A Microarchitecture for a Superconducting Quantum Processor

Fu, X.; Rol, M.A.; Bultink, C.C.; van Someren, J.; Khammassi, N.; Ashraf, I.; Vermeulen, R.F.L.; De Sterke, J.C.; Vlothuizen, W.J.; Schouten, R.N.

DOI
10.1109/MM.2018.032271060

Publication date
2018

Document Version
Accepted author manuscript

Published in
IEEE Micro

Citation (APA)
Fu, X., Rol, M. A., Bultink, C. C., van Someren, J., Khammassi, N., Ashraf, I., Vermeulen, R. F. L., De Sterke, J. C., Vlothuizen, W. J., Schouten, R. N., Almudéver, C. G., DiCarlo, L., & Bertels, K. (2018). A Microarchitecture for a Superconducting Quantum Processor. *IEEE Micro*, *38*(3), 40-47. https://doi.org/10.1109/MM.2018.032271060

A Microarchitecture for a Superconducting Quantum Processor

X. Fu[1,2,*] M. A. Rol[1,3] C. C. Bultink[1,3] J. van Someren[1,2] N. Khammassi[1,2] I. Ashraf[1,2] R. F. L. Vermeulen[1,3]
J. C. de Sterke[4,1] W. J. Vlothuizen[5,1] R. N. Schouten[1,3] C. G. Almudever[1, 2] L. DiCarlo[1,3,†] K. Bertels[1,2,‡]
[1] QuTech, Delft University of Technology
[2] Computer Engineering Lab, Delft University of Technology
[3]Kavli Institute of Nanoscience, Delft University of Technology
[4] Topic Embedded Systems B.V.
[5] Netherlands Organisation for Applied Scientic Research (TNO)
{* x.fu-1, † l.dicarlo, ‡ k.l.m.bertels}@tudelft.nl

ABSTRACT

[1]This paper proposed a quantum microarchitecture, QuMA, and a quantum microinstruction set, QuMIS, to bridge the gap between quantum software and hardware. Flexible programmability of a quantum processor is achieved by multi-level instructions decoding, abstracting analog control into digital control, and translating instruction execution with non-deterministic timing into event trigger with precise timing. QuMA and QuMIS are validated by several single-qubit experiments on a superconducting qubit.

1 INTRODUCTION

Quantum computers promise to solve certain problems that are intractable for classical computers, such as factoring large numbers and simulating quantum systems. A fully programmable quantum computer requires a seamless collaboration across all layers in a system stack as shown in Figure 1. The compiler translates high-level language described quantum algorithms into instructions belonging to the quantum instruction set architecture (QISA). The microarchitecture takes the QISA instructions as a direct input and generates the corresponding control signals, which perform quantum operations on the quantum chip after the conversion of the quantum-classical interface.

To date, research in quantum computer engineering has focused primarily at the top and bottom layers of the system stack, leaving a gap in the between of 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 controlling qubits are mainly based on autonomous arbitrary waveform generators (AWG) and data collection units, which 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.

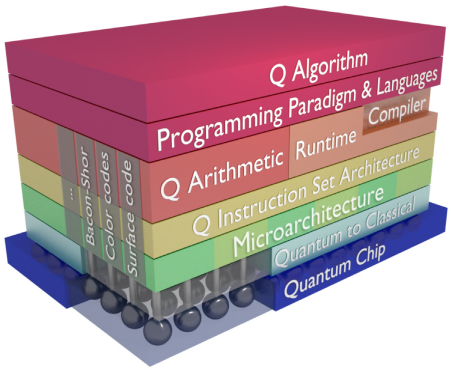

Figure 1: Overview of the quantum computer system stack from [1].

In this article, we present a Quantum MicroArchitecture (QuMA) for a superconducting quantum processor that bridges the gap between quantum software and quantum hardware.

2 QUANTUM MICROARCHITECTURE

A quantum computer can be seen as a coprocessor acting as an accelerator. As shown in Figure 2, the proposed quantum microarchitecture (QuMA), is a heterogeneous architecture, which consists of a host CPU executing the classical code, and a quantum coprocessor executing the quantum code. 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 which and when quantum operations will be applied on qubits.

Three key mechanisms are at the core of QuMA: (i) a flexible multilevel instruction decoding mechanism, (ii) queue-based precise event timing control, and (iii) a codeword-based event control scheme.

