendly gate decompositions and present their corresponding SFQ-based architecture designs.

Our first architecture, denoted as *DigiQ_min*, is based on a minimal universal single-qubit gate set that includes only 2-4 gates (e.g., $[Ry(\frac{\pi}{2}), T]$ is sufficient for universal single-

qubit computation [37], [38]) with the goal of minimizing the hardware cost. Our digital controller works with a clock (denoted as *quantum controller clock*), and similar to classical computers, each quantum program is decomposed into a number of quantum gates and each quantum gate is finished in a number of controller cycles. In *DigiQ_min*, single-qubit gates are decomposed into a sequence of the gates in the gate set, and each gate of the sequence is executed in one controller cycle. At the beginning of each controller cycle, the SFQ bitstreams corresponding to all the gate are broadcasted to a group of qubit controllers. Each qubit controller uses a simple SFQ-based multiplexer to select/apply one of the possible bitstreams using the control bits that are sent from the room temperature in each controller cycle.

A concern is that *DigiQ_min* needs long gate sequences to perform arbitrary single-qubit gates. Thus, we also present *DigiQ_opt*, in which we implement the continuous gate set $\{Ry(\frac{\pi}{2}), Rz(\phi); \phi \in [0, 2\pi)\}$, enabling constant-depth single-qubit gate decomposition [39]. $Ry(\frac{\pi}{2})$ can be implemented by storing an SFQ bitstream on the SFQ chip. The next question is: how to implement $Rz(\phi)$ gates for arbitrary ϕ efficiently?

In microwave-based systems, $Rz(\phi)$ gates can be done virtually (in software) by changing the phase of the microwave pulses of the consecutive gates [25], [39]. But, in the SFQ case, we do not have control over the phase of the SFQ bitstream. Thus, we need to perform the $Rz(\phi)$ gates in hardware. Thanks to the ultra-fast, precise clock on the SFQ chip (in the order of ps), we can perform arbitrary $Rz(\phi)$ by letting the qubits evolve freely for a precise number of SFQ chip clock cycles, which is equivalent to applying an SFQ bitstream consisting of only "0" bits. Every qubit oscillation period, a qubit performs a full rotation around the z-axis with a trajectory in the form of a unit circle (see Fig. 2). Now, for any ϕ, there is a point on the unit circle corresponding to $Rz(\phi)$ and we need to get as close as possible to that point in order to perform $Rz(\phi)$ with high fidelity. By applying a "0" bit every SFQ chip cycle, we get to multiple points on the unit circle in one qubit oscillation period, and we cover more points if we let the qubit evolve freely for more than one qubit period. The longer we let the qubit evolve freely, the more points we can get to on the unit circle and the more precisely we can approximate $Rz(\phi)$ for arbitrary ϕ. The fidelity of these gates is analyzed in Sec. V-A.

We now discuss implementation details of *DigiQ_opt* single-qubit gates. Any one-qubit gate can be decomposed as $U(\phi_3, \phi_2, \phi_1) = Rz(\phi_3)Ry(\frac{\pi}{2})Rz(\phi_2)Ry(\frac{\pi}{2})Rz(\phi_1)$. In *DigiQ_opt*, we break the $U(\phi_3, \phi_2, \phi_1)$ into three parts, $Ry(\frac{\pi}{2})Rz(\phi_1)$, $Ry(\frac{\pi}{2})Rz(\phi_2)$, and $Rz(\phi_3)$. We perform each of the first two parts in one controller cycle, and absorb the $Rz(\phi_3)$ to the next controller cycles. The controller cycle length is equal to the $Ry(\frac{\pi}{2})$ gate time plus $N * SFQ\_chip\_cycle\_length$, where N is the maximum number of SFQ chip cycles that we let the qubits evolve freely in order to perform $Rz(\phi)$ gates. We perform $Ry(\frac{\pi}{2})Rz(\phi)$ by applying a "0" bitstream of length $0 \leq d \leq N$ followed by the $Ry(\frac{\pi}{2})$ bitstream, which is equivalent to delaying the $Ry(\frac{\pi}{2})$ bitstream by d SFQ chip cycles as shown in Fig. 3. The value of d depends on the angle ϕ and is sent by the compiler at room temperature every controller cycle; the SFQ logic in *DigiQ_opt* delays the stored $Ry(\frac{\pi}{2})$ bitstream by d SFQ chip cycles and broadcasts it to a group of qubit controllers. The residual $Rz(\gamma)$ rotation at the end of the controller cycle (see Fig. 3) can be absorbed into the next gate on that qubit.

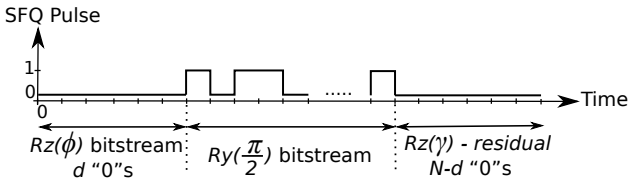


Figure 3: The sequence of gates in one cycle of *DigiQ_opt*.

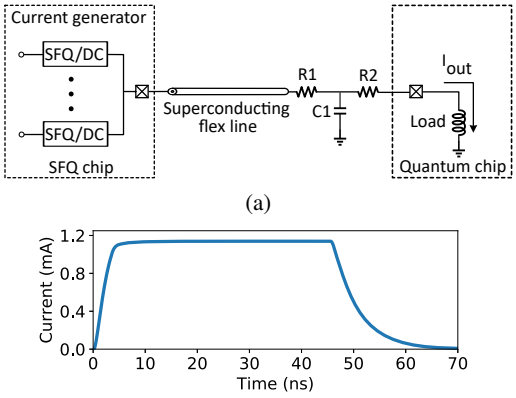


Figure 4: (a) Circuit schematic of our current generator design based on SFQ/DCs; (b) The electrical current pulse generated by our design to realize CZ gates on flux-tunable transmons.

The next question is: how many distinct $Ry(\frac{\pi}{2})Rz(\phi)$ gates out of a total of $N+1$ possible gates (denoted as BS) are needed every controller cycle? The answer depends on (1) the available power and area budget inside the fridge: providing more distinct gates requires more complex logic to delay the stored $Ry(\frac{\pi}{2})$ bitstream by more distinct values at the same time, and also broadcast and process more SFQ bitstreams; (2) the similarity between the gates that qubits inside a group perform in a given instant: there will be serialization inside a group if not enough distinct gates are available in a controller cycle. The lower the BS value, the lower the hardware cost but the higher the chance of serialization (see Sec. VI).

