A Microarchitecture for a Superconducting Quantum Processor

X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, and R. F. L. Vermeulen
QuTech, Delft University of Technology

J. C. de Sterke
Topic Embedded Systems

W. J. Vlothuizen
Netherlands Organization for Applied Scientific Research

R. N. Schouten, C. G. Almudéver, L. DiCarlo, and K. Bertels
QuTech, Delft University of Technology

This article proposes a quantum microarchitecture, QuMA. Flexible programmability of a quantum processor is achieved by multilevel instructions decoding, abstracting analog control into digital control, and translating instruction execution with non-deterministic timing into event trigger with precise timing. QuMA is validated by several single-qubit experiments on a superconducting qubit.

To construct a fully programmable quantum computer based on the circuit model, a system stack[1] composed of several layers is required (see Figure 1). Quantum algorithms are formulated and then described using a high-level quantum programming language. Depending on the choice of quantum error correction code, such as surface code, the compiler takes that description as input, performs optimization, and generates a fault-tolerant implementation of the original quantum algorithm. Next, it implements the algorithm using instructions belonging to a quantum instruction set architecture (QISA). Just like in classical architectures, the QISA is the interface between software and hardware. A control microarchitecture is needed to decode the quantum instructions into microcode. The microcode can represent the required control signals with precise timing, as well as real-time quantum error detection and correction. Finally, based on the specific quantum technology—superconducting qubits, trapped ions, spin qubits, nitrogen-vacancy centers, and so on—control signals are translated into required pulses and sent to the quantum chip through the quantum-classical interface.

To date, research in quantum computer engineering has focused primarily on the top and bottom layers of the system stack, leaving a gap between quantum software and hardware. On the one hand, most of the existing quantum compilers mainly focus on efficiently describing and optimizing the application for a large number of qubits and pay little attention to low-level constraints of controlling physical qubits, such as the complex analog waveforms or the precise timing of operations on the nanosecond timescale. On the other hand, current popular methods of

Published by the IEEE Computer Society
0272-1732/18/$33.00 ©2018 IEEE

controlling qubits are mainly based on autonomous arbitrary waveform generators (AWG) and data collection units. These methods introduce high resource consumption, long configuration times, and control complexity, all of which scale poorly with the number of qubits. Hence, a conversion from the compiler output to the control medium accepted by the quantum processor is required to enable operating a quantum processor.

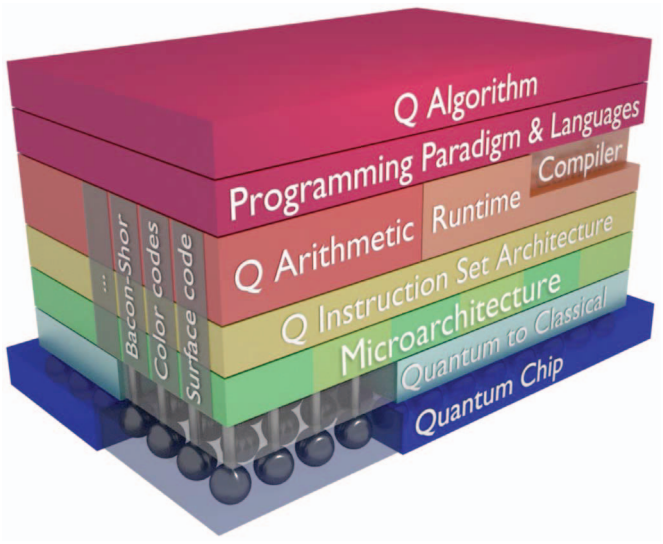

Figure 1. Overview of the quantum computer system stack.

In this article, we present a quantum microarchitecture, QuMA, for a superconducting quantum processor based on the circuit model that bridges the gap between quantum software and quantum hardware (see Figure 2).

QUMA

A quantum computer can be seen as a coprocessor acting as an accelerator. QuMA is a heterogeneous architecture, which includes a classical CPU as a host and a quantum coprocessor as an accelerator.

