A Survey of Quantum Control Architecture

A. Brassel, J. Kuo, J. Young

October 2019

1 Introduction

Classical computers store and manipulate information in the form of bits, units of binary data that are either zero or one. In constrast, quantum computation exploits quantum properties such as entanglement and superposition by working with qubits, two-dimensional vectors that can be some combination of both zero and one. Complex vectors created by quantum algorithms are then measured, "collapsing" the qubit into either zero or one. The result of this measurement may then be used classically. This allows quantum computers to work with exponentially many classical bits worth of information simultaneously and describe more complex relationships between information before making a measurement. Furthermore, whereas classical logic gates define a limited amount of fundamental operations, all unitary matrices represent valid quantum gates, allowing for infinitely many operations, even on a single qubit.

With these tools, quantum computers can carry out algorithms and solve problems that are classically infeasible, a property known as quantum supremacy. However, making use of quantum supremacy necessitates the construction of highly optimized quantum computers with ever-increasing amounts of qubits. The new tools available to quantum computers are accompanied by new architectural demands in the construction of a quantum processor. Scaling up current quantum processor designs is an ever-evolving challenge in quantum computation.

2 Quantum Control

Classical processors are built to support a fundamental set of instructions that allow programmatic control of the processor. Compilers break down code written in higher-level languages into assembly, a low-level representation consisting of instructions that are implemented on the underlying hardware. Within the processor, complex instructions correspond to microprograms made up of simpler instructions. Basic instructions can be directly converted to binary data and loaded onto various registers and functional blocks, flipping the electronic switches that control command execution. This entire process is done digitally through the use of wires, transistors, resistors, and other electrical components. Advances

in traditional computing technology allow the execution of these commands to occur at an extraordinary rate, typically on the order of several billion operations per second.

However, delivering instructions to a quantum processor is a vastly more complicated process. Modern models of quantum computation treat a quantum processor as a performance-accelerating coprocessor, adding a quantum coprocessor to a classical CPU to grant it significant speedups for specific problems. This quantum coprocessor is therefore subject to classical control and must have digital input and output. However, the qubits in a quantum computer are built out of quantum systems such as trapped ions, trapped electrons, superconducting Josephson junctions, photons, quantum dots, and more.[7] The information stored in these qubits is represented by the physical quantum state the system is in. In most models, the quantum coprocessor is responsible for translating these digital instructions into physical control of a quantum system, measuring the physical properties of the system, and reporting the results of these measurements digitally [7].

In algorithm design, the low-level requirements for controlling a quantum processor are often abstracted away by compiling algorithms to a gate-level quantum instruction set. However, in sharp contrast to classical instruction sets, most quantum instruction sets are "virtual." In particular, most are low-level intermediate representations comprised of assembly-like code specifying gate-level operations that do not necessarily correspond to physical implementation on underlying hardware. This abstraction masks many of the underlying challenges of interacting with a quantum system. Because quantum states are real-world physical properties that are not already classically digitized, each qubit requires digital-to-analog and analog-to-digital converters (DAC and ADC, respectively) to control the underlying quantum system [1] and measure its state. In particular, a quantum coprocessor must generate and apply physical signals called control pulses to change the state of a qubit and probe the physical properties of the system. The result of this measurement of the quantum system must be digitized and interpreted as a single bit of information. The structure and implementation of these control pulses at the lowest level is technology-dependent, and the amount of computational and hardware resources these systems require is high enough to dramatically limit scalability [9].

3 Scalability Concerns

There are several factors that contribute to the hardware cost of quantum control. First of all, DAC and ADC can have significant hardware and time cost. For many qubit designs, control pulses are nontrivial sequences of analog output, and these sequences consume resources to store, compute, or generate [2]. Furthermore, reading in physical properties from a quantum system requires dedicated hardware to sample and sequence a digital representation of some physical signal. Interpreting this digitized measurement signal, a process known as measurement discrimination, often requires complex computation. For example, measurement discrimination in superconducting qubit architectures requires integration [1], a highly specialized and intensive hardware task. This leads to additional resource consumption, hampering the feedback and control process.