[1]**Original paper title**: An Experimental Microarchitecture for a Superconducting Quantum Processor. **Conference**: The 50th Annual IEEE/ACM International Symposium on Microarchitecture. **Pages**: 813-825. **Publication Date**: 2017-10-14.

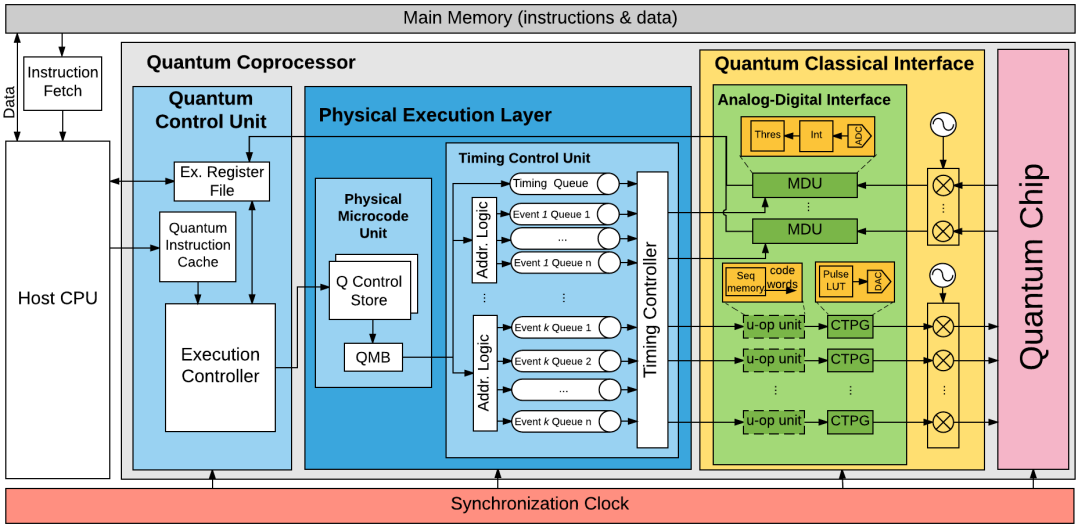

Figure 2: Overview of the Quantum MicroArchitecture (QuMA).

2.1 Multilevel Instruction Decoding

Quantum instructions, especially those for quantum gates, are firstly decoded into quantum microinstructions, which are then decomposed into separate micro-operations at the microcode unit. After being issued by the timing control unit, each micro-operation is translated into a sequence of codeword (index) triggers at the micro-operation unit (u-op unit). Each codeword triggers the codeword-triggered pulse generation unit (CTPG) to generate a primitive operation pulse.

Since the microcode unit and the micro-operation unit are definable by the user, it 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.

2.2 Queue-Based Event Timing Control

As shown in Figure 2, the timing control unit implements the queue-based event timing control. It contains a set of queues which divide QuMA into two parts: the non-deterministic timing domain on the left and the deterministic timing domain on the right.

In the non-deterministic timing domain, instructions are executed in an as-fast-as-possible fashion to feed the queues, which constructs a timeline with events at specific timing points. The timeline is constructed by specifying intervals between consecutive and labeled timing points. Every event, or micro-operation, is associated with a specific timing point by using a timing label. These events can be a quantum gate, a measurement, or any other operation. Both timing points and events are buffered in the timing and event queues, respectively. The deterministic timing domain maintains a clock corresponding to the timeline. The first timing point of the timeline can be designated by an instruction, or another source, e.g., an external trigger. After every interval, a timing point is reached, and the assigned timing label is broadcast, which triggers all events with the same timing label.

2.3 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 firstly converted into codewords. These codewords trigger the codeword-triggered pulse generation unit (CTPG) to generate analog pulses that operate on the qubits, or the customized measurement discrimination unit that translates analog qubit measurement waveforms into binary results.

In this way, complex analog waveform generation is abstracted into simple digital control with precise timing.

3 VALIDATION

We implemented the mentioned mechanisms in the quantum control box with two slight differences. (i) 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. The current quantum microinstruction set (QuMIS) is shown in Table 1. (ii) Writing measurement results from the MDU to the exchange register file has not been implemented yet.

We validated QuMA by performing several single-qubit experiments on a superconducting quantum processor, including measurement of the relaxation time (T_1) and dephasing time (T_2) of the qubit, a standard gate-characterization

Table 1: QuMIS instructions.

