УДК 512.6, 517.9, 519.6

QUANTUM SUPREMACY IN END-TO-END INTELLIGENT IT. PT. I: QUANTUM SOFTWARE ENGINEERING – QUANTUM GATE LEVEL APPLIED MODELS SIMULATORS

Ivancova Olga[1], Korenkov Vladimir[2], Tyatyushkina Olga[3], Ulyanov Sergey[4], Fukuda Toshio[5]

[1]*Senior researcher;*
Joint institute for nuclear researches, Laboratory of Information Technologies;
Dubna State University,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: o_ivancova@mail.ru.

[2]*Doctor of Technical Science, professor;*
Joint institute for nuclear researches, Laboratory of Information Technologies;
Deputy Director of the Laboratory;
141980, Moscow reg., Dubna, Joliot-Curie, 6;
e-mail: korenkov@cv.jinr.ru.

[3] *PhD, Associate professor;*
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: tyatyushkina@mail.ru.

[4]*Doctor of Science in Physics and Mathematics, professor;*
Joint institute for nuclear researches, Laboratory of Information Technologies;
Dubna State University,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ulyanovsv@mail.ru.

[5]*PhD, professor; president IEEE;*
Dept. of Micro System, Dept. of Mechanics- Informatics, Nagoya University;
Furo-cho, Chikusa-ku, Nagoya, Japan;
e- mail: fukuda@mein.nagoya u.ac.jp.

Principles and methodologies of quantum algorithmic gates design for master course and PhD students in computer science, control engineering and intelligent robotics described. The possibilities of quantum algorithmic gates simulation on classical computers discussed. Applications of quantum gate of nanotechnology in intelligent quantum control introduced. A new approach to a circuit implementation design of quantum algorithm gates for fast quantum massive parallel computing presented. The main attention focused on the development of design method of fast quantum algorithm operators as superposition, entanglement and interference, which are in general time-consuming operations due to the number of products that have performed. SW & HW support sophisticated smart toolkit of supercomputing accelerator of quantum algorithm simulation on small quantum programmable computer algorithm gate (that can program in SW to implement arbitrary quantum algorithms by executing any sequence of universal quantum logic gates) described. As example, the method for performing Grover's interference operator without product operations introduced. The background of developed information technology is the "Quantum / Soft Computing Optimizer" (QSCOptKB^{TM}) SW based on soft and quantum computational intelligence toolkit.

Keywords: quantum algorithm, quantum computer, quantum computation intelligence, quantum programming.

КВАНТОВОЕ ПРЕВОСХОДСТВО В СКВОЗНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ИТ. Ч. 1: КВАНТОВАЯ ПРОГРАММНАЯ ИНЖЕНЕРИЯ – МОДЕЛИРОВАНИЕ КВАНТОВЫХ АЛГОРИТМИЧЕСКИХ ЯЧЕЕК

Иванцова Ольга Владимировна[1], Кореньков Владимир Васильевич[2], Тятюшкина Ольга Юрьевна[3], Ульянов Сергей Викторович[4], Фукуда Тошио[5]

[1]*Старший преподаватель;*
ГБОУ ВО МО «Университет Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: o_ivancova@mail.ru.

[2]*Доктор технических наук, профессор;*
Объединенный институт ядерных исследований,
Директор лаборатории информационных технологий;
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, 6;
e-mail: korenkov@cv.jinr.ru.

[3]*Кандидат технических наук, доцент;*
ГБОУ ВО МО «Университет Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: tyatyushkina@mail.ru.

[4]*Доктор физико-математических наук, профессор;*
ГБОУ ВО МО «Университет Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

[5]*Доктор наук, профессор;*
Факультет микросистем, механики и информатики;
Нагоя университет;
Япония, Нагоя, Фуро-чо, Чикуса-ку;
e- mail: fukuda@mein.nagoya u.ac.jp.

Описаны принципы и методологии проектирования квантовых алгоритмических ячеек для магистрантов и аспирантов в области компьютерных наук, теории управления и интеллектуальной робототехники. Обсуждаются возможности моделирования квантовых алгоритмических ячеек на классических компьютерах. Описаны приложения схемотехнических решений квантовых ячеек. Представлен новый подход к реализации схемотехнических решений квантовых алгоритмов для быстрых квантовых параллельных массивных вычислений. Основное внимание уделено разработке метода проектирования операторов быстрых квантовых алгоритмов, таких как суперпозиция, запутывание и интерференция, которые в общем случае являются трудоемкими операциями из-за количества выполненных продуктов. Программно-алгоритмическая платформа поддерживает сложный интеллектуальный инструментарий ускорителя моделирования квантового алгоритма на малом квантовом компьютере, на котором реализуются квантовые алгоритмы путем выполнения последовательности универсальных логических элементов квантовой логики. В качестве примера, представлен способ выполнения оператора Гровера. Основой разработанной информационной технологии является ПО "Quantum / Soft Computing Optimizer" (QSCOptKB^{TM})*, основанное на мягких и квантовых вычислениях и является платформой сильного квантового вычислительного интеллекта.*

Ключевые слова: квантовые алгоритмы, малый квантовый компьютер, сильный вычислительный интеллект, квантовое программирование.

1. Introduction: quantum algorithm gate level computing supremacy – quantum deep learning applications in intelligent cognitive control and robotics

R. Feynman and Yu. Manin (see, below photo on Fig. 1), independently, suggested and correctly shown that quantum computing can be effectively applied for simulation and searching of solutions of classically intractable and algorithmically unsolved quantum systems problems using quantum programmable computer (as physical devices).

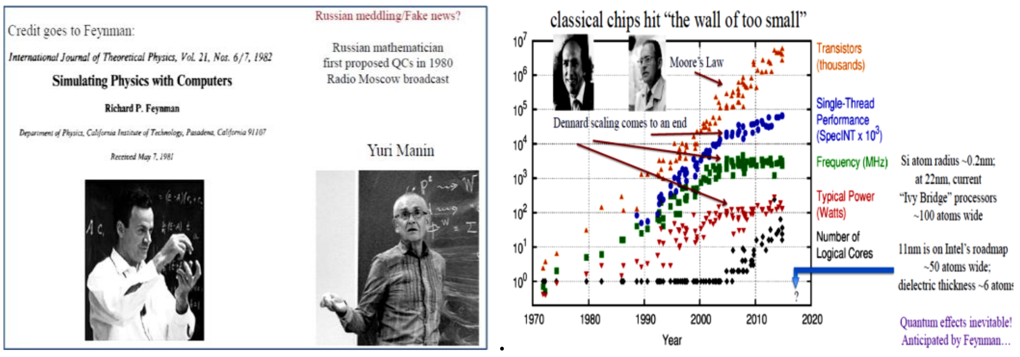
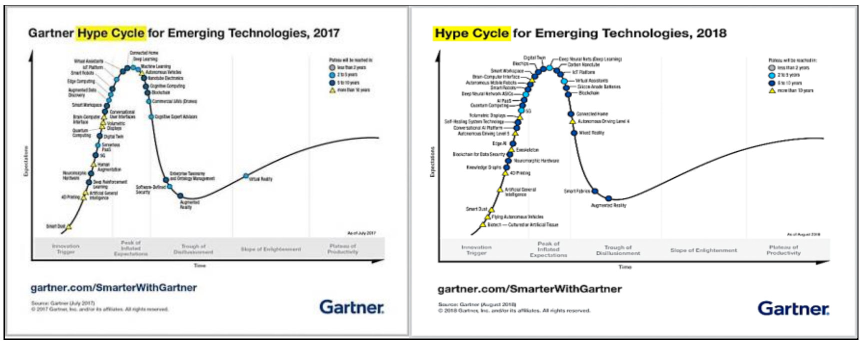
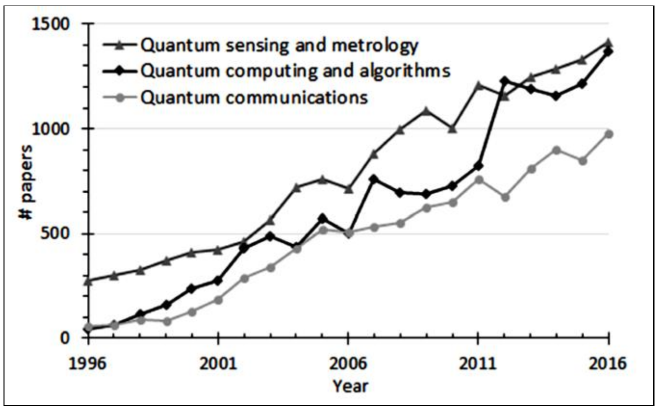


Figure 1. Original data up to the year 2010 [1]

Gartner hyper-cycle for emerging technologies (Fig. 2) predicted [2, 3] the smarter developments in quantum computing and end-to-end quantum IT (quantum supremacy) for quantum engineering in intelligent control and robotics.