Quantum error correction also adds significant overhead to control costs. Since a quantum

system cannot be perfectly isolated from the environment, its state cannot be perfectly controlled. Unwanted deviations from a qubit state, known as decoherence [1], can be caused by any number of couplings with the environment. In these cases, the environment manipulates the state of the quantum system in some unpredictable way, causing unpredictable errors. The decoherence time of a quantum system, a representation of how long a qubit implemented with the associated quantum system can store information reliably, provides a direct limit on allotted time for a computation. Furthermore, since decoherence is probabilistic, "reliable" storage simply has a low error rate prior to the decoherence time of the system, often in the realm of 0.1% [1]. This stands in sharp contrast to classical computation, where errors occur at negligible rates and largely consist of isolated bit flip errors. While modern quantum computers cannot achieve this level of reliability, establishing error correcting schemes within a quantum processor helps to mitigate this concern.

Classically, reliable fault-tolerant computation can be easily achieved by copying bits and saving backups to detect errors[8]. However, quantumly, the well-known No Cloning Theorem guarantees that qubits cannot be copied for verification. This necessitates the implementation of more complicated quantum error correction schemes. Chief among them are error correcting codes designed to encode one qubit worth of quantum information, known as a "logical qubit," into an entangled multi-qubit system consisting of real-world "physical qubits." While this entangled system multiplies the hardware cost per logical qubit, it allows a processor to take error syndrome measurements- measurements of qubits within and adjacent to the entangled system that are designed to obtain diagnostic information about the system itself, serving as indicators of various errors. By taking these measurements carefully, the single logical qubit of information within the system may be protected from observation while still allowing system errors to be reported and corrected. Unfortunately, this scheme necessitates a dramatic increase in both the numbers of physical qubits in a quantum chip and the number of control pulses that must be applied to the quantum core. Furthermore, whereas algorithms may only require the application of a limited number of control pulses at any given time, error correction must be continuously performed on all qubits relevant to a given quantum program, even if they might be unmodified at that point in execution of the algorithm [1].

Error correction also demands rapid classical control and measurement feedback from the quantum processor [1]. Otherwise, it is impossible to determine and execute the proper error correcting steps within the strict runtime limitations of the quantum processor. Failure to do so will lead to the accumulation of uncorrected errors, making computation results meaningless. Rapid feedback and control systems are also crucial for the implementation of quantum or classical control flow statements- conditionally or repeatedly executing blocks of code based on conditions. For example, in the later stages of the teleportation protocol, one party takes a measurement of some quantum state and communicates the result classically to the other party. This classical information allows the second party to perform some operations to reconstruct the teleported state. This necessitates rapid interpretation of measurement results and low-latency control pulse generation, lest the second party's qubit decohere before they are able to transform and use it.

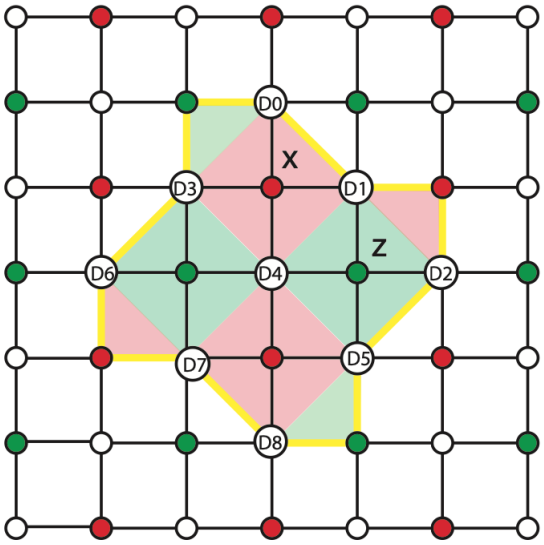

Figure 1: The ninja star error correcting code. This system uses eight data qubits and nine ancilla measurement qubits to encode one logical qubit of information. Courtesy of [1].

Even with some form of error control, computation must still be completed prior to the decoherence time of a qubit in order to obtain correct results with an acceptable probability. This means that latency is one of the major hurdles preventing the execution of more complicated algorithms on larger quantum computers. However, restrictions on instruction bandwidth limit parallelism and increase control latency. Since modern quantum processors must be kept isolated from the environment in dedicated supercooled facilities, there are extremely limited communication channels between the traditional classical processor and the quantum coprocessor. There are power, energy, and space costs associated with local data storage and data transmission, and reserving and delivering the necessary resources compromises the isolation of the quantum processor. This means that communication with the classical processor must be minimized, since parallelizing communication compromises isolation and serializing communication drives up latency [9] [10].