Assembly Format	Description
Wait *Interval*	Wait for the number of cycles indicated by the immediate value *Interval*.
Pulse ($QAddr_0$, uOp_0)[, ($QAddr_1$, uOp_1), . . .]	Apply the micro-operation uOp_i on each of the qubit(s) 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*.

experiment, called *AllXY* [2, 3], and a gate error estimation experiment, called randomized benchmarking [4].

4 POTENTIAL IMPACT

The quantum microarchitecture presented in this paper, 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 [5]. 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 forms 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 feed input to a qubit state evolution simulator, such as CHP or QX [6]. Our previous work on simulating Pauli frame [7] is a step towards building the required architecture simulator.

Programmable AWGs became available recently in industry [8–10]. In these devices, the analog channels are coupled to a processor with a large memory. Instead of instructions with explicit quantum semantics, low-level output instructions are used, such as *waveform* with a physical memory address. 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.

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. 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 [11].

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

Current methods allocate most electronics at room temperature and coaxial cables are used to sent analog signals to qubits that are in the cryogenic environment. The number of cables grows roughly linearly to the number of qubits. The footprint as well as the thermal conductance of the cables forms a challenge for a large number of qubits [12]. Addressing this issue, some research [13, 14] investigates allocating the part of the electronics, such as waveform generators, in the 4 K environment. Whether a part of QuMA can be allocated in the 4 K environment highly depends on the available power budget and the power consumption of each component of the QuMA implementation.

Various quantum technologies are being developed for quantum computing, including superconducting qubits, trapped ions, and so on. 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. In our recent experiment, QuMA has been demonstrated to control spin qubits.

We expect QuMA to set 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 [2].

ACKNOWLEDGMENTS

We thank M. Tiggelman, S. Visser, J. Somers, L. Riesebos, E. Garrido Barrabés, and E. Charbon for contributions to an early version of the CBox, A. Bruno for fabricating the quantum chip, H. Homulle for drawing 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 (X. Fu), 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, L. Riesebos, L. Lao, C. Almudever, F. Sebastiano, R. Versluis, E. Charbon, and K. Bertels, "A heterogeneous quantum computer architecture," in *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 2016, pp. 323–330.

[2] J. M. Chow, L. DiCarlo, J. M. Gambetta, F. Motzoi, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, "Optimized driving of superconducting artificial atoms for improved single-qubit gates," *Physical Review A*, vol. 82, p. 040305, 2010.

[3] M. D. Reed, "Entanglement and quantum error correction with superconducting qubits," Ph.D. dissertation, Yale University, 2013.

[4] J. M. Epstein, A. W. Cross, E. Magesan, and J. M. Gambetta, "Investigating the limits of randomized benchmarking protocols," *Physical Review A*, vol. 89, no. 6, p. 062321, 2014.

[5] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. OâĂŹBrien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, 2014.

[6] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudever, and K. Bertels, "Qx: A high-performance quantum computer simulation platform," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 464–469.

[7] L. Riesebos, X. Fu, S. Varsamopoulos, C. Almudever, and K. Bertels, "Pauli frames for quantum computer architectures," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 76.

[8] Raytheon BBN, "Bbn technologies arbitrary pulse sequencer 2," http://libaps2.readthedocs.org/en/latest/, 2017.

[9] Keysight, "M3202a pxie arbitrary waveform generator, 1 gsa/s, 14 bit, 400 mhz," http://www.keysight.com/en/pd-2747446-pn-M3202A/pxie-arbitrary-waveform-generator-1-gs-s-14-bit-400-mhz?cc=US&lc=eng, 2017.

[10] Zurich Instrument, "Uhfawg arbitrary waveform generator," https://www.zhinst.com/products/uhfawg#overview, 2017.

[11] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, 2012.

[12] C. G. Almudever, L. Lao, X. Fu, N. Khammassi, I. Ashraf, D. Iorga, S. Varsamopoulos, C. Eichler, A. Wallraff, L. Geck *et al.*, "The engineering challenges in quantum computing," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 836–845.

[13] J. M. Hornibrook, J. I. Colless, I. D. Conway 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," *Physical Review Applied*, vol. 3, p. 024010, 2015.

[14] H. Homulle, S. Visser, B. Patra, G. Ferrari, E. Prati, C. G. Almudéver, K. Bertels, F. Sebastiano, and E. Charbon, "Cryocmos hardware technology a classical infrastructure for a scalable quantum computer," in *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 2016, pp. 282–287.

[2] Suggested reference format: 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, K. Bertels. A Microarchitecture for a Superconducting Quantum Processor. 2017.