3) Two-qubit gate design: In this paper, we develop SFQ circuits to generate electrical current waveforms necessary to realize CZ gates on flux-tunable transons (see Sec. II)

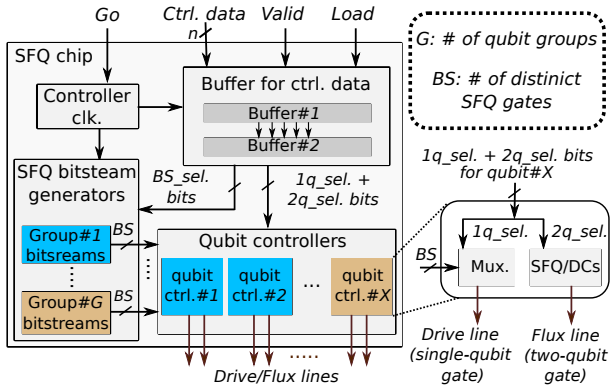


Figure 5: Overview of our *DigiQ* architecture.

inside the fridge. The electrical currents are transmitted to the quantum chip using superconducting microstrip flex lines (see Fig. 4(a)). We deploy an array of SFQ/DCs, which are commonly used SFQ blocks to convert the SFQ pulsed representation to DC voltage levels [40]. When a *start* signal is received by the SFQ/DC blocks, they start outputting electrical current, and they keep doing so until they receive a *stop* signal (we need to turn the SFQ/DCs on (off) in the beginning (end) of the CZ gate). In order to target specific pairs of qubits on a multi-qubit system we require current waveforms to be applied to both qubits simultaneously (we need one current generator per qubit). The only difference between this approach and prior flux-tunable implementation of CZ gates is that the electrical current is generated inside the fridge. Note that we use the same two-qubit gate design for both *DigiQ_min* and *DigiQ_opt*.

We use JSIM which is a detailed circuit simulator for superconductive electronic applications [21] to simulate the Fig. 4(a) current generator circuit. Fig. 4(b) shows the simulated current waveform, which is generated by enabling 25 SFQ/DC blocks (the values of $R1$, $R2$, and $C1$ are 0.05 ohm, 0.05 ohm, and 10 nF, respectively). In Sec. V-B, we use physical simulations to show that this waveform can be combined with software calibration techniques to realize low-error CZ gates even when qubits exhibit independent frequency variation.

Recent studies have also suggested realizing two-qubit cross-resonance gates using only SFQ pulses [35], [41]. The design of an SFQ controller architecture based on such gates would present a different set of challenges and opportunities, the study of which we save for future work.

B. *Overview of* DigiQ *architecture*

Now we put it all together and demonstrate an overview of our *DigiQ* architecture that is shown in Fig. 5. There are G groups of qubit controllers. At the beginning of each controller cycle, *BS* distinct single-qubit SFQ gates per each group are broadcasted to all the qubit controllers in the group. *BS_sel* bits are used to select the *BS* distinct single-qubit gates in *DigiQ_opt* (*DigiQ_min* does not need *BS_sel* bits since its universal single-qubit gate set includes only a few gates, which are all broadcasted). Each qubit controller uses an SFQ-based multiplexer and the *1q_sel* bits to either select/apply one of the *BS* delayed bitstreams, or apply none of them (e.g., in the two-qubit gate case). For two-qubit gates, the qubit controllers of the two qubits involved use the *2q_sel* bits to determine whether to start/stop performing CZ gate by sending *start/stop* signals to the SFQ/DCs.

BS_sel, *1q_sel*, and *2q_sel* control bits are sent from the room temperature every controller cycle using *Ctrl. data* cables; a *Valid* cable is used to determine the validity of control data on data cables. A *Load* cable is also used to load the SFQ bitstreams through the data cables, which is done offline (i.e., not during the program execution); each SFQ bitstream has $\leq$300 bits (the actual bitstream length depends on the target gate and system Hamiltonian). After the transmission of the control bits of the first controller cycle is finished, a *Go* signal is sent from room temperature to start the controller clock (which is implemented using a counter that counts up every SFQ chip cycle and resets every controller cycle). At the beginning of every controller cycle, the control bits that are already buffered in a buffer (*Buffer#1* in Fig. 5) are transferred to a second buffer (*Buffer#2* in Fig. 5) to be used by qubit controllers and SFQ bitstream generators. While executing the current controller cycle, the control bits of the next controller cycle are buffered in the first buffer.

V. Software Calibration of SIMD Hardware

Real superconducting qubits exhibit variations in their oscillation frequency which can drift from day to day in experimental settings [42], [43]. Quantum gates are implemented using precise control signals designed using careful physical models of qubit evolution (that is, their Hamiltonian) [25], which are extremely sensitive to small deviations in qubit frequency. The accuracy of quantum gates optimized for one set of qubit frequencies will therefore degrade rapidly if the same control signals are used on qubits exhibiting slight deviations from those frequencies. State-of-the-art quantum gates therefore require regular recalibration using experimental measurements of each qubit [42]. For small systems with MIMD control, qubit-specific gate calibration can be performed at the hardware level by precisely shaping control pulses for each qubit. This is not possible for a SIMD architecture such as *DigiQ*, in which control signals are shared among many qubits and so cannot be calibrated independently for each qubit.

Here, we show that this challenge can be overcome by moving gate calibration to the software level, eliding unscalable qubit-specific hardware configurability. Our approach is summarized in Fig. 6. In short, basis SFQ bitstreams that are stored on SFQ chip are fixed, and calibration is performed by adjusting the decomposition into the (frequency-dependent) set of basis quantum operations resulting from those fixed

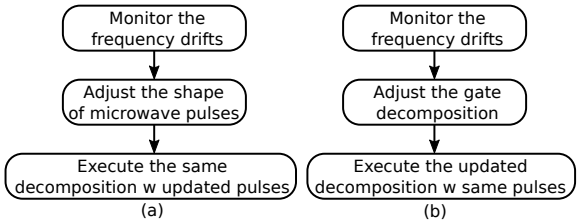


Figure 6: Calibration process in (a) today's microwave-based quantum machines; (b) *DigiQ*.

bitstreams; note that applying the same SFQ bitstream to multiple qubits with different qubit frequencies results in different quantum operations. Though the focus of this section is compensation for frequency drift, similar techniques could be employed to mitigate other sources of systematic error.