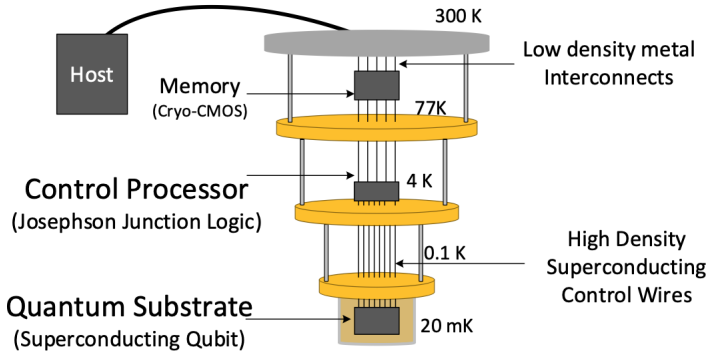

Figure 2: Diagram of an isolated cryogenic quantum processor. Different architectural components of the computer are kept at various temperatures and interconnections between layers are extremely limited. Courtesy of [9].

Low-level algorithm implementation presents another obstacle to minimizing algorithm runtime. Many modern quantum computation systems target a virtual quantum instruc-

tion set like OpenQASM for algorithm implementation. These instructions sets are used to control a theoretical quantum abstract machine, a collection of qubits upon which arbitrary quantum circuits may be run without error and subject to classical control. This toolchain implicitly assumes that a quantum processor will support any arbitrary gate necessary for the computation. However, gates represent theoretical manipulations of a quantum system that are highly non-trivial to perform. Underlying hardware generally supports only a few key quantum gates, forming a universal gate set. Classically, all possible boolean functions may be implemented using a universal gate set with relatively low complexity, but quantumly, universal gate sets are only capable of approximating any desired unitary matrix to an arbitrary degree of precision. This approximation makes no promises as to its complexity, potentially requiring untenable sequences of operations. Decomposing a desired unitary into some acceptable combination of supported gates presents a computational challenge, and since the available universal gate set is dependent on the underlying quantum system technology, the decomposition is technology-dependent as well. Since the difficulty of execution of a particular quantum gate may vary based on implementation, the underlying hardware must be considered when generating both gate-level representations and control pulses [6]. Determining the appropriate level of vertical integration necessary in the quantum compilation and execution process is a major challenge in modern quantum computation. Quantum architectures aim to divide up the tasks involved in controlling a quantum coprocessor and its underlying quantum systems and assign them to various hardware and software components in a highly optimized and scalable manner.

4 Proposed Architectural Solutions

Fu et al. [1] have identified a need for further specifying the layout of a quantum computer- in particular, describing the connections between gate-level algorithm implementation and the physical layer. Many current systems assume a superconducting qubit design and implement non-scalable control logic with low-speed feedback, further suggesting that all compilation of programs below the gate level must be technology dependent and highly customized. In contrast, Fu et al. wish to attain a greater degree of technological independence by specifying the structure of a microarchitecture responsible for control logic and managing the instruction data path. This microarchitecture consists of multiple hardware sections designed to translate gate-level instructions into precisely-timed and parallelized sequences of control pulses using a microcode and codeword approach. This approach is analogous to classical complex instructions- in this case, all quantum instructions are treated as complex instructions that must be broken down into processor-defined microprograms. Once a quantum instruction is delivered to the quantum processing unit, it undergoes symbol assignment and control flow in a Quantum Execution Controller, translation into a sequence of microoperations with a given timing in a Physical Microcode Unit, and parallelization and enqueueing of microoperations for translation into codewords and pulse sequences at a fixed latency in the Analog-Digital Interface. While synthesizing control pulses for a wide array of quantum operations can have significant hardware cost and latency, the microcode approach breaks down a quantum operation into a sequence of simpler operations. Correspondingly,

this allows storing simpler control pulses and combining them in precise sequences to form more complex operations, dramatically reducing the cost of this process. This final step is carried out in a technology-dependent quantum classical interface. Whereas this system would typically require time, storage resources, and communication bandwidth, the above optimizations and organization schemes allow major reductions in cost [1].