The input of QuMA is a binary file generated by a compiler infrastructure where classical and quantum code are combined. The classical code is produced by a conventional compiler such as the GNU compiler collection (GCC) and executed by the classical host CPU. Quantum code is generated by a quantum compiler and executed by the quantum coprocessor. The quantum code contains auxiliary classical instructions and quantum instructions. Auxiliary classical instructions perform basic arithmetic and logic operations and program flow control. Quantum instructions describe when and which quantum operations will be applied on which qubits.

As shown in Figure , the host CPU fetches quantum code from the memory and forwards it to the quantum coprocessor. In the quantum coprocessor, in general, executed instructions flow through modules from left to right. The execution controller performs register updates and program flow control, as well as streams quantum instructions to the physical execution layer. The physical microcode unit translates quantum instructions into microinstructions using the Q control store. These are further decomposed into micro-operations by the quantum microinstruction buffer (QMB). The timing of each micro-operation is also determined by the physical microcode unit. Based on the output of the QMB, the timing control unit triggers micro-operations at a deterministic timing. The analog-digital interface converts digitally represented micro-operations into corresponding analog pulses with precise timing that perform quantum operations on qubits, as well as analog signals containing measurement information of qubits into binary signals. Required modulation and demodulation with radio-frequency carrier waves are also carried out in the quantum-classical interface.

Three key mechanisms are at the core of QuMA:

- A multilevel instruction decoding scheme, which successively decodes a quantum instruction into microinstructions at the Q control store, micro-operations at the QMB, and, finally, codeword triggers at the micro-operation unit (μ-op unit).
- The queue-based event timing control scheme is implemented by the timing control unit, which issues event triggers with precise timing at nanosecond scale to the measurement discrimination unit (MDU) and the μ-op unit.
- The codeword-based event control scheme is implemented by the codeword-triggered pulse generation unit (CTPG), which produces analog input to the quantum processor based on the received codeword triggers, and the MDU converting the analog output from the quantum processor into binary results.

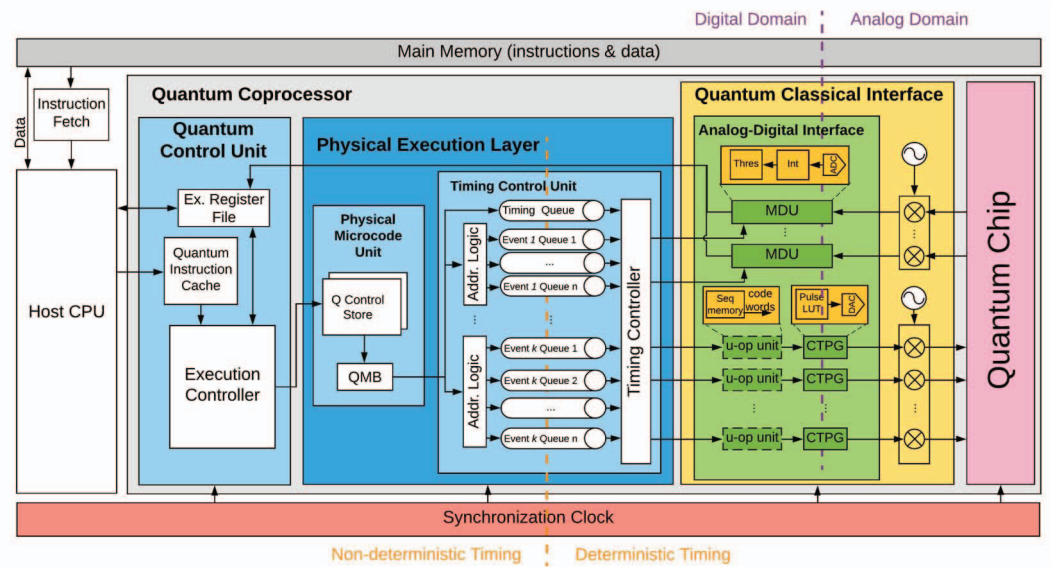

Figure 2. Overview of QuMA.