The gate errors reported through the remainder of this paper are calculated as $\epsilon = 1 - \overline{F}$, where $\overline{F}$ is the average gate fidelity [44], [45]. We calculate gate fidelities by modeling and simulating the Hamiltonian of the quantum hardware to provide an upper bound for the gate fidelity. This approach is also used in studies on both SFQ-based and microwave-based systems as a precursor to experimental work [9], [46]; *DigiQ* shows comparable gate fidelity to these studies as shown in Sec. VI-B2. Although we model the key dominant sources of systematic error, experimental evaluations on real hardware (currently available only at a small scale) are subject to various sources of noise which are not captured in the models, resulting in lower gate fidelities [8], [47].

A. Calibrating single-qubit gates

The key idea behind gate calibration on *DigiQ* is that these frequency-dependent operations still constitute universal gate sets for single-qubit computation [37]. We can therefore account for frequency drifts and hardware variation on each qubit by decomposing single-qubit gates on each qubit using the unique set of basis operations resulting from shared SFQ bitstreams. For both *DigiQ_opt* and *DigiQ_min*, the single-qubit calibration procedure then consists of four steps: (1) find SFQ bitstreams implementing a desired set of basis gates with high fidelity on qubits with no frequency variation, (2) characterize each qubit's actual oscillation frequency using experimental measurements [7], (3) use the learned bitstreams and measured frequencies to determine the actual basis operations implemented on each qubit by the shared bitstreams, and (4) compile quantum circuits using the actual single-qubit basis operations determined for each qubit. The (classical) complexity of these steps scales linearly with number of qubits and circuit size.

For steps (1) and (3) we employ single-qubit physical simulations of the expected single-qubit unitary evolution U_{bs} produced by an SFQ bitstream on transmon qubits (as done in [9]). To ensure we are fully accounting for state leakage, we model transmons using six energy levels, and then compute single-qubit gate fidelities by projecting the resulting evolution back into the two-level subspace (causing any leakage to be captured as additional error [45]).

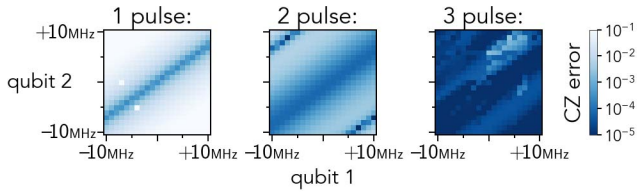
Table II: Optimal parking frequencies and drift tolerance for $Rz(\phi)$ gates with $\leq 10^{-4}$ error for $N = 255$.

Parking frequency (GHz)	Drift (GHz) for error $\leq 10^{-4}$
6.21286	$\pm$0.01282
5.02978	$\pm$0.01049
4.14238	$\pm$0.00820

For *DigiQ_opt*, calibrating gate decomposition means finding a new set of delay intervals such that the target operation is approximated by a sequence $Rz(\phi_L)U_{bs}...Rz(\phi_1)U_{bs}Rz(\phi_0)$ (where the final $Rz(\phi_L)$ is absorbed into a subsequent gate). We choose sets of delays holistically for each gate, numerically searching for the best combination of the available delays to implement that gate. We find that $L \leq 2$ is sufficient for most gates, but a subset of gates nearing π rotations around the x or y axis (e.g., Pauli X and Y operations) need $L = 3$ to obtain high fidelity on qubits with the most significant drift (whereas in the ideal case (i.e., $U_{bs} = Ry(\frac{\pi}{2})$), $L \leq 2$ is enough for all single-qubit gates).

Two factors affect performance of the *DigiQ_opt* decomposition. First, the set of available $Rz(\phi)$ rotations is highly dependent on qubit frequency, which determines how well the $N + 1$ possible delay values cover the unit circle. In the ideal case, the $N + 1$ phases are equally spaced around the unit circle, and any $Rz(\phi)$ can be approximated to within $\pi/(N + 1)$ radians; in this case we find that $N = 255$ is sufficient for error $\epsilon \leq 0.25 \cdot 10^{-4}$. We choose target frequencies with the highest tolerance for variation, as measured by the width of the interval in which any ϕ can be approximated with $< 10^{-4}$ error using one of the available delays. These optimal parking frequencies and their drift tolerance are shown for $N = 255$ in Table II. Second, frequency variations can limit SIMD scheduling. For example, if a circuit calls for the same gate to be applied to many qubits, these may still require unique delay values when the decomposition of that gate differs for each qubit. However, we can increase parallelism by allowing a small error margin when choosing delay values: often, multiple sets of delays will approximate the same operation with nearly equal error, so we can choose the one with lowest cost in terms of serialization.

For *DigiQ_min*, we decompose arbitrary single-qubit gates into sequences of discrete, qubit-specific basis operations. We use a brute-force search algorithm to find the optimal decomposition of single-qubit gates for each qubit, working with the full six-level representation of the unitary basis operations so that the decomposition accounts for and mitigates leakage resulting from each basis operation.



(a) (b) (c)

Figure 7: CZ gate error as a function of frequency drift, assuming 1, 2, or 3 U_{qq} operations and ideal single-qubit gates.

B. Calibrating two-qubit gates

The implementation of CZ gates using flux-tunable transmons requires the shape and duration of each current pulse to be carefully calibrated to the precise qubit frequencies and hardware parameters. On a small system with MIMD control, these pulses can be individually calibrated for each pair of coupled qubits to account for variation and drift. Without this fine-grained control, we are instead left with a unique 2-qubit operation U_{qq} for each coupled pair of qubits. Here, we argue that we can instead compose CZ gates for each pair of qubits in *DigiQ* using pair-specific sequences of U_{qq} operations and single-qubit gates, again relegating calibration to software.

We can compute the unitary evolution U_{qq} for a pair of qubits using physical simulations of the empirical current waveform described in Sec. IV-A3. Starting with the nominal pre-drift frequencies of 6.21286 and 4.14238 GHz from Table II, we vary each qubit's frequency and compare the resulting U_{qq} to the target CZ operation. We compute unitary evolution by numerically integrating the Schrödinger equation using well-understood Hamiltonian models of pairs of capacitively-coupled flux-tunable asymmetric transmons [25]. We assume that each transmon has an anharmonicity of 250 MHz, and the capacitive coupling strength is 10 MHz.