Fu et al. [1] further specify the presence of a Quantum Error Correction Block in parallel with the Quantum Execution Block. The QEC unit makes use of Pauli frames for error management, shifting the burden for error correction to dedicated hardware systems. A Pauli frame is an error correcting scheme that classically records the errors present in physical qubits as a combination of Pauli gates that can be later reapplied for correction. This sequence, known as a Pauli record, is made up of gates that commute, allowing a high degree of simplification and the removal of redundant corrections, massively reducing the amount of instructions that must be decoded and executed. These instructions are delivered directly to the QEX, eliminating error processing at the higher levels [1].

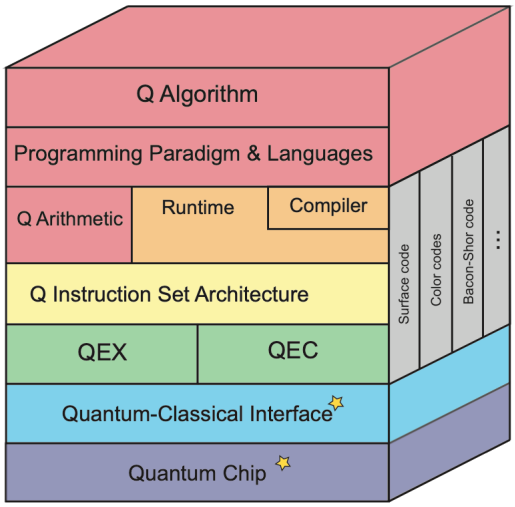

Figure 3: Quantum architecture stack as described by Fu et al., courtesy of [1].

In a follow-up experiment, Fu et al. [2] fully design QuMA, a control microarchitecture based on the above criteria. They implement a simplified version of QuMA, focusing on the conversion of codewords to control pulses with proper timing and leave error correction and high-level instruction translation to future work. They implement their microarchitecture on several Field-Programmable Gate Arrays (FPGAs), utilizing several arbitrary waveform generators to control a superconducting quantum core. They then experimentally validate QuMA by using it to conduct a standard gate characterization test known as *AllXY*. In this experiment, single-qubit gates requiring the application of precisely-timed sequences of control pulses are applied to qubits and then tested for accuracy. Their results demonstrate that QuMA is a viable system with many technology-independent components [2].