(a)



(b)

Figure 2. (a) Gartner hype cycle for emerging technologies; (b) The number of research papers published per year in quantum computing and algorithms, quantum communications, and quantum sensing and metrology, respectively

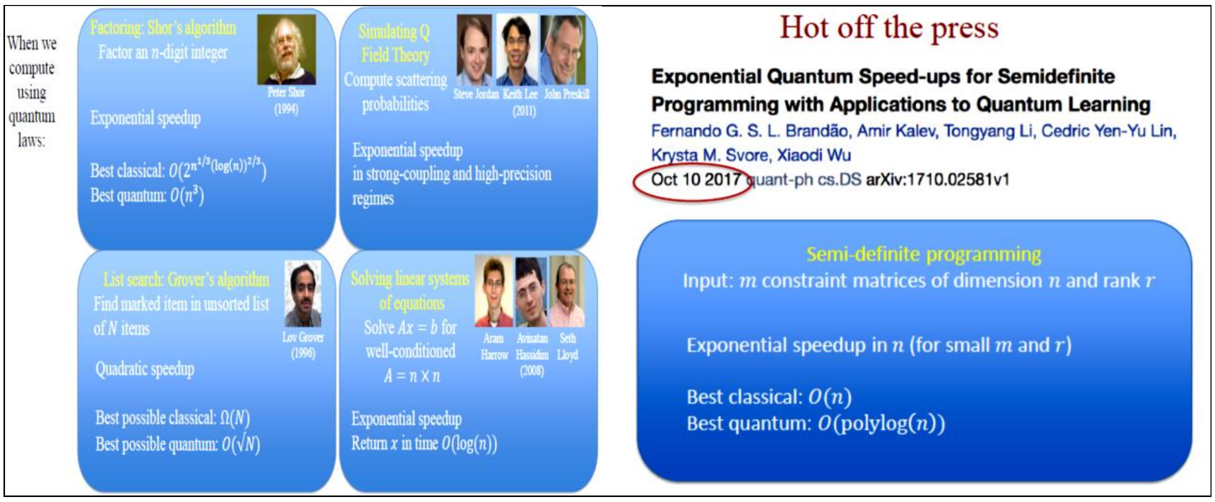
Recent research shows successful engineering application of end-to-end quantum computing information technologies (as quantum sophisticated algorithms and quantum programming) in searching of solutions of algorithmic unsolved (classically intractable) problems in classical dynamic intelligent control systems, artificial intelligence (AI) and intelligent cognitive robotics.

Perhaps the most important open problem in the theory of quantum information processing and problem-oriented engineering applications is to understand the nature of quantum mechanical speed-up for the solution of computational problems:

Q: What problems can be solved more rapidly using quantum computers than is possible with classical computers, and what ones cannot?

Computation, based on the laws of classical physics, leads to different constraints on information processing than computation processes based on quantum mechanics.

Figure 3 demonstrate efficient quantum computing for different algorithm computation problem.



When we compute using quantum laws:

Factoring: Shor's algorithm
Factor an n-digit integer
Peter Shor (1994)
Exponential speedup
Best classical: $O(2^{n^{1/3}(\log(n))^{2/3}})$
Best quantum: $O(n^3)$

Simulating Q Field Theory
Compute scattering probabilities
Steve Jordan Keith Lee John Preskill (2011)
Exponential speedup in strong-coupling and high-precision regimes

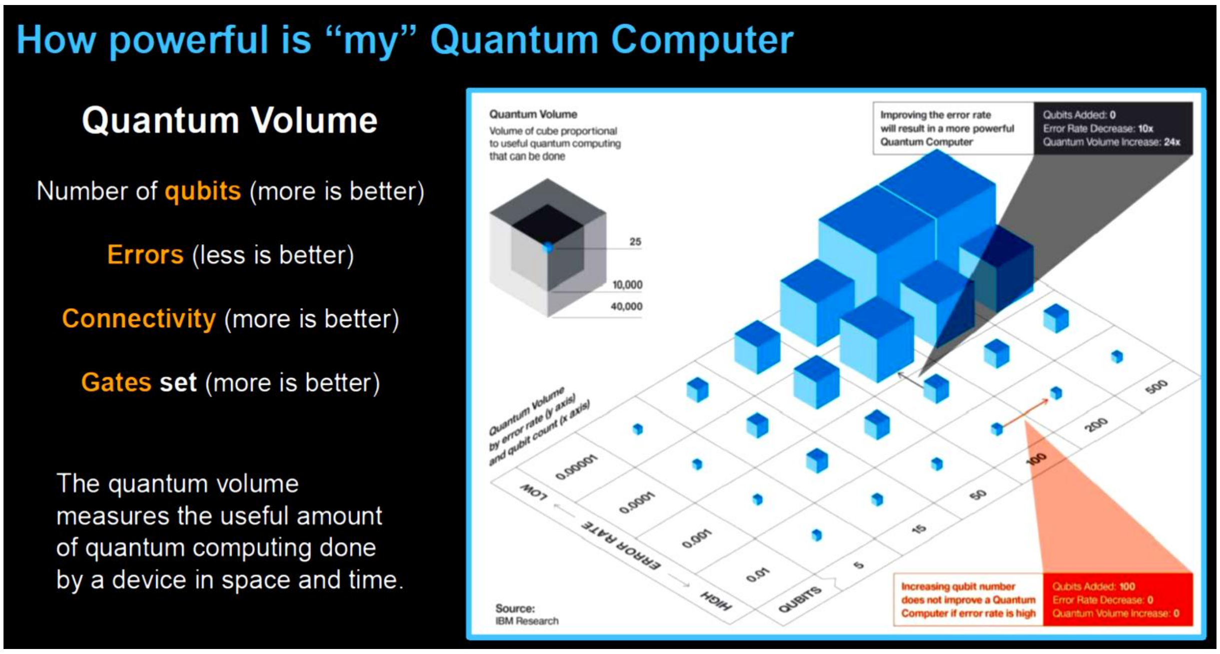
List search: Grover's algorithm
Find marked item in unsorted list of N items
Lov Grover (1996)
Quadratic speedup
Best possible classical: $\Omega(N)$
Best possible quantum: $O(\sqrt{N})$

Solving linear systems of equations
Solve $Ax = b$ for well-conditioned $A = n \times n$
Aram Harrow Avinatan Hassidim Seth Lloyd (2008)
Exponential speedup
Return x in time $O(\log(n))$

Hot off the press

Exponential Quantum Speed-ups for Semidefinite Programming with Applications to Quantum Learning
Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, Xiaodi Wu
Oct 10 2017 quant-ph cs.DS arXiv:1710.02581v1

Semi-definite programming
Input: m constraint matrices of dimension n and rank r
Exponential speedup in n (for small m and r)
Best classical: $O(n)$
Best quantum: $O(\text{polylog}(n))$



How powerful is "my" Quantum Computer

Quantum Volume

Number of **qubits** (more is better)

Errors (less is better)

Connectivity (more is better)

Gates set (more is better)

The quantum volume measures the useful amount of quantum computing done by a device in space and time.

Quantum Volume
Volume of cube proportional to useful quantum computing that can be done
25
10,000
40,000
Improving the error rate will result in a more powerful Quantum Computer
Qubits Added: 0
Error Rate Decrease: 10x
Quantum Volume Increase: 24x
Quantum Volume by error rate (y axis) and qubit count (x axis)
0.00001
0.0001
0.001
0.01
LOW
ERROR RATE
HIGH
QUBITS
5
15
50
100
200
500
Increasing qubit number does not improve a Quantum Computer if error rate is high
Qubits Added: 100
Error Rate Decrease: 0
Quantum Volume Increase: 0
Source: IBM Research

Figure 3. Speed-up Benchmarks of quantum computing.