In Fig. 7(a), we show the expected minimum error when implementing a CZ gate using a single U_{qq} pulse as a function of each qubit's drift, allowing for arbitrary single-qubit gates before and after the pulse. At the ideal qubit frequencies, we find an expected error of $\epsilon = 3 \cdot 10^{-4}$. This error grows rapidly as the frequency difference between qubits drifts. In Fig. 7(b) and (c), we show the gate error after compiling CZ gates into sequences of 2 or 3 U_{qq} operations and intermediary single-qubit gates, similar to the "echo" sequences described in [47], [48] but with single-qubit gates obtained via numerical optimization. Assuming ideal single-qubit gates, we find that 3 U_{qq} operations are sufficient to achieve $\epsilon \leq 10^{-4}$ over a wide range of frequencies. In Sec. VI-B2, we report empirical gate errors after single-qubit gates are decomposed for either *DigiQ_opt* or *DigiQ_min*.

Table III: The library of RSFQ cells and corresponding characteristics used for synthesis.

Cell	Area (μm^2)	JJ Count	Delay (ps)
AND2	3500	16	8.4
OR2	3500	14	6.1
XOR2	3500	18	5.8
NOT	3500	12	13.2
DRO DFF	3000	11	6.2
NDRO DFF	4500	18	9.3
Splitter	2000	6	7.1

VI. Methodology and Results

In this section, we present hardware synthesis results of *DigiQ*, followed by an analysis of its algorithmic performance.

A. Hardware results of DigiQ

We use detailed SFQ synthesis tools [17], [18], [49]–[51] to synthesize, map, and finally calculate the power, area, and delay values. The employed synthesis and technology mapping flow is as follows: first, technology-independent optimizations including balanced factorization and rewriting [49] are performed on the input circuit, then the circuit is mapped using a path balancing technology mapping algorithm [17] and fully path balanced [51]. Next, a standard retiming algorithm similar to [52] is employed to further reduce the total memory element count. Finally, the memory elements (e.g., latches) are replaced with SFQ DRO DFFs, and splitters are inserted at the output of gates with more than one fanout.

Rapid Single Flux Quantum (RSFQ) logic family [12] is used in this paper, and the library of cells is listed in Table III. For wiring, we use Josephson Transmission Line (JTL) and Passive Transmission Line (PTL). JTL is used for short connection for cells next to each other and its delay is ∼1.5-2 ps. PTL is used for transmitting SFQ pulses from one logic gate to another.

1) Validity of our synthesis results: We have the physical layouts for all cells, including wires and logic cells. Our RSFQ power, area and delay numbers are obtained post-layout based on accurate extraction of the gate layouts and passive transmission lines and subsequently simulated using WRSpice [20] and JSim [21]. The simulation results are thus highly accurate and reliable. The SFQ library cells have been validated and their power/timing values calibrated against manufactured test chips done in the MIT Lincoln lab's SFQ5ee process node [53]. The synthesis, place&route, modeling/simulation, and formal verification tools have been used to design and formally/simulatively verify tens of SFQ circuits ranging from individual data path modules and register files to major building blocks of the RISC-V Sodor core [16], [54], [55]. In addition, prior work validated comparable SFQ tools and simulators with SFQ chip fabrication [14].

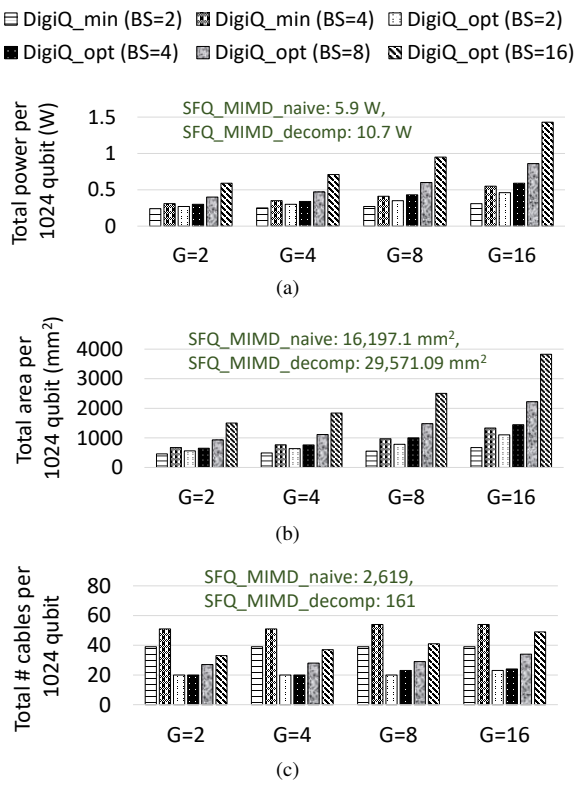


Figure 8: Power (a), area (b), and cable count (c) results of *DigiQ_min* and *DigiQ_opt* architectures. *SFQ_MIMD_naive* and *SFQ_MIMD_decomp* results are shown for comparison.

2) Delay results: Our synthesis results show that the worst stage delay in *DigiQ* is 34.5 ps, which determines the maximum clock frequency that our SFQ chip can work with. We choose 40 ps as our SFQ chip clock period (similar to [9]), thus the bit period in our SFQ bitstreams is 40 ps.

3) Power and area results: Fig. 8(a) and Fig. 8(b) show the total power and area of *DigiQ*, respectively. We present the results for different BS and G values.