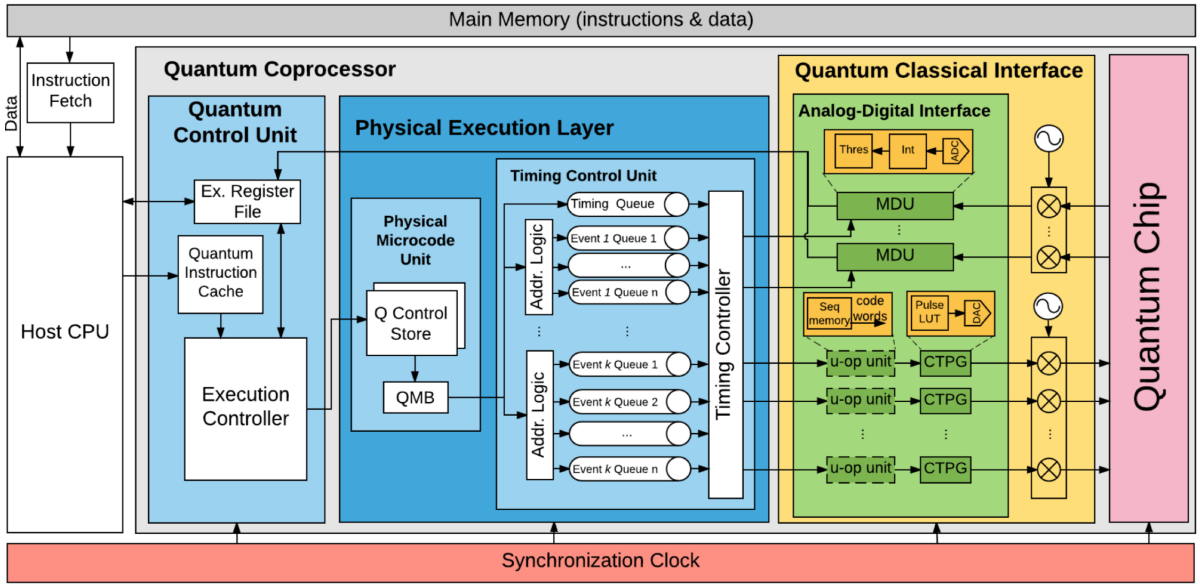

Figure 4: Experimental implementation of QuMA. The Quantum Control Unit is responsible for delivering quantum instructions to the correct clients, the physical microcode unit delivers seqeunced microinstructions to the quantum-classical interface with precise timing, and the microoperation unit translates these into codewords used to look up and trigger simplififed pulse generation. Note the exclusion of the QEC section. Courtesy of [1].

Hornibrook et al. [10] have developed the "Prime-line / Address-line" architecture to generate control pulses at a higher level, separate quantum control instructions from pulse generation, and minimize communication between supercooled and room temperature facilities. This architecture identifies several "Prime waveforms" that serve as control pulses for hardware-supported fundamental gates. These waveforms are bussed together on "Prime-lines." The "Address-line" conveys classical instructions to switching matrices, dedicated hardware systems in close proximity to the qubits they control. These matrices select a prime waveform from the bus to convert to physical pulses for qubit control. This system minimizes the energy required to generate and transmit control pulses within the quantum processor and allows classical instructions to trigger quantum gates instead of being converted to control pulses. These local switching matrices may also be tuned to customize the control pulses delivered to particular qubits, further optimizing performance. Hornibrook et al. experimentally verified this architecture by building a switching matrix into a cryogenically compatible FPGA and using it to control semiconductor quantum dot qubits. While individual switching matrices and prime waveforms are technology-dependent, the underlying principles hold regardless of platform. Many different quantum systems have compatible switching matrix designs, allowing "Prime-line / Address-line" architectures to be generally applicable [10].

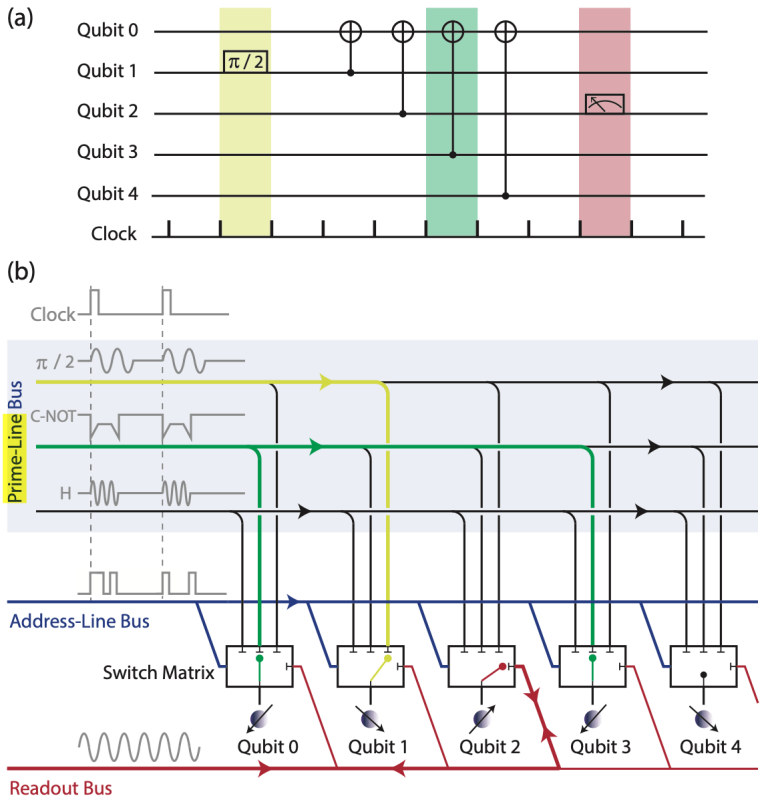

Figure 5: Prime-line multiplexing architecture. The prime waveforms are bussed together and locally multiplexed for qubit control. The switch matrices are controlled by the Address-line bus. Courtesy of [10].