Multilevel Instruction Decoding

Quantum instructions are translated into a sequence of microinstructions in the physical microcode unit based on the microprograms uploaded into the Q control store. The timing for each quantum operation is also determined at this stage. In the QMB, quantum microinstructions for quantum gates are decomposed into separate micro-operations with timing labels and are pushed into the queues in the timing control unit. The timing control unit then emits the micro-operations at the expected timing. At the μ-op unit, each micro-operation is translated into a sequence of codeword (index) triggers with predefined latency, which further makes associated CTPGs generate primitive operation pulses. The microcode unit and the μ-op unit can be configured by the user, which enables QuMA to flexibly support instructions with explicit quantum semantics, which can be as independent as possible of a particular technology and its current state of the art.

Queue-Based Event Timing Control

The timing control unit implements the queue-based event timing control. It divides the microarchitecture into two timing domains: non-deterministic and deterministic (which are on the left and right side of the timing control unit, respectively). In the non-deterministic timing domain, instructions are executed in an as-fast-as-possible fashion, which assigns events, or micro-operations, on various timing points of a timeline. In the deterministic timing domain, micro-operations are emitted to the analog-digital interface with deterministic and precise timing.

The timing control unit consists of a timing queue, multiple event queues, and a timing controller. The timing queue buffers the time points with corresponding timing labels. The location of the time points can be designated in the timeline, such as by specifying the intervals between consecutive time points. Each event queue buffers a sequence of events with a time point at which the event is expected to take place. The time point is indicated by the aforementioned timing label. An event can be a quantum gate, a measurement, or any other operation.

The timing controller maintains the clock of the deterministic timing domain T_D, which can be started by an instruction or another source such as an external trigger. When T_D reaches the assigned time point, the timing controller broadcasts the timing label. The timing label signals the queues to fire the events matching that time point and emits them to the analog-digital interface.

Codeword-Based Event Control

The codeword-based event control scheme is implemented by the analog-digital interface. After uploading all the required primitive pulses into the memory, an index called codeword is assigned to each of the pulses, as well as to the measurement operation.

Digital-format micro-operations are first converted into codewords at the μ-op unit. These codewords trigger the CTPGs to generate analog pulses, or the customized MDU that translates analog qubit measurement waveforms into binary results. The CTPG and MDU should have a short and fixed latency.

In this way, the analog-digital interface abstracts the complex analog waveform generation and puts forward the responsibility of codeword control with precise timing to the upper digital layers. Therefore, it enables controlling analog pulse generation using instructions. Fast and flexible feedback control is also possible in principle because the CTPG scheme does not require the waveform to be uploaded at runtime, and codeword triggers with precise timing can be efficiently generated dynamically.

VALIDATION

In view of running physics experiments, we implemented the mentioned mechanisms in the quantum control box with two slight differences:

- Writing measurement results from the MDU to the exchange register file have not been implemented yet.
- Only the timing management part of the physical microcode unit has been implemented, and the conversion from quantum instructions to quantum microinstructions is yet to be supported.

Hence, a combination of auxiliary classical instructions and quantum microinstructions is accepted by the QuMA core. For now, the microinstruction set, QuMIS, consists of the following instructions (see Table 1):

- The Wait instruction used to specify the interval between consecutive time points,
- The Pulse instruction used to apply quantum gates on qubits,
- The MPG instruction used to generate the measurement pulse, and
- The MD instruction used to trigger the measurement discrimination process.

We validated QuMA by performing several single-qubit experiments on a superconducting quantum processor. These experiments include measurement of the relaxation time T_1 and dephasing time T_2 of the qubit, a standard gate-characterization experiment called *AllXY*,[2] and a gate error estimation experiment called randomized benchmarking.[3]

Table 1. QuMIS instructions.

Assembly Format	Description
Wait *Interval*	Wait for the number of cycles indicated by the immediate value *Interval*.
Pulse (*QAddr*$_0$, *μOp*$_0$) [, (*QAddr*$_1$, *μOp*$_1$), …]	Apply the micro-operation *μOp*$_i$ on each of the qubits specified by the address *QAddr*$_i$.
MPG *QAddr*, *D*	Generate the measurement pulse for the qubits specified by the address *QAddr*. *D* indicates the duration of the measurement pulse in number of cycles.
MD *QAddr*, *$rd*	Discriminate the measurement results of the qubits specified by *QAddr* and store the result into register *$rd*.