As mentioned in Sec. IV-A1, in *DigiQ_opt*, only BS distinct delays are available every controller cycle for single-qubit gates, and that can potentially lead to quantum gate serialization inside a group. One solution to mitigate this serialization is to increase the value of BS, which would increase the hardware cost of quantum controller modules as they would need more complex SFQ-based multiplexers to select one out of a larger number of delayed bitstreams (see Fig. 5). Another solution to mitigate the serialization is to increase the G value. This solution decreases the number of qubits per each group which means less congestion and less serialization, while not increasing the hardware cost of gate controller modules. Thus, at the design time, we expected that among all the designs with the same $BS * G$, the ones with higher G value have lower hardware cost as they need less complex quantum controller modules. However, our synthesis results surprised us. As shown in Fig. 8, the hardware cost of the designs with the same $BS * G$ value does not differ significantly. The reason is that increasing the G value also increases the overall hardware cost due to an increase in the number of SFQ bitstream generators (see Fig. 5). Given our synthesis results, we conclude that keeping the G value small and increasing the BS value is preferred because it provides more flexibility in allocating delayed bitstreams to qubits. Smaller G values are preferred for *DigiQ_min* as well, as increasing the G value increases the hardware cost with no significant algorithmic advantage. The smallest G value or the maximum number of qubits in a group is determined by the area of one SFQ chip; if we cannot fit a group on one chip, we need to use multiple chips and replicate the bitstreams on each chip to avoid long distance communications, which is equivalent to dividing the qubits into multiple groups. Our results show that for 1024-qubit scale, we can fit all the designs with $G = 2$ on at most two SFQ chips (one per group), thus we use $G = 2$ for 1024-qubit benchmarks in Sec. VI-B.

Our results show that even our largest designs can operate within the power budget of typical dilution refrigerators at 4 K stage (i.e., a few watts [7], [10], [28]), and can be implemented on only a few SFQ chips at 1,024-qubit scale; clock synchronization between the SFQ chips is done using SFQ phase locked loops (PLLs) [56]. We replicate the 1,024-qubit design in order to scale to larger number of qubits (note that replicating the 1,024-qubit design naturally increases the number of groups). *DigiQ_min*(BS=2) has the lowest hardware cost and highest scalability (>42,000 qubits given 10 W power budget [7]). Increasing the BS value in *DigiQ_min* increases the hardware cost, but also can potentially increase the algorithmic performance (we see diminishing returns after $BS = 4$). The scalability of *DigiQ_opt* is also high, allowing >25,000 qubits (>17,000 qubits) for $BS = 8$ ($BS = 16$).

4) SFQ storage and Cable count results: *DigiQ_min* stores BS bitstreams per group and each bitstream has ≤300 bits. *DigiQ_opt* stores one bitstream with ≤300 bits per group which is delayed by BS different values at each controller cycle (see Sec. IV). With both designs, for each qubit at each controller cycle, we need enough control bits from room temperature to determine whether to apply one of the BS distinct gates, start/stop performing two-qubit gates, or perform no operation. For *DigiQ_opt*, an additional BS delay values for each group at each cycle are needed; since we have 256 possible delay values, each delay value requires $\log_2 256 = 8$ bits.

Fig. 8(c) shows the number of required cables to send *Go*, *Valid*, *Load*, *BS_sel*, *1q_sel*, and *2q_sel* bits from the room temperature in one controller cycle given 10 Gbps return-to-zero (RZ) cables [12]. We calculate the number of data cables assuming a minimum controller clock period of 9 ns for *DigiQ_min* and 9 ns + 10.2 ns for *DigiQ_opt* (10.2 ns corresponds to 255 delay cycles) – we need

Table IV: NISQ benchmark algorithms.

QGAN	Quantum generative adversarial learning network [59]
Ising	Linear Ising model spin chain simulation [60]
BV	1024-bit Bernstein-Varzirani algorithm [61]
Add1	256-bit ripple-carry adder [62]
Add2	256-bit parallel carry-lookahead adder [63]
Sqrt10	10-bit square root via Grover search [64]–[66]

enough data cables to send one set of control bits from the room temperature within one controller cycle (see Sec. IV-B). *DigiQ_min*(G=2,BS=2) requires only 39 cables per 1,024 qubits (52.5X less than a microwave-based quantum computer with 2 cables per qubit, 1 drive line and 1 flux line [3]). *DigiQ_opt* requires just 33 cables per 1,024 qubits in $G = 2$, $BS = 16$ design. *DigiQ_min*'s high cable count relative to most *DigiQ_opt* configurations is due to its shorter controller cycle.

B. Algorithmic performance results of DigiQ

In order to study the algorithmic impacts of *DigiQ*, we compile a common set of NISQ benchmarks for each design. The benchmarks chosen are described in Table IV, and together represent a diverse sample of common circuit formulations. Benchmark circuits are algorithmically generated and mapped to a 32×32 square grid via SWAP-gate insertion using the stochastic transpiler pass packaged with Qiskit Terra [57]. Each circuit is then decomposed into CZ and single-qubit gates, and a crosstalk-aware scheduling pass [58] is used to sort and group commuting two-qubit gates which can be executed simultaneously without interference.

To model frequency variation, each qubit is modeled as an asymmetric transmon with $\sigma = 0.2\%$ variability in each of its Josephson energies (sampled from a normal distribution). At our target frequencies, this corresponds to about ± 6 MHz fluctuation in oscillation frequency, in keeping with design targets for future superconducting systems [67]. Hardware variability is considered with the addition of a $\sigma = 1\%$ error to the output of each current generator. As in Sec. V, we use these sampled hardware parameters to physically simulate each basis operation on each qubit or qubit pair (similar to prior work [9], [46]). We then use the resulting unitary basis operations to decompose each gate in the benchmark circuits. Gate errors in Sec. VI-B2 are determined by computing the overlap between the decomposed and target gates.

For *DigiQ_opt*, we use a 20.32 ns controller cycle time, comprising 10.12 ns for SFQ bitstreams and 255 delay cycles. The CZ gate time is 60 ns (determined from the analysis in Sec. V-B), which expands over three controller cycles. Single-qubit gates are expanded into at most three U_{bs} pulses (see Sec. V-A). For *DigiQ_min*, single-qubit gate times for the 6.21286 and 4.14238 GHz qubit frequencies are respectively 10.12 ns and 9.00 ns, again with a 60 ns CZ gate time. We decompose single-qubit gates until the approximation error falls below 10^{-4}, up to a maximum depth of 28 (at which point we find that only about 1% of gates have errors above 10^{-4}). There is no feedback loop in our benchmarks, thus we do not consider the communication latency from outside the fridge.

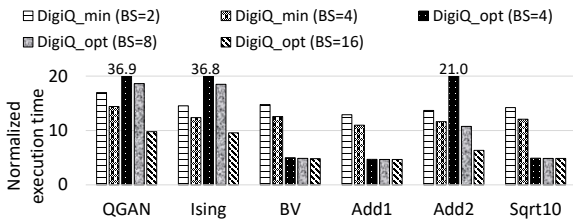


Figure 9: *DigiQ* quantum circuit execution time normalized to the Impossible MIMD system.