Tannu et al. [9] have identified instruction bandwidth as a major scalability concern and demonstrated that 99.999% of instructions are used for error correction. To that end, they have proposed an alternative hardware-managed system for error correction known as QuEST, or Quantum Error-Correction Substrate. Similarly to QuMA, this system is designed to parallelize corrections on a given qubit and decouple error correction instructions from ordinary algorithmic instructions, enabling continuous error correction on all qubits in use without compromising the execution of standard instructions. However, QuEST employs Micro-coded Control Engines, or MCEs, to locally execute both QEC codes and logical instructions. Each MCE serves a region of the quantum processor and is controlled by a master. Tannu et al. have focused their efforts on using these MCEs to optimize the instruction and data pipelines in this system, minimizing latency and instruction bandwidth. Because QEC instructions are run continuously and don't require global synchronization, they may be recorded and delivered by the MCEs without intervention from the master. MCEs may cache logical instructions for algorithm execution, further reducing the instruction bandwidth. Through experimental simulation techniques, Tannu et al. have demonstrated that QuEST can reduce instruction bandwidth by up to eight orders of magnitude [9].

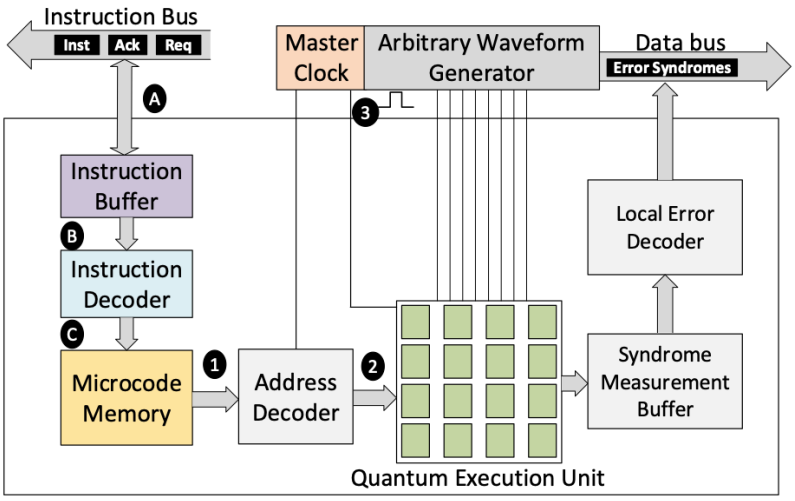

Figure 6: Example architecture of a Micro-coded control engine. Digital instructions are converted locally into microcode. The microcode memory remembers QEC instruction sequences and some logical instructions. Codewords from this unit are used to trigger the switching matrices of a primeline multiplexing architecture. Courtesy of [9].

Shi et al. [6] describe a mismatch between gate-level and hardware-level descriptions of quantum circuitry as a major scalability concern. When quantum programs are compiled to the gate level, they make simplifications and assumptions of a fundamental gate set that may not match underlying hardware. For example, underlying hardware may readily support a CNOT, SWAP, cZ, or other controlled gate, while a compiler may decompose a program into SWAP gates. In extreme examples, a compiler may represent a less common gate as a combination of more "fundamental" gates, but if the underlying hardware had direct support for the original gate, then it would need to further decompose each fundamental gate when an optimal circuit would have been trivial. This suboptimal control dramatically increases the complexity of circuitry unnecessarily. To counter this, Shi et al. propose novel compiler methods that can take advantage of quantum optimal control [6].

Quantum optimal control is an algorithmic approach that maximizes the efficiency of control pulses applied to a quantum system. For a desired qubit evolution, a sequence of control pulses are applied to the system, and recursive analysis allows the adjustment of each pulse to maximize the efficiency of the system transformation [6]. However, this is an incredibly computationally expensive procedure and is impractical for large-scale circuits. Shi et al. have extended the applications of this technique by aggregating gate-level instructions into blocks that commute and can be managed in parallel before applying optimal control processing to the aggregates. While other approaches covered previously are designed to separate high-level software processing from control pulse specification, this approach uses a much greater level of vertical integration. While it is much more computationally demanding on a compiler, this approach can achieve speedups of 5x to 10x. This allows the execution of algorithms that are too complex to run quickly and therefore reliable enough on modern quantum computers. While this heavily fine-tuned approach is likely not scalable into larger qubit regimes, it does highlight that significant optimizations may be made by providing

higher-level compilers with a greater degree of hardware information [6].