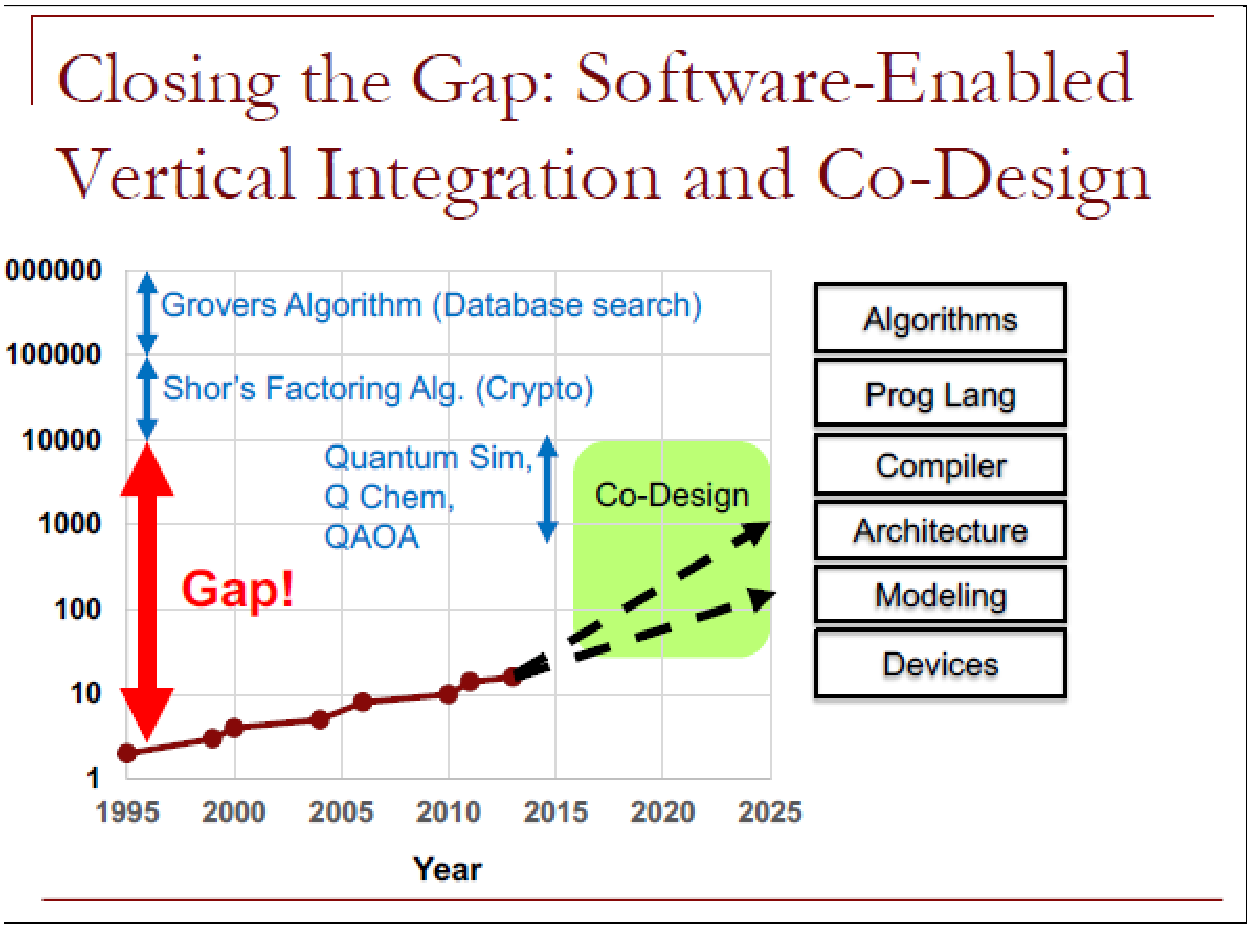
To realize how fast a quantum algorithm is in comparison to the classical methods of solving computationally difficult tasks, let us consider the best conventional algorithm known at present, which is the general sieve number (GNFS) algorithm, famous for factoring large n-bit integers, $N \le 2^n - 1$, larger than 10^{100}, in the sub-exponential time of $\mathcal{O}\left(\exp\sqrt[3]{\frac{64}{9}n\ \log n^{\ 2}}\right)$. On the other hand, the Shor quantum algorithm is only $\mathcal{O}\left(n^3\right)$ time, at the cost of $\mathcal{O}\left(n\right)$ quantum gates. Thus, the Shor's quantum algorithm is $\dfrac{\mathcal{O}\left(\exp\left(\sqrt[3]{\frac{64}{9}n\left(\log n\right)^2}\right)\right)}{\mathcal{O}\left(\sqrt[3]{n}\right)} = \mathcal{O}\left(\exp\left(\sqrt[3]{\frac{64}{9}n\left(\log n\right)^2}\right)\right)$ and faster than the GNFS method, which is at least a fascinating speed-up.

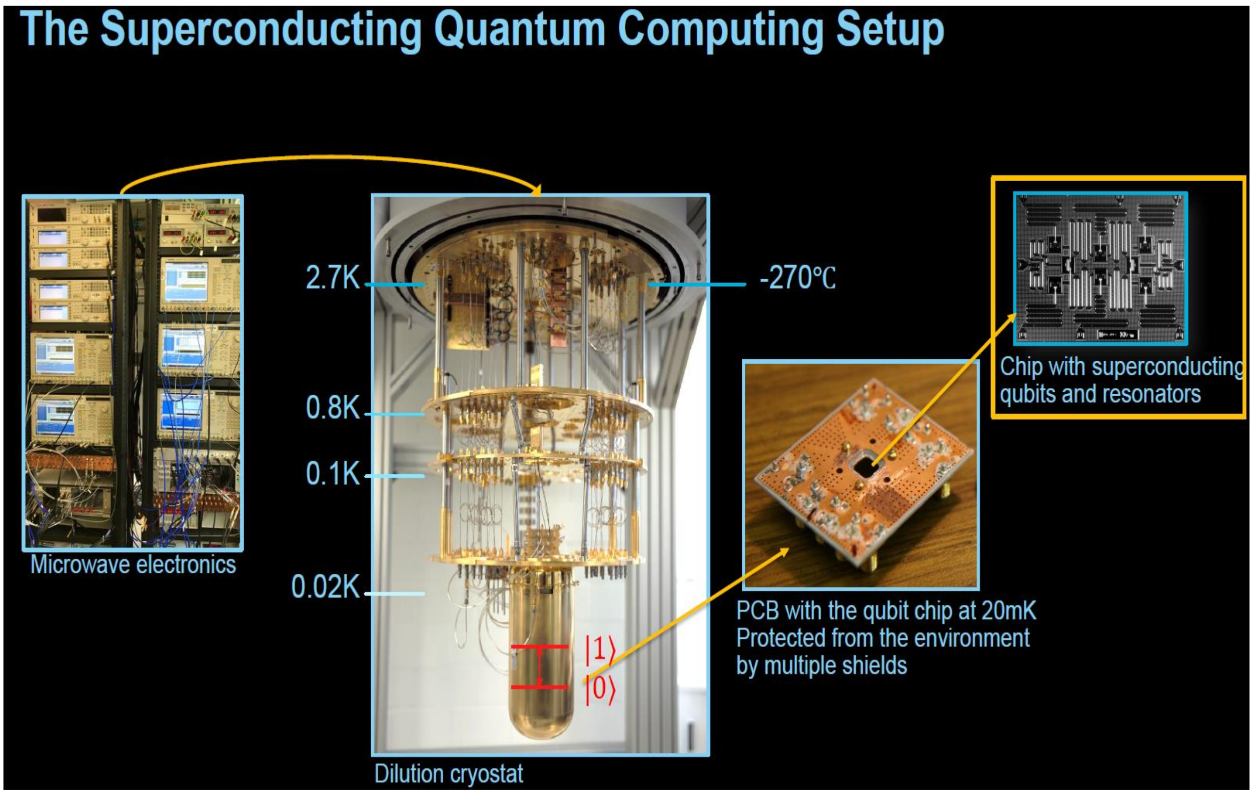
Another very clever application, having as kernel the Deutch-Jozsa quantum decision-making algorithm, is the also famous Grover quantum search in unstructured large data base algorithm. The Grover key-search method is a quantum algorithm capable of violating a password authentication system, such as the RFS encryption system in only $\mathcal{O}\left(\sqrt{N}\right)$ (or steps $\mathcal{O}\left(\sqrt{N/m}\right)$ with m marked items), where $N = 2^n$ is the number of possibilities of matching the oracle's kept-in-secret *n*-bit key. To realize its quadratic speed-up, the Grover's algorithm would be $\mathcal{O}\left(2^{128}\right)$ faster than the conventional brute force method to break a 256-bit secret key.