1) Circuit execution time: Fig. 9 shows the execution time results for *DigiQ* with 1024 qubits, normalized to an *Impossible MIMD* system which is assumed to have the same gate times as *DigiQ* (which are also similar to those of recent microwave-based systems [3], [42]) but with unlimited parallelism. We emphasize that the MIMD system is impossible at large scales (see Sec. II); we compare our results with the Impossible MIMD system just to quantify the serialization cost of realizing a controller design that is feasible at large scales, and to give readers a sense of how *DigiQ* would compare to today's prototypes if they did not have scalability issues. Both *DigiQ_opt* and *DigiQ_min* have some overhead in terms of execution time compared to the Impossible MIMD system (*DigiQ_opt* due to serialization and *DigiQ_min* due to longer gate decompositions). The performance of *DigiQ_opt* increases by increasing the BS value, as expected, with $BS = 16$ providing the best performance. This difference is minimal for benchmarks without significant parallelism (BV, Add1, Sqrt10). For $BS = 16$, the execution time is only $\leq 2X$ longer than what we would be able to achieve if *DigiQ_opt* could be built with zero quantum gate serialization (i.e., $BS = \infty$). In *DigiQ_min*, increasing BS from 2 to 4 reduces the depth of single-qubit gate decompositions by roughly half. However, the benefit is limited beyond $BS = 4$ due to the dominance of CZ gate decompositions which do not benefit substantially from the larger single-qubit gate set. *DigiQ_min* performs similarly to *DigiQ_opt*(BS=8) for the benchmarks with more parallelism (QGAN, Ising, Add2), in which *DigiQ_opt* experiences more gate serialization. For the remaining benchmarks, *DigiQ_min*'s long gate decomposition leads to worse performance than any *DigiQ_opt* configuration. Compared to Impossible MIMD system, *DigiQ_opt*(BS=16) is 4.7X-9.8X worse in terms of circuit execution time, while *DigiQ_min*(BS=4) is 11.0X-14.4X worse.

2) Gate/circuit error: We estimate the overall circuit error due to gate decomposition by taking the product of

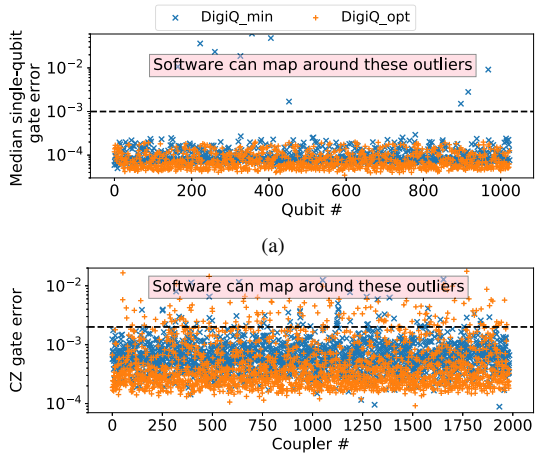


Figure 10: (a) Median single-qubit gate error on *DigiQ_opt* (BS=8) and *DigiQ_min* (BS=2) with 1024 qubits (well representative of other configurations); (b) CZ gate error on each qubit pair. Software can map around the outliers using the noise-adaptive mapping techniques [68].

the errors of each of its gates. For both designs, typical gate error varies between qubits with different frequency drifts. Fig. 10(a) shows the median error of single-qubit gates (evaluated across all gates in all benchmarks) for each qubit on *DigiQ_opt*(BS=8) and *DigiQ_min*(BS=2) (the same conclusions hold for other BS values). We find that *DigiQ_opt* exhibits higher variability between qubits, whereas *DigiQ_min* is generally more stable but has a small number of especially bad outlier qubits (where the frequency drift brings the nominal T gate very close to the identity, resulting in a poor single-qubit gate set). Similar to microwave-based systems, we can map around these outlier cases in software using noise-adaptive mapping techniques [68]. The CZ gate error (Fig. 10(b)) is driven both by the decomposition of CZ into U_{qq} and single-qubit gates, and by the error with which we can decompose those single-qubit gates on each architecture. The CZ error is $>$0.002 for 3% and 7% of qubit pairs in *DigiQ_min* and *DigiQ_opt*, respectively. Note that the CZ error would be $>$0.002 for 84% of qubit pairs if we did not use our software calibration techniques. Our results show that with software calibration *DigiQ* can achieve similar gate error to that of hardware-calibrated microwave-based gates [46].

VII. Conclusions and Future Work

Large-scale quantum computers are essential in order to perform many useful quantum algorithms. However, superconducting quantum computer prototypes have severe scalability issues due to the massive overhead of generating and routing control pulses from room temperature to quantum chip inside the dilution refrigerator. In this work, we present *DigiQ*, the first system-level design of a digital SFQ-based quantum controller architecture to maximize scalability. We provide architecture guidelines to design large-scale SFQ-based controllers by taking into consideration the opportunities (efficient on-chip broadcast and ultra-fast clock) and challenges (lack of dense memory/logic) of SFQ. Our study shows that we must deploy a SIMD architecture to operate within the tight power/area budget of dilution refrigerators, however, SIMD also introduces new challenges in terms of control pulse calibration in NISQ machines with imperfect qubits. We propose novel software-solution to address these calibration challenges efficiently, and show that software plays a key role in ensuring good quantum algorithmic performance at large scales. We fully characterize *DigiQ* controller hardware using state-of-the-art/validated SFQ synthesis tools. We also show the effectiveness of *DigiQ* in terms of quantum algorithmic performance under a variety of NISQ algorithms. We model dominant sources of systematic error in our evaluations and show that *DigiQ* has similar fidelity to microwave-based systems under the same models. Our analysis shows that *DigiQ* is a realistic path forward to realize large scale quantum computers, and we hope the promising results of this paper motivate experimentalists to further explore SFQ-based controllers.