POTENTIAL IMPACT

QuMA fills the gap between quantum compilers and quantum hardware by providing a control system that translates quantum code into low-level analog signals that operate on the qubits. In addition, QuMA makes a move towards the first definition of an executable QISA. In our recent research, we improved the microcode unit by enabling the translation from a single instruction to multiple operations on different qubits. An executable QISA, named eQASM, is also defined on top of QuMIS. With certain low-level information exposed in eQASM, such as timing, the quantum compiler can generate executable instructions for real devices.

Some quantum algorithms for near-term devices ask for quantum-classical mixed computation, such as a variational eigenvalue solver.[4] Because data can be gathered into the register file in QuMA, it is natural to construct a heterogeneous computing platform with a classical host and a quantum coprocessor by adding extra data exchange instructions to interact with the host CPU and the main memory.

The verification of quantum software design creates a challenge. QuMA can assist the verification of quantum software and the estimation of their performance by simulating the generated instructions targeting QuMA. To this end, an architecture simulator for QuMA is required, which can simulate the execution of the instructions respecting hardware constraints and generate operations for each qubit with timing information. These timed operations can then be fed to a qubit state evolution simulator, such as QX[5] or QuantumSim.[6] In this way, the correctness of quantum software can be checked at both the architecture level and the qubit state level. Our previous work on the Quantum Platform Development framework (QPDO)[7] is a step towards building the required architecture simulator.

Programmable AWGs became available recently in industry.[8-9] In these devices, the analog channels are coupled to a processor with a large memory. Instead of instructions with explicit quantum semantics, low-level instructions are used to generate the output, such as the waveform instruction, which takes a physical memory address as parameter. A distributed architecture with a synchronization mechanism is assumed to provide more analog channels. The required hardware resources go up almost linearly to the number of qubits. In contrast, QuMA is a centralized architecture with quantum semantics and timing of operations explicitly defined at the instruction level. It does not depend on an external synchronization mechanism and can scale up to control tens of qubits. By adopting the codeword-triggered pulse generation scheme, the AWG complexity can be reduced, which costs modest hardware. Also, the requirement for multiple control processors can be eliminated, making a simple compilation model and again asking for less hardware resources.

In recent years, quantum processors with more qubits are being produced. More qubits, in general, ask for more operations per unit time on average, which requires more operations to be fed

into the queues. Only one instruction stream in QuMA results in a limited instruction issue rate, just as in classical processors. The limited instruction issue rate might be insufficient to issue all instructions in time that describe the required operations, which forms a bottleneck of QuMA. It is possible to make use of conventional processor design methods to optimize the non-deterministic timing domain without affecting the deterministic timing of the output. Inspired by conventional processor design techniques, such as the Intel Streaming SIMD Extensions (SSE), we proposed a Single-Operation-Multiple-Qubit (SOMQ) execution fashion for QuMA in our recent research. Together with a very-long-instruction-word architecture (VLIW) update, we implemented the digital part of the improved QuMA in a device capable of controlling seven qubits. With a slight change to the configuration, such as VLIW width, the device can be, in principle, extended to control at least 49 qubits, which can form a distance-5 surface code logical qubit.[10]

To further scale up the system, a tiled architecture consisting of multiple QuMA nodes with each node controlling tens of qubits would be a potential solution. In such a tiled architecture, the mechanisms in QuMA are still valid, but a communication protocol among nodes and a compilation model for a tiled system requires investigation.

Current methods allocate most electronics at room temperature, and coaxial cables are used to send analog signals to qubits that are in the cryogenic environment. The number of cables grows roughly linearly to the number of qubits. The footprint and the thermal conductance of the cables forms a challenge for a large number of qubits.[11] Addressing this issue, some research[12] investigates allocating part of the electronics, such as waveform generators, in the 4K environment. Whether a part of QuMA can be allocated in the 4K environment highly depends on the available power budget and the power consumption of each component of the QuMA implementation.