Quantum computers hold promise for solving many intractable (in classical mean) problems. However, unfortunately, there currently exist no algorithms for "programming" a quantum computer. Calculation in a quantum computer (like calculation in a conventional computer) can be described as a marriage of quantum HW (the physical embodiment of the computing machine itself, such as quantum gates and the like), and quantum SW (the computing algorithm implemented by the HW to perform the calculation). To date, SW of quantum algorithms (QA), such as Shor's algorithm, used to solve problems on a quantum computer have been developed on an *ad hoc* basis without any real structure or programming methodology.

Important computer-scientific challenges for quantum information science are to discover efficient QAs for interesting engineering problems and to understand the fundamental capabilities and limitations of quantum computation in comparison to those of classical computation. The bulk of this article is concerned with the problem of discovering new QAs. To take full advantage of the power of quantum computers, we should try to find new problems that are amenable to quantum speed-up.

As Fig. 4 conceptually illustrates, many well-known QC algorithms have qubit resource requirements that far exceed the current scale at which QCs can built.


Closing the Gap: Software-Enabled Vertical Integration and Co-Design
000000
100000
10000
1000
100
10
1
1995
2000
2005
2010
2015
2020
2025
Year
Grovers Algorithm (Database search)
Shor's Factoring Alg. (Crypto)
Quantum Sim, Q Chem, QAOA
Gap!
Co-Design
Algorithms
Prog Lang
Compiler
Architecture
Modeling
Devices


The Superconducting Quantum Computing Setup
Microwave electronics
2.7K
0.8K
0.1K
0.02K
-270℃
|1⟩
|0⟩
Dilution cryostat
PCB with the qubit chip at 20mK
Protected from the environment
by multiple shields
Chip with superconducting
qubits and resonators

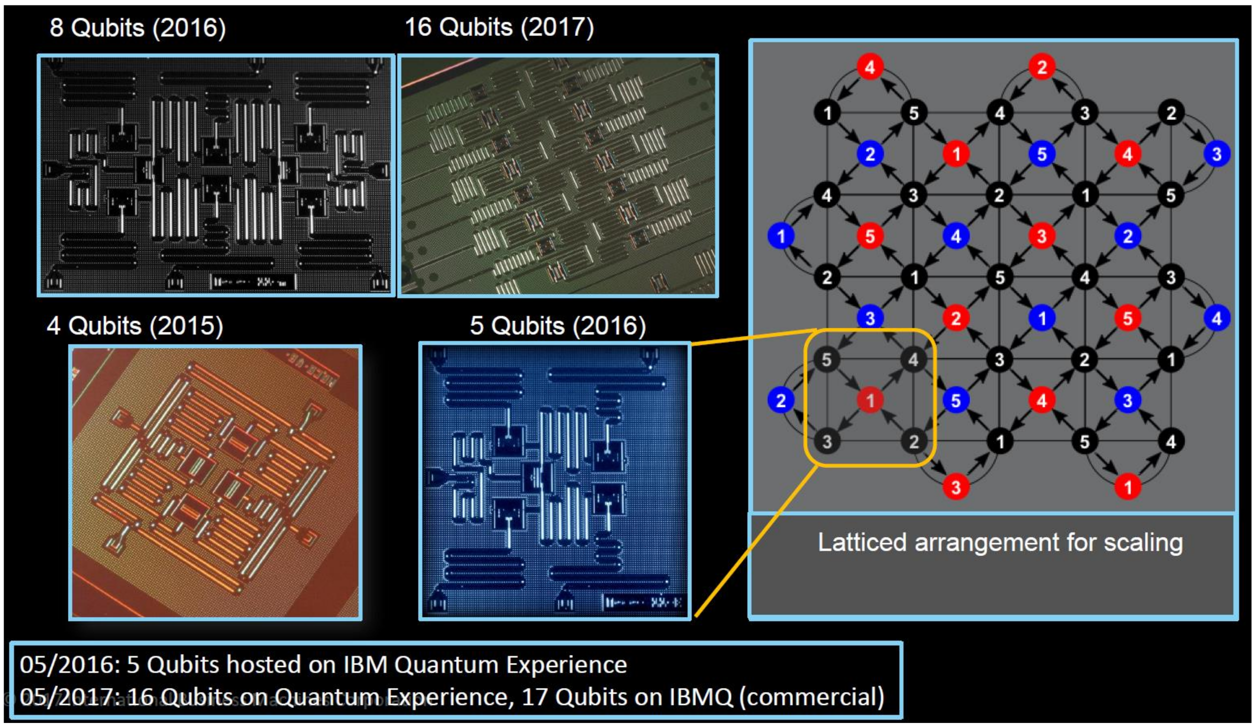

8 Qubits (2016)
16 Qubits (2017)
4 Qubits (2015)
5 Qubits (2016)
Latticed arrangement for scaling
05/2016: 5 Qubits hosted on IBM Quantum Experience
05/2017: 16 Qubits on Quantum Experience, 17 Qubits on IBMQ (commercial)

Figure 4. The Algorithms-to-Machines gap illustrates how well-known quantum computing algorithms (such as Shor's and Grover's algorithms have resource requirements that far exceed the qubit counts (shown in yellow) of systems we are able to build) and superconducting quantum computing setup.

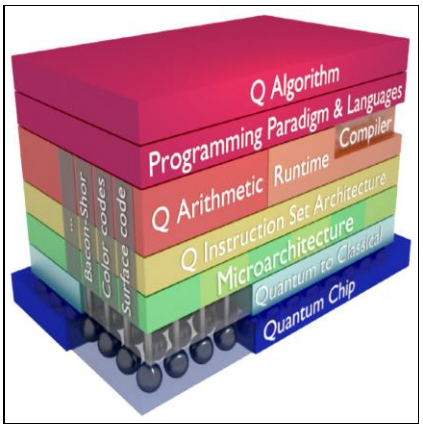
More importantly, we should try to broaden the range of available algorithmic techniques for quantum computers, which is presently quite limited. The first examples of problems that can solved faster with a quantum computer than with a classical computer were oracle, or black box, problems. In standard computational problems, the input is simply a string of data such as an integer or the description of a graph. In contrast, in the black box model, the computer given access to a black box, or oracle that can queried to acquire information about the problem. The goal is to find the solution to the problem using as few queries to the oracle as possible. This model has the advantage that proving lower bounds is tractable, which allows one to demonstrate provable speed-up over classical algorithms, or to show that a given QA is the best possible. There is a huge gap between the problems for which a quantum computer might be useful (such as chemistry problems, material science problems, etc.) and what we can currently build, program, and run.

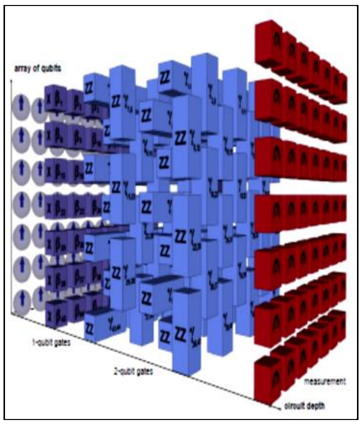
Remark. Quantum microarchitecture is a key component in bridging the gap between quantum software and quantum hardware of a fully programmable quantum computer. Useful quantum computation would require full stack architecture. To construct a universal and fully programmable quantum computer, clarifying the required component of a quantum computer is essential. As example, in the QuTech quantum computer system stack, multiple layers from top-level quantum algorithms to underlying quantum chips are included in a full stack quantum computer. In the middle of this system view, the Quantum Instruction Set Architecture (QISA) and the Quantum Microarchitecture layers play a key role in bridging the gap between quantum SW and quantum HW. Nowadays, research in quantum computer engineering has focused primarily at devising high-level programming languages and compilers, and building reliable low-level quantum HW. Relatively little studies have focused on how to control operations on experimental quantum processors. However, with Noisy Intermediate-Scale Quantum (NISQ) (cite Preskill 2018) quantum devices with 50 to hundreds of qubits coming soon, we need to start thinking about how to use the compiler output to fully control the quantum chip. To solve this problem, an experimental microarchitecture (QuMA) is proposed for controlling a superconducting quantum processor.