Going forward, the fidelity of quantum gates/circuits can further be improved by performing further research at both hardware and software levels. First, the main bottleneck in increasing the gate fidelity in *DigiQ* is frequency drift of imperfect qubits, thus developing better qubits with less drift can increase the fidelity significantly. Second, decreasing the power and area consumption of SFQ circuits would allow the realization of more complex SFQ-based controllers that can potentially perform hardware calibration, which combined with our software calibration can further increase the fidelity of quantum gates/circuits. Third, further research at the software level can potentially lead to more robust decompositions with higher fidelity.

Finally, we would like to mention that the ideas proposed in this paper such as SIMD design and software calibration can potentially increase the scalability of today's Cryo-CMOS controllers as well. However, further research on such controllers based on Cryo-CMOS is needed. First, although SIMD can reduce the cost of on-chip storage in Cryo-CMOS, novel approaches are needed to share/broadcast the instructions (i.e., amplitudes/phases of microwave pulses) on the Cryo-CMOS chip with low cost to obtain overall cost reduction. Second, Cryo-CMOS has denser memory/logic, but much higher power, which would result in significantly different set of design tradeoffs. Third, *DigiQ* relies on ultra-fast clock in SFQ (i.e., in the order of ps), and Cryo-CMOS requires different architectures and gate implementations as it works with clocks in the order of ns.

ACKNOWLEDGMENT

This work is funded in part by EPiQC, an NSF Expedition in Computing, under grants CCF-1730082/1730449; in part by STAQ under grant NSF Phy-1818914; in part by DOE grants DE-SC0020289 and DE-SC0020331; and in part by NSF OMA-2016136 and the Q-NEXT DOE NQI Center. This work was completed in part with resources provided by the University of Chicago's Research Computing Center.

REFERENCES

[1] M. Brink, J. M. Chow, J. Hertzberg, E. Magesan, and S. Rosenblatt, "Device challenges for near term superconducting quantum processors: frequency collisions," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2018, pp. 6.1.1–6.1.3.

[2] G. Li, Y. Ding, and Y. Xie, "Towards efficient superconducting quantum processor architecture design," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Mar. 2020, pp. 1031–1045.

[3] F. Arute, K. Arya, R. Babbush, D. Bacon *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[4] J. Kelly, "A preview of Bristlecone, Google's new quantum processor," *Google Research Blog*, Mar. 2018. [Online]. Available: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html

[5] M. Steffen, D. P. DiVincenzo, J. M. Chow, T. N. Theis, and M. B. Ketchen, "Quantum computing: an IBM perspective," *IBM Journal of Research and Development*, vol. 55, no. 5, pp. 13:1–13:11, 2011.

[6] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren *et al.*, "An experimental microarchitecture for a superconducting quantum processor," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Oct. 2017, pp. 813–825.

[7] R. McDermott, M. Vavilov, B. Plourde, F. Wilhelm *et al.*, "Quantum–classical interface based on single flux quantum digital logic," *Quantum Science and Technology*, vol. 3, no. 2, 2018.

[8] E. Leonard, M. A. Beck, J. Nelson, B. Christensen *et al.*, "Digital coherent control of a superconducting qubit," *Phys. Rev. Applied*, vol. 11, no. 1, Jan. 2019.

[9] K. Li, R. McDermott, and M. G. Vavilov, "Hardware-efficient qubit control with single-flux-quantum pulse sequences," *Phys. Rev. Applied*, vol. 12, no. 1, Jul. 2019.

[10] J. P. G. Van Dijk, B. Patra, S. Subramanian, X. Xue *et al.*, "A scalable cryo-CMOS controller for the wideband frequency-multiplexed control of spin qubits and transmons," *IEEE J. Solid-State Circuits*, vol. 55, no. 11, pp. 2930–2946, 2020.

[11] D. Kirichenko, S. Sarwana, and A. Kirichenko, "Zero static power dissipation biasing of RSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 776–779, 2011.

[12] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, 1991.

[13] P. J. Liebermann and F. K. Wilhelm, "Optimal qubit control using single-flux quantum pulses," *Phys. Rev. Applied*, vol. 6, no. 2, Aug. 2016.

[14] K. Ishida, I. Byun, I. Nagaoka, K. Fukumitsu *et al.*, "SuperNPU: An extremely fast neural processing unit using superconducting logic devices," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Oct. 2020, pp. 58–72.

[15] J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes *et al.*, "Compiler management of communication and parallelism for quantum computation," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Mar. 2015, pp. 445–456.

[16] C. J. Fourie, K. Jackman, M. M. Botha, S. Razmkhah *et al.*, "ColdFlux superconducting EDA and TCAD tools project: Overview and progress," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Jan. 2019.

[17] G. Pasandi and M. Pedram, "A dynamic programming-based path balancing technology mapping algorithm targeting area minimization," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des. (ICCAD)*, Nov. 2019.

[18] G. Pasandi and M. Pedram, "An efficient pipelined architecture for superconducting single flux quantum logic circuits utilizing dual clocks," *IEEE Trans. Appl. Supercond.*, vol. 30, no. 2, Nov. 2019.

[19] S. N. Shahsavani, T.-R. Lin, A. Shafaei, C. J. Fourie, and M. Pedram, "An integrated row-based cell placement and interconnect synthesis tool for large SFQ logic circuits," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, 2017.

[20] Whiteley Research Inc., "WRspice circuit simulator," 2017. [Online]. Available: http://www.wrcad.com/wrspice.html

[21] E. S. Fang and T. Van Duzer, "A Josephson integrated circuit simulator (JSIM) for superconductive electronics application," in *Extended Abstracts of 1989 Int. Supercond. Electronics Conf. (ISEC)*, Jun. 1989, pp. 407–410.

[22] K. Bertels, A. Sarkar, and I. Ashraf, "Quantum computing–from NISQ to PISQ," *IEEE Micro*, vol. 41, pp. 24–32, Sep. 2021.

[23] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.

[24] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck *et al.*, "Charge-insensitive qubit design derived from the Cooper pair box," *Phys. Rev. A*, vol. 76, no. 4, Oct. 2007.

[25] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando *et al.*, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, 021318, Jun. 2019.

[26] M. D. Hutchings, J. B. Hertzberg, Y. Liu, N. T. Bronn *et al.*, "Tunable superconducting qubits with flux-independent coherence," *Phys. Rev. Applied*, vol. 8, no. 4, Oct. 2017.

[27] K. R. Brown, J. Kim, and C. Monroe, "Co-designing a scalable quantum computer with trapped atomic ions," *npj Quantum Info.*, vol. 2, Nov. 2016.