CONCLUSION

Various quantum technologies are being developed for quantum computing, including superconducting qubits and trapped ions. However, it is still unknown which quantum technology will be used to build future quantum computers. Though QuMA originally targets superconducting qubits, it can also be adapted to operate on different quantum technologies; some changes are required, including the microcode unit, the number and width of queues, and the quantum-classical interface. Our recent experiment demonstrates that QuMA is capable of controlling spin qubits.

We expect QuMA to spark a new line of research on a flexible and scalable approach to control near-term and future quantum chips. Building a quantum control microarchitecture and defining the required QISA can help the design of the control hardware, as well as the quantum software.

ACKNOWLEDGMENTS

A previous version of this article appears in the Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. We thank M. Tiggelman, S. Visser, J. Somers, L. Riesebos, E. Garrido Barrabés, and E. Charbon for contributions to an early version of the control box; H. Homulle for rendering Figure 1; and L. Lao, H.A. Du Nguyen, R. Versluis, and F.T. Chong for discussions. We acknowledge funding from the China Scholarship Council, Intel Corporation, an ERC Synergy Grant, and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the U.S. Army Research Office grant W911NF-16-1-0071. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

REFERENCES

1. X. Fu et al., "A heterogeneous quantum computer architecture," *Proceedings of the ACM International Conference on Computing Frontiers*, 2016.
2. M. D. Reed, "Entanglement and quantum error correction with superconducting qubits," dissertation, Yale University, 2013.
3. J. M. Epstein et al., "Investigating the limits of randomized benchmarking protocols," *Physical Review A*, vol. 89, 2014.
4. A. Peruzzo et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, 2014, p. 4213.
5. N. Khammassi et al., "QX: A high-performance quantum computer simulation platform," *2017 Design, Automation & Test in Europe Conference & Exhibition* (DATE), 2017, pp. 464–469.
6. T. E. O'Brien, B. Tarasinski, and L. DiCarlo, "Density-matrix simulation of small surface codes under current and projected experimental noise," *NPJ Quantum Information*, vol. 3, 2017, p. 39.
7. L. Riesebos et al., "Pauli frames for quantum computer architectures," *Proceedings of the 54th Annual Design Automation Conference 2017* (DAC), 2017, p. 76.
8. "BBN technologies arbitrary pulse sequencer 2," 2017; libaps2.readthedocs.org/en/latest/.
9. *M3202a PXIe arbitrary waveform generator, 1 GSa/s, 14 bit, 400 MHz*, 2017; www.keysight.com/en/pd-2747446-pn-M3202A/pxie-arbitrary-waveform-generator-1-gs-s-14-bit-400-mhz.
10. A. G. Fowler et al., "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, 2012.
11. C. G. Almudever et al., "The engineering challenges in quantum computing," *2017 Design, Automation & Test in Europe Conference & Exhibition* (DATE), 2017, pp. 836–845.
12. J. M. Hornibrook et al., "Cryogenic control architecture for large-scale quantum computing," *Physical Review Applied*, vol. 3, 2015.

ABOUT THE AUTHORS

Xiang Fu is a PhD student at QuTech in the Quantum and Computer Engineering department at Delft University of Technology. His research interests include quantum computer microarchitecture and quantum instruction set architecture. He has a master's degree in computer engineering from the National University of Defense Technology in China. He is a student member of the IEEE. Contact him at X.Fu-1@tudelft.nl.

Adriaan Rol is a PhD student at QuTech in the faculty of Applied Science at Delft University of Technology. His research interests include novel methods to calibrate and characterize qubit operations, as well as finding the right abstractions to scale up these methods to many qubit quantum systems in the context of demonstrating quantum fault tolerance. Before joining the superconducting transmon group, he worked on nitrogen vacancy centers in diamond for his master's degree, also at QuTech, Delft University of Technology. Contact him at M.A.Rol@tudelft.nl.

Niels Bultink is a PhD student at QuTech in the faculty of Applied Physics at Delft University of Technology. His research interests include quantum information processing on superconducting quantum processors and implementations of fault-tolerant quantum computing. His PhD focuses on improving multi-qubit state measurement to achieve faster and higher fidelity readout of increasing numbers of qubits. Bultink has a master's degree in applied physics. Contact him at C.C.Bultink@tudelft.nl.