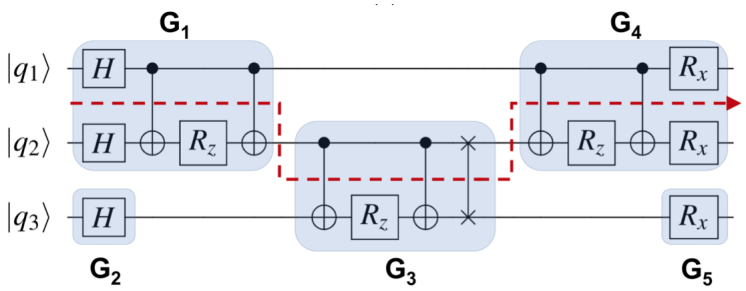

Figure 7: Example circuit broken up into aggregated sections. Within each section, the gates commute and control pulses for the section may be applied simultaneously. These aggregates may be assigned a single complex pulse. Courtesy of [6].

In summary, while most quantum programs abstract away low-level hardware control, the scalability difficulties of modern quantum computers is partially caused by challenges at this level. There have been many proposed quantum architectural changes designed to improve scalability. Fu et al. have proposed a cohesive microarchitecture designed to shift the burden of error correction to hardware, decouple error correction instructions from logical instructions, and reduce the hardware and time cost of the sequencing and execution of control pulses. Similarly, Tannu et al. have proposed a reprogrammable microarchitecture that separates logical and error correction instructions at the hardware level while massively reducing the demand on instruction bandwidth. Rather than focus on defining architectural sections for instruction translation and control pulse generation, Shi et al. have extended quantum optimal control for use with modern quantum computers by compiling gates into highly optimal aggregates that are directly converted into control pulses.

Each of these proposed solutions propose some division of labor and dedicated hardware and software systems to design scalable quantum architectures. Reaching a consensus about where these divisions should be and the appropriate level of vertical integration will be of utmost importance to enable standardized implementation and further optimization during the era of Noisy Intermediate-Scale Quantum Computing.

References

[1] X. Fu, L. Riesebos, L. Lao, C. Almudever, F. Sebastiano, R. Versluis, E. Charbon, and K.Bertels, "A heterogeneous quantum computer architecture," *Proceedings of the ACM International Conference on Computing Frontiers*, pp. 323–330, 2016.

[2] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, "An experimental microarchitecture for a superconducting quantum processor," *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 813–825, 2017. arXiv: 1708.07677.

[3] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen, J. M. Chow, A. D. Córcoles, D. Egger, S. Filipp, J. Gomez, M. Hush, A. Javadi-Abhari, D. Moreda, P. Nation, B. Paulovicks, E. Winston, C. J. Wood, J. Wootton, and J. M. Gambetta, "Qiskit backend specifications for OpenQASM and OpenPulse experiments," 2018. arXiv: 1809.03452.

[4] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A practical quantum instruction set architecture," 2017. arXiv: 1608.03355.

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

[6] Y. Shi, N. Leung, P. Gokhale, Z. Rossi, D. I. Schuster, H. Hoffman, and F. T. Chong, "Optimized compilation of aggregated instructions for realistic quantum computers," 2019. arXiv: 1902.01474.

[7] F. Chong, D. Franklin, and M. Martonosi, "Programming languages and compiler design for realistic quantum hardware," *Nature*, vol. 549, pp. 180–187, 2017.

[8] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems," 2015.

[9] 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," 2017.

[10] J. M. Hornibrook, J. I. Colless, I. D. C. Lamb, S. J. Pauka, H. Lu, A. C. Gossard, J. D. Watson, G. C. Gardner, S. Fallahi, M. J. Manfra, and D. J. Reilly, "Cryogenic control architecture for large-scale quantum computing," 2015.