[28] J. M. Hornibrook, J. I. Colless, I. D. Conway Lamb, S. J. Pauka *et al.*, "Cryogenic control architecture for large-scale quantum computing," *Phys. Rev. Applied*, vol. 3, no. 2, Feb. 2015.

[29] Q. P. Herr, A. Y. Herr, O. T. Oberg, and A. G. Ioannidis, "Ultra-low-power superconductor logic," *Journal of applied physics*, vol. 109, no. 10, 2011.

[30] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Supercond. Sci. Technol.*, vol. 26, no. 3, 2013.

[31] M. H. Volkmann, A. Sahu, C. J. Fourie, and O. A. Mukhanov, "Experimental investigation of energy-efficient digital circuits based on eSFQ logic," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, Jan. 2013.

[32] O. A. Mukhanov, "Energy-efficient single flux quantum technology," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 760–769, Jun. 2011.

[33] R. McDermott and M. G. Vavilov, "Accurate qubit control with single flux quantum pulses," *Phys. Rev. Applied*, vol. 2, no. 1, Jul. 2014.

[34] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, "Taming the instruction bandwidth of quantum computers via hardware-managed error correction," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Oct. 2017, pp. 679–691.

[35] M. Dalgaard, F. Motzoi, J. J. Sørensen, and J. Sherson, "Global optimization of quantum dynamics with AlphaZero deep exploration," *npj Quantum Info.*, vol. 6, Jan. 2020.

[36] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding *et al.*, "NISQ+: Boosting quantum computing power by approximating quantum error correction," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA '20)*, May 2020, pp. 556–569.

[37] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys*, vol. 52, no. 6, pp. 1191–1249, Dec. 1997.

[38] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Info. Comput.*, vol. 6, no. 1, pp. 81–95, Jan. 2006.

[39] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, "Efficient Z gates for quantum computing," *Phys. Rev. A*, vol. 96, no. 2, Aug. 2017.

[40] V. Kaplunenko, M. Khabipov, V. Koshelets, K. Likharev *et al.*, "Experimental study of the RSFQ logic elements," *IEEE Trans. Magn.*, vol. 25, no. 2, pp. 861–864, Mar. 1989.

[41] M. R. Jokar, R. Rines, and F. T. Chong, "Practical implications of SFQ-based two-qubit gates," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2021, pp. 402–412.

[42] 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. (MICRO)*, Oct. 2020, pp. 186–200.

[43] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Apr. 2019, pp. 987–999.

[44] M. A. Nielsen, "A simple formula for the average gate fidelity of a quantum dynamical operation," *Phys. Lett. A*, vol. 303, no. 4, pp. 249–252, Oct. 2002.

[45] J. Ghosh, "A note on the measures of process fidelity for non-unitary quantum operations," *arXiv preprint*, Nov. 2011, arXiv:1111.2478.

[46] N. Leung, M. Abdelhafez, J. Koch, and D. Schuster, "Speedup for quantum optimal control from automatic differentiation based on graphics processing units," *Phys. Rev. A*, vol. 95, no. 4, Apr. 2017.

[47] S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," *Phys. Rev. A*, vol. 93, no. 6, Jun. 2016.

[48] A. D. Córcoles, J. M. Gambetta, J. M. Chow, J. A. Smolin *et al.*, "Process verification of two-qubit quantum gates by randomized benchmarking," *Phys. Rev. A*, vol. 87, no. 3, Mar. 2013.

[49] G. Pasandi and M. Pedram, "Balanced factorization and rewriting algorithms for synthesizing single flux quantum logic circuits," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 183–188.

[50] G. Pasandi, A. Shafaei, and M. Pedram, "SFQmap: A technology mapping tool for single flux quantum logic circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018.

[51] G. Pasandi and M. Pedram, "PBMap: a path balancing technology mapping algorithm for single flux quantum logic circuits," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 4, Nov. 2019.

[52] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.

[53] S. K. Tolpygo, V. Bolkhovsky, T. J. Weir, A. Wynn *et al.*, "Advanced fabrication processes for superconducting very large-scale integrated circuits," *IEEE Trans. Appl. Supercond.*, vol. 26, no. 3, 2016.

[54] H. Cong, M. Li, and M. Pedram, "An 8-bit multiplier using single-stage full adder cell in single flux quantum circuit technology," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 6, Jun. 2021.

[55] UC Berkeley Architecture Research, "The sodor processor collection." [Online]. Available: https://github.com/ucb-bar/riscv-sodor

[56] D. K. Brock and M. S. Pambianchi, "A 50 GHz monolithic RSFQ digital phase locked loop," in *Proc. IEEE MTT-S Int. Microw. Symp. Digest (Cat. No. 00CH37017)*, vol. 1, Jun. 2000, pp. 353–356.

[57] M. S. ANIS, H. Abraham, AduOffei, R. Agarwal *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[58] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Mar. 2020, pp. 1001–1016.

[59] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Phys. Rev. Lett.*, vol. 121, no. 4, Jul. 2018.

[60] R. Barends, A. Shabani, L. Lamata, J. Kelly *et al.*, "Digitized adiabatic quantum computing with a superconducting circuit," *Nature*, vol. 534, no. 7606, pp. 222–226, 2016.

[61] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1411–1473, Oct. 1997.

[62] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint*, Oct. 2004, arXiv:quant-ph/0410184.

[63] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Info. Comput.*, vol. 6, no. 4&5, pp. 351–369, Jul. 2006.

[64] A. Javadi-Abhari, "Towards a scalable software stack for resource estimation and optimization in general-purpose quantum computers," Ph.D. dissertation, Princeton University, 2017.

[65] P. Murali, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Formal constraint-based compilation for noisy intermediate-scale quantum systems," *Microprocessors and Microsystems*, vol. 66, pp. 102–112, Mar. 2019.

[66] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory of Computing (STOC '96)*, Jul. 1996, pp. 212–219.

[67] J. B. Hertzberg, E. J. Zhang, S. Rosenblatt, E. Magesan *et al.*, "Laser-annealing Josephson junctions for yielding scaled-up superconducting quantum processors," *npj Quantum Info.*, vol. 7, Aug. 2021.

[68] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Apr. 2019, pp. 1015–1029.