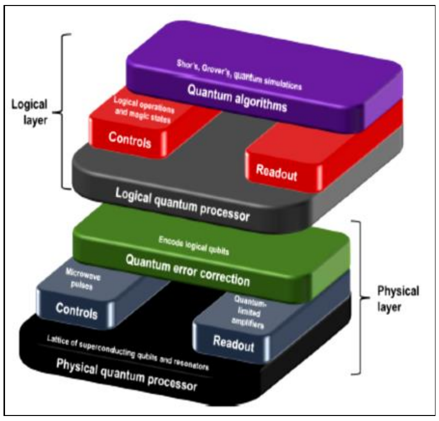
A quantum virtual machine is implemented on the basis of QuMAsim and other quantum software/emulators. OpenQL is a high-level quantum programming framework which has an eQASM back-end and generates executable code for the CC-Light. Therefore, the user can write his quantum algorithm in the OpenQL high-level language then compile it to generate the eQASM code. There is also some qubit state simulator like QX and QuantumSim, which can perform the quantum simulation.

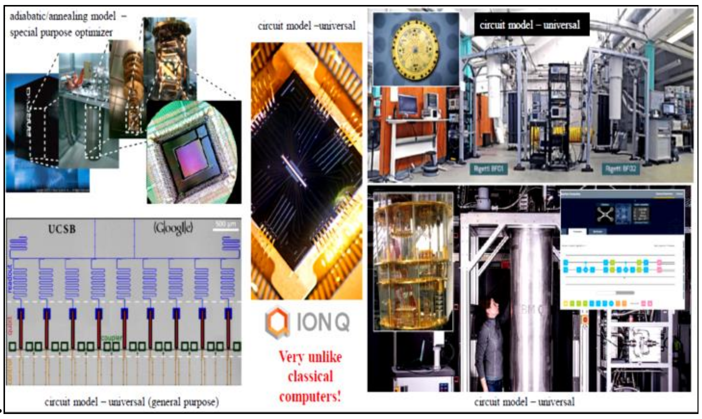
Quantum computer: Full stack

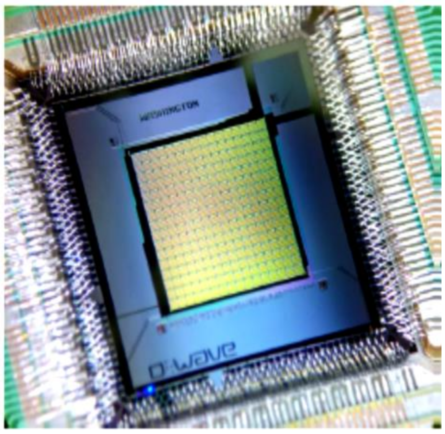
Figures 5 a, b provides a high-level view of the quantum system stack and success probabilities achievement for different types of quantum computers and software toolkits.

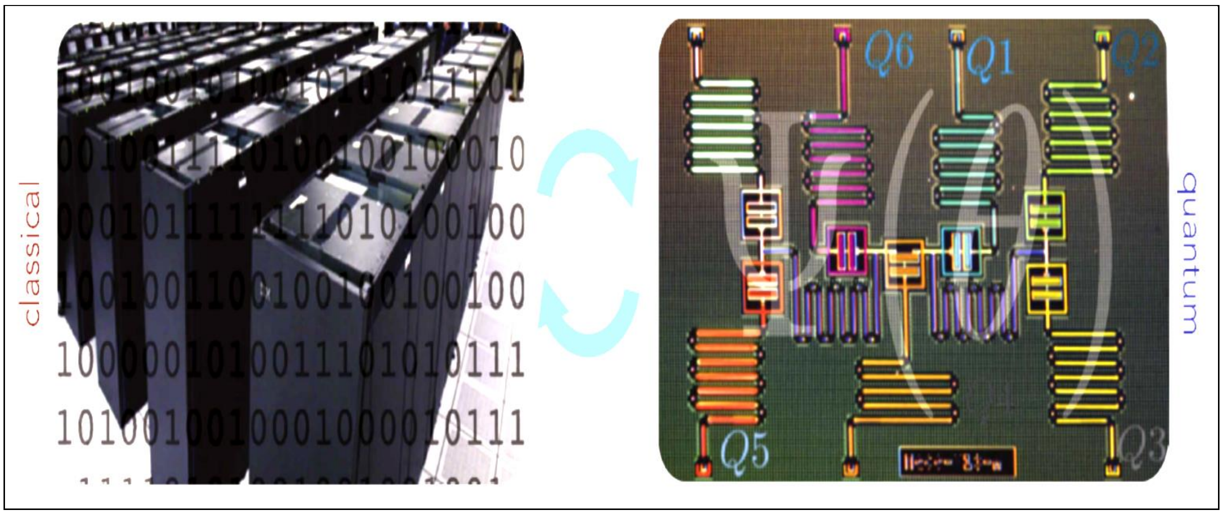


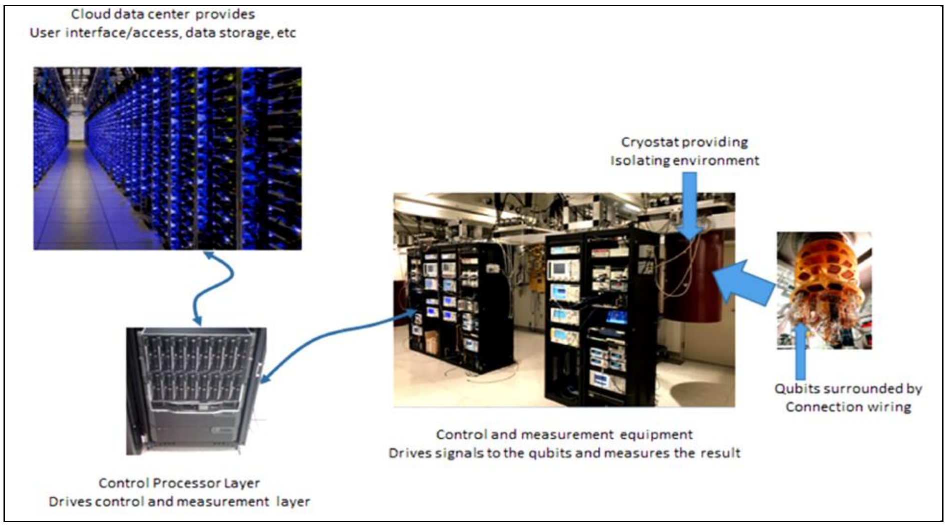



Logical layer
Shor's, Grover's, quantum simulations
Quantum algorithms
Logical operations and magic states
Controls
Readout
Logical quantum processor
Encode logical qubits
Quantum error correction
Microwave pulses
Controls
Quantum-limited amplifiers
Readout
Lattice of superconducting qubits and resonators
Physical quantum processor
Physical layer


adiabatic/annealing model – special purpose optimizer
circuit model –universal
circuit model – universal
UCSB
Google
IONQ
Very unlike classical computers!
circuit model – universal (general purpose)
circuit model – universal






Cloud data center provides
User interface/access, data storage, etc
Cryostat providing
Isolating environment
Qubits surrounded by
Connection wiring
Control and measurement equipment
Drives signals to the qubits and measures the result
Control Processor Layer
Drives control and measurement layer