Hans van Someren is a researcher at QuTech in the Quantum and Computer Engineering department at Delft University of Technology, where he investigates computer system architectures and supporting tools for quantum computing, especially specialized scheduling, mapping and routing, and the semantics of the various architectural layers and interface representations. Van Someren has a master's degree in mathematics from Delft University of Technology. Contact him at J.vanSomeren-1@tudelft.nl.

Nader Khammassi is a researcher at QuTech in the Quantum and Computer Engineering department of Delft University of Technology. His works include the design of the QX quantum computer simulator and the OpenQL quantum programming framework. He is investigating different layers of a scalable quantum computer architecture for different qubit technologies being developed at QuTech in collaboration with Intel. He has a PhD (cum laude) from the National Engineering School of Advanced Technology in Brittany, France, where he researched high-performance computing for multicore architectures. Contact him at N.Khammassi@tudelft.nl.

Imran Ashraf is a postdoctoral researcher at QuTech in the Quantum and Computer Engineering department at Delft University of Technology. His recent research focuses on compilation techniques for quantum computing. He has a PhD in computer engineering from Delft University of Technology. The focus of his research was advanced profiling, code parallelization, and communication-driven mapping of applications on multicore platforms. He is a member of the IEEE. Contact him at I.Ashraf@tudelft.nl.

Raymond Vermeulen is an electronic instrumentation engineer at QuTech at Delft University of Technology, where he designs and builds electronics in support of ongoing and future research into quantum computing. Vermeulen has a bachelor's degree in electrical engineering from Zuyd University of Applied Sciences. Contact him at R.F.L.Vermeulen@tudelft.nl.

Jacob de Sterke is a senior hardware designer specializing in FPGA design at Topic Embedded Systems. In support of the Applied Physics team at QuTech, he works on the electronics and FPGA development for ongoing and future research on many-qubit superconducting quantum systems. He has a bachelor's degree in electrical engineering and information technology. Contact him at Jacob.de.sterke@topic.nl.

Wouter Vlothuizen is a senior systems architect at QuTech in the Radar Technology department of the Netherlands Organization for Applied Scientific Research. His research focusses on hardware/software co-design for real-time control and signal processing applications. Vlothuizen has a master's degree in electrical engineering from Delft University of Technology. Contact him at wouter.vlothuizen@tno.nl.

Raymond Schouten is a senior electronics engineer at QuTech at Delft University of Technology, where he works on improving the research measurement results. He gives advice on measurement techniques and equipment, designs front-end electronics, evaluates commercial measurement equipment, gives training, and troubleshoots. Designing for low-noise analog and achieving low interference levels are main areas of interest. He has a bachelor's degree in electrical engineering. Contact him at R.N.Schouten@tudelft.nl.

Carmen G. Almudéver is an assistant professor at the Quantum and Computer Engineering Department at Delft University of Technology. Her research interests include scalable quantum computer architecture and mapping of fault-tolerant quantum circuits. She has a PhD in electrical engineering from Polytechnic University of Catalonia. She is a member of the IEEE. Contact her at C.GarciaAlmudever-1@tudelft.nl.

Leonardo DiCarlo is an associate professor in the Department of Quantum Nanoscience and a roadmap leader at QuTech at Delft University of Technology. His main research interest is the development of a full-stack quantum computer based on superconducting quantum circuits. DiCarlo has a master's degree in electrical engineering from Stanford University and a PhD in physics from Harvard University. Contact him at L.Dicarlo@tudelft.nl.

Koen Bertels is a professor and the head of the Quantum and Computer Engineering department at Delft University of Technology. His research focuses on quantum computing, specifically the overall system design and architecture aspects. He is a principal investigator at QuTech, where he collaborates with experimental physicists on building prototype quantum computers. He has a PhD from the University of Antwerp. He is a member of the IEEE and ACM. Contact him at K.L.M.Bertels@tudelft.nl.