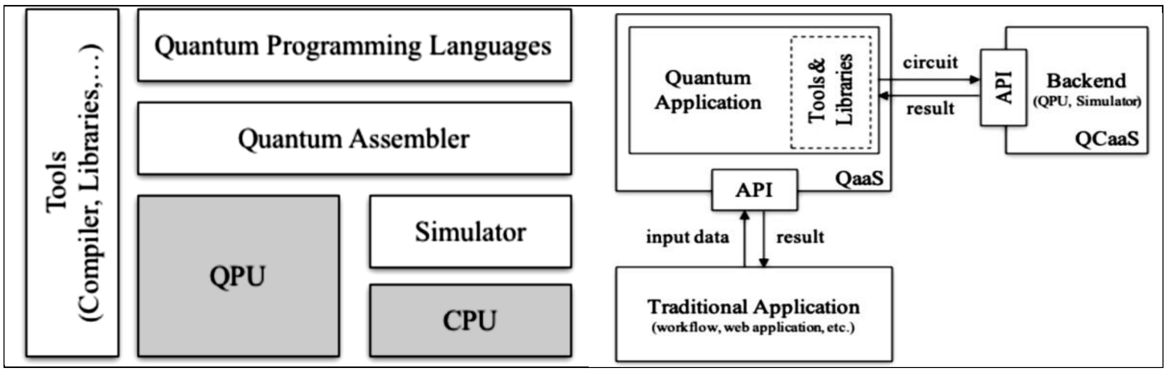
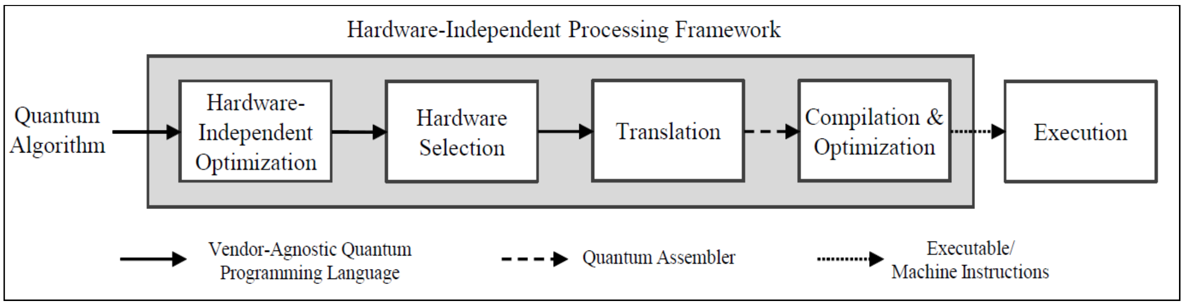
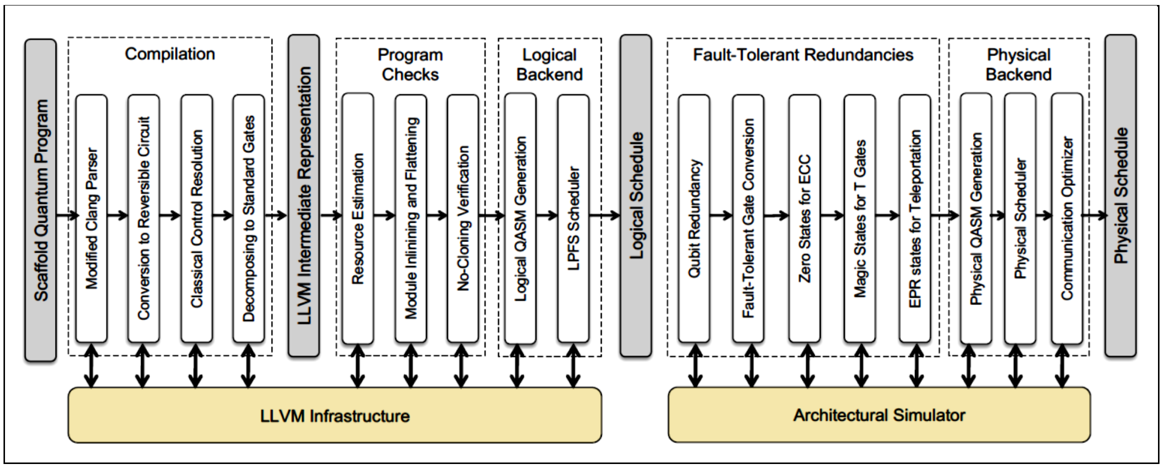


Figure 5a. Overview of quantum computer system stack and D-Wave 2X, 1098 qubits. Schematic of a hybrid quantum–classical computing architecture. Principle architecture of today's quantum software stack. Quantum Algorithm as a Service (QaaS) and Quantum Computing as a Service (QCaaS).


Hardware-Independent Processing Framework
Quantum Algorithm
Hardware-Independent Optimization
Hardware Selection
Translation
Compilation & Optimization
Execution
Vendor-Agnostic Quantum Programming Language
Quantum Assembler
Executable/ Machine Instructions



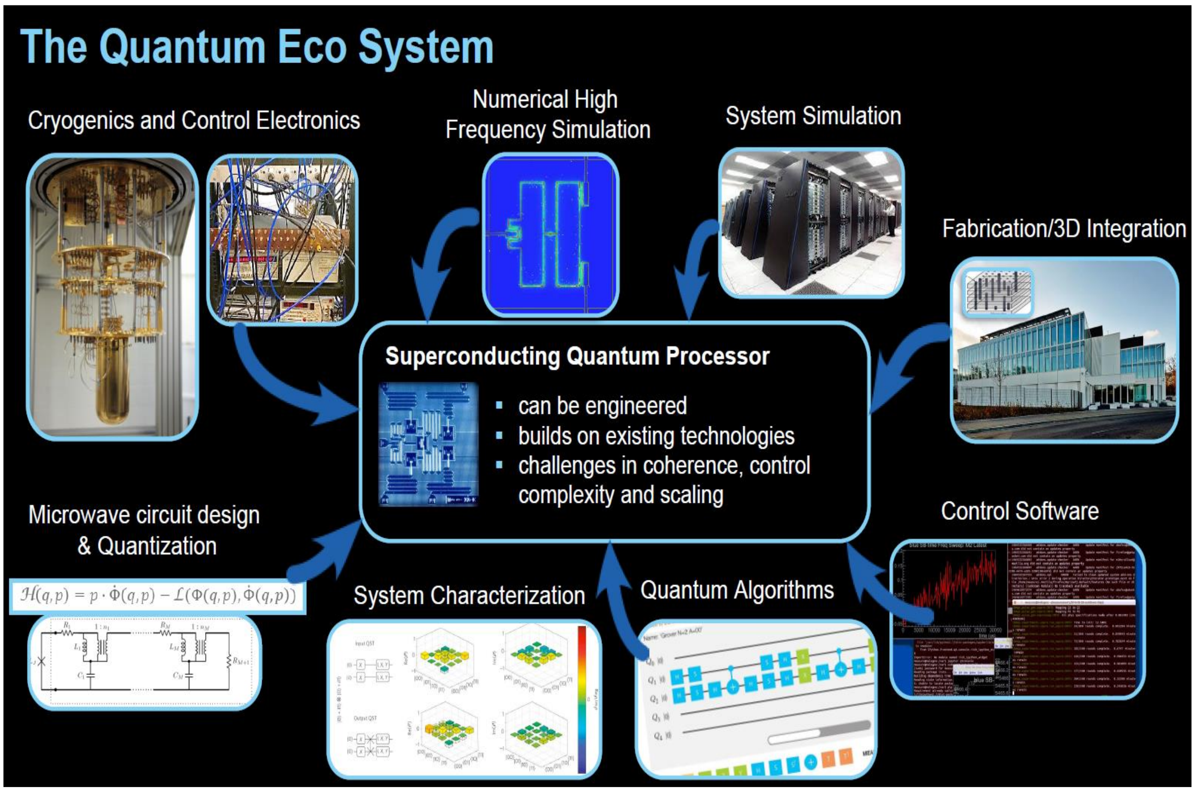

The Quantum Eco System
Cryogenics and Control Electronics
Numerical High Frequency Simulation
System Simulation
Fabrication/3D Integration
Superconducting Quantum Processor
can be engineered
builds on existing technologies
challenges in coherence, control complexity and scaling
Microwave circuit design & Quantization
$\mathcal{H}(q,p) = p \cdot \dot{\Phi}(q,p) - \mathcal{L}(\Phi(q,p), \dot{\Phi}(q,p))$
System Characterization
Quantum Algorithms
Control Software

Figure 5b. Processing of hardware-independent quantum algorithms

A quantum algorithm consists of a sequence of operations and measurements applied to a quantum processor. To date, the instruction set which defines this sequence has been provided by a classical computer and passed via control hardware to the quantum processor. Here, the first experimental realization of a quantum instruction set demonstrated, in which a fixed sequence of classically defined gates performs an operation that is fully determined only by a quantum input to the fixed sequence. Specifically, we implement the density matrix exponentiation algorithm, which consumes N copies of the instruction state to approximate the operation (an arbitrary angle). The implementation relies on a 99.7% fidelity controlled-phase gate between two superconducting transmon qubits. An average algorithmic fidelity 0.9 achieved, independent of the setting of fidelity, to circuit depth nearly 90. This new paradigm for quantum instructions has applications to resource-efficient protocols for validating entanglement spectra, principal component analysis of large quantum states, and universal quantum emulation.

Programmable computation, whether classical or quantum, consists of two fundamental components: an instruction set, and a machine to execute those instructions. For classical computation, there is no intrinsic

distinction between these components — the same physical instrument may be used both to generate and execute the instructions (Fig. 6 (A)). To date, the same has not been true for experimental demonstrations of quantum computing, whether in gate-based systems, quantum annealing, or one-way quantum computing. In conventional quantum computing applications, shown schematically in Fig. 6 (B), the instructions are programmed using classical resources and then delivered via hardware to a quantum processor that executes the instructions. In other words, the parity between instruction set and the processor executing the instructions is broken: one is fully classical, the other quantum.

An implementation of quantum instructions, in which a quantum state provides on-the-fly programming to a quantum computer (Fig. 6 (C)) demonstrated. In this approach, a fixed sequence of classically-defined gates form the scaffolding for a variable operation on a target system σ; an auxillary quantum state with density matrix ρ completes the encoding of the instructions. This hybrid approach to quantum programming partially restores the parity between the instructions and the processor in a quantum computer.

Quantum instructions have a variety of applications, including executing private quantum functions and quantum simulation. Quantum instructions are central to an efficient implementation of quantum emulation, which enables the implementation of an unknown unitary U with a finite set of known input-output relations

$\{\rho_{in}\}\xrightarrow{U}\{\rho_{out}\}$. Quantum emulation consumes fewer copies of the instruction qubits than would be sufficient for tomographic reconstruction, enabling the application of U to an arbitrary state without compromising the privacy of U itself. A quantum instruction set has also been theoretically proven to provide quantum speedups in quantum semi-definite programming. Additionally, if a Hamiltonian is encoded in the instruction state, quantum instructions enable sample-optimal Hamiltonian simulation.

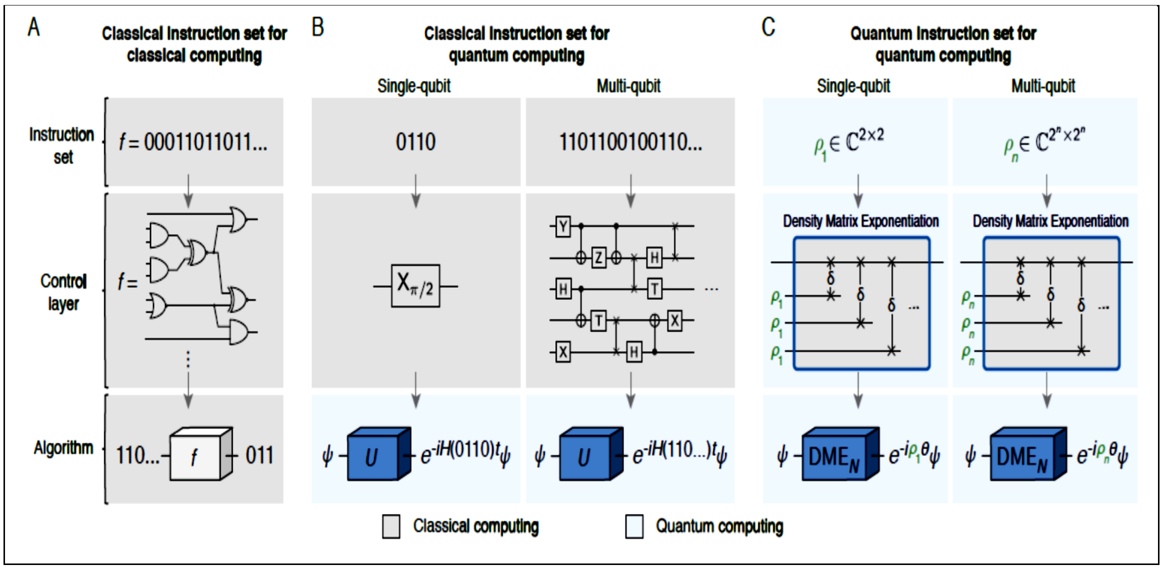

A
Classical instruction set for classical computing
B
Classical instruction set for quantum computing
Single-qubit
Multi-qubit
C
Quantum instruction set for quantum computing
Single-qubit
Multi-qubit
Instruction set
f = 00011011011...
0110
1101100100110...
$\rho_1 \in \mathbb{C}^{2\times 2}$
$\rho_n \in \mathbb{C}^{2^n\times 2^n}$
Control layer
f =
$X_{\pi/2}$
Density Matrix Exponentiation
Density Matrix Exponentiation
Algorithm
110...
f
011
ψ U $e^{-iH(0110)t}\psi$
ψ U $e^{-iH(110...)t}\psi$
ψ DME_N $e^{-i\rho_1\theta}\psi$
ψ DME_N $e^{-i\rho_n\theta}\psi$
Classical computing
Quantum computing

Figure 6. (A) Schematic representation of classical computing. Instructions are expressed by a classical function f defined by a bitstring '00011...', and is then executed on a dataset 110.... (B) Schematic representation of conventional quantum computing. The instruction set encoding a quantum circuit is generated using classical resources. A control layer generates the corresponding gate sequence, which is sent to the quantum hardware and implements the operation U = exp(−iH(0110) t). Here H(0110) is the Hamiltonian with parameters given by the bitstring 0110. (C) Quantum instruction set using the density matrix exponentiation (DME) algorithm. A single-qubit instruction (ρ_1 *) is used to implement the operator* $\mathrm{DME}_N(\rho_1, N, \theta) \approx e^{-i\rho_1\theta}$ *using a sequence of N partial SWAP operations, each supplied with a new copy of* ρ_1. *A multi-qubit program using quantum instructions shares the same structure of the classical gates.*

A quantum instruction set can be implemented efficiently using an approach called density matrix exponentiation (DME). DME consumes N copies of the quantum instruction density matrix ρ, and approximately performs the unitary gate $e^{-i\rho\theta}$, where θ is an arbitrary angle. It has been shown that DME asymptotically outperforms any tomographic strategy to implement $e^{-i\rho\theta}$. Without access to a quantum instruction set, implementing $e^{-i\rho\theta}$ necessitates a full tomographic construction of ρ. This in turn requires $O(d^2/\varepsilon^2)$ copies of ρ, where d is the dimension of the instruction system and ε is the desired precision. DME as implemented with quantum instructions requires only $O(\theta^2/\varepsilon)$ copies, resulting in an exponential reduction in resource requirements. This advantage makes DME a powerful platform to implement quantum operations based on quantum states, avoiding the need for classical learning of ρ. In addition to implementing quantum instructions, DME is a useful tool for using the target system to learn about the instruction set. In particular, a controlled-DME protocol combined with quantum phase estimation can be used to extract the dominant eigenvalues and eigenvectors of ρ, using only $O(\theta^2/\varepsilon)$ copies.

For intuition, if ρ is a single-qubit pure state, DMEN rotates σ about the axis defined by the Bloch sphere vector of ρ, through an angle θ. The physics of DMEN are related to the Trotterization of non-commuting Hamiltonians to perform quantum simulation. Dividing a quantum simulation into smaller steps reduces errors stemming from the Trotter approximation, and similarly, supplying DME_N with more copies reduces algorithmic error.

Gate-model quantum computers are theoretically capable of exceptional performance in certain applications, although it is unclear how useful they will be in general. The Quantum Approximate Optimization Algorithm (QAOA) of Farhi et al. has been proposed as a possible path towards making gate-model quantum computers effective at solving problems in combinatorial optimization. Recently, Rigetti Computing published results of QAOA run on their 19-qubit gate-model quantum computer. The inputs they considered can also be solved on D-Wave quantum annealing systems, providing an opportunity to compare the two quantum processing units (QPUs) directly. Reproducing their tests, it was found the probabilities of returning an optimal solution to be 99.6% for the D-Wave 2000Q and 0.001% for the Rigetti 19Q. In addition, the D-Wave 2000Q was able to solve 102 copies of the problem in parallel. The advantages in quality and size of the D-Wave 2000Q, taken together, provide an improvement of 10 million times in terms of ground-state throughput per sample (Fig. 7).

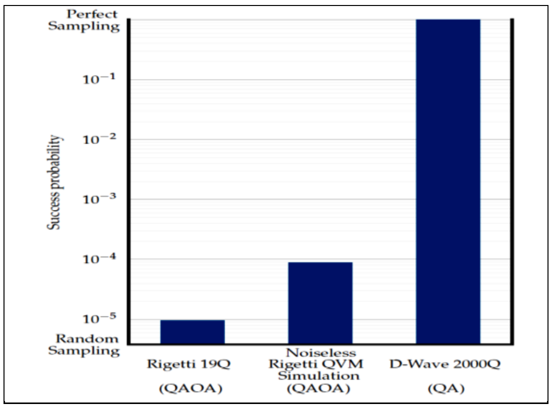


Figure 7. Success probabilities for Rigetti 19Q, Rigetti Quantum Virtual Machine (QVM) simulator, and D-Wave 2000Q on the 19-qubit input from Otterbach et al.

The integration of all these necessary ingredients into a full-stack QCCD (quantum charge-coupled device) quantum computer. The device is built around a microfabricated cryogenic surface trap (Fig. 8) containing five zones used for gating operations and ten storage zones.

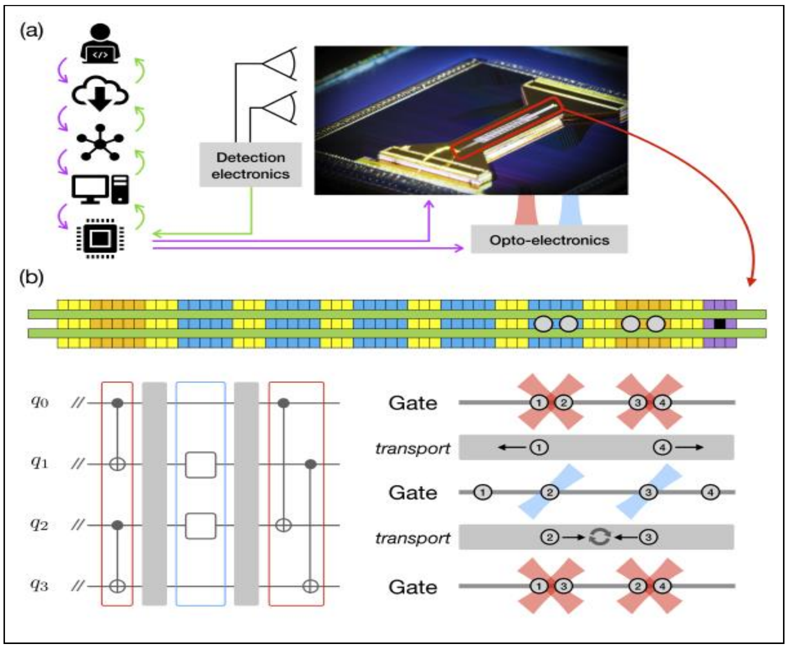


Figure 8. Illustration of the programmable QCCD quantum computing system along with a photograph of the trap. (a) On the right, a picture of the trap. On the left, the information flow from the user to the trapped qubits. From top to bottom we illustrate: user, cloud, internal tasking, machine control system, FPGA. The circuits are processed by a compiler as described in the text to generate control signals (purple) sent to both the trap electrodes as well as optoelectronic devices that control laser beams. An imaging system and PMT array collects and counts scattered

photons, and the result (green) is sent back to the software stack and user. (b) A schematic of the trap detailing: the load hole (black), load zone (purple), storage zones (orange), gate zones (blue), as well as auxiliary zones (yellow) for additional qubit storage.

The illustration of how a general quantum circuit is carried out shows that ions already sharing a gate zone are gated, then spatially isolated for single-qubit gates, then the second and third ions are swapped so that the final two-qubit gates can be executed. While not shown, readout, two-qubit gates, and single-qubit rotations can all be performed in parallel. Using ^{171}Yb+ and ^{138}Ba+ as the qubit and coolant ions, respectively, parallel operation and communication between two adjacent gate zones separated by 750 μ m demonstrated.

The software stack from the user down to the qubits in Fig. 8. The processor is programmed using the quantum circuit model. A quantum circuit is submitted remotely through a cloud-based service and tasked in Honeywell's internal cloud. The algorithm is compiled into the various primitives needed to execute the quantum circuit and sent to the machine control system. This system is responsible for programming the field-programmable gate array (FPGA) to execute the specific quantum circuit as well as scheduling and executing calibration routines. These automated calibrations are either executed on a predetermined time interval or triggered when a drift tolerance is exceeded. The FPGA handles the timing of operational primitives and real-time decision-making based on mid-circuit measurement outcomes. Clock synchronization between qubits is maintained via a phase-tracking protocol handled by the FPGA, which updates the qubit phases after transport and gate operations to account for phase accumulation generated by AC-Stark shifts.

In the rare event of ion-loss or detectable ion-reordering events, the data is discarded at the machine control level, and circuits are repeated as necessary to produce valid data. Finally, results are reported back through the cloud service to the user.

Quantum gate teleportation is a protocol in which a pair of maximally entangled qubits is used as a resource for applying a gate between a pair of remote data qubits. The protocol requires local entangling operations, mid-circuit measurements, and classically-conditioned quantum gates. The circuit for teleportation of the CNOT gate is shown in Fig. 9.

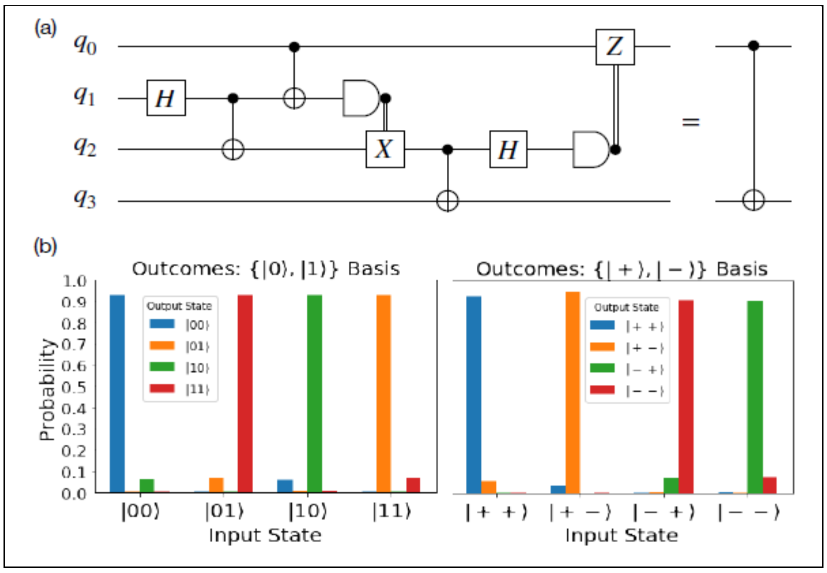


Figure 9. (a) Circuit implementing a teleported CNOT gate, with q_0 and q_3 the control and target qubits, respectively. (b) Bar plots showing the distribution of measurement outcomes when qubits q_0 and q_3 are prepared and measured in the $\{|0\rangle,|1\rangle\}$ and $\{|+\rangle,|-\rangle\}$ bases [2].

Here, qubits q_1 and q_2 are initially prepared in the Bell state $|\psi_B\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Two rounds of CNOT gates, each followed by a measurement and conditional gate, result in a circuit that is logically equivalent to a CNOT controlled on q_0 and targeting q_3 (see Fig. 9). To realize this circuit, transport operations are required to distribute the Bell state between two zones for remote gating. Each CNOT in the circuit is compiled into a native U_{zz} gate and single-qubit rotations.

To efficiently benchmark the teleported CNOT gate, the method for bounding the fidelity of a process from its action on two mutually-unbiased bases. This amounts to verifying the following quantum truth table:

$$CNOT: \begin{matrix} |00\rangle \to |00\rangle & |++\rangle \to |++\rangle \\ |01\rangle \to |11\rangle & |+-\rangle \to |+-\rangle \\ |10\rangle \to |10\rangle & |-+\rangle \to |--\rangle \\ |11\rangle \to |01\rangle & |-+\rangle \to |--\rangle \end{matrix}$$

where the states are labeled $|q_3q_0\rangle$. We prepare q_0 and q_3 in each state of the $\{|0\rangle,|1\rangle\}$ and $\{|+\rangle,|-\rangle\}$ bases, apply the circuit in Fig. 9 (a) and measure in the appropriate basis. Repeating the circuit 500 times for each input state, and randomize the order in which the eight variations are run. The data is shown in Fig. 9 (b).

Also notable are results for QAOA running on a classical simulation of a noiseless gate-model quantum computer. Even when running on a small, easy input using ideal hardware, QAOA success probabilities appear to be four orders of magnitude lower than D-Wave QPU success probabilities. This indicates that the single step of QAOA is insufficient. Running more steps of QAOA as necessary will require significantly higher coherence and lower error rates. The results do not provide evidence that QAOA will be practical for solving these problems on near-term gate-model devices.

The Rigetti 19Q is a gate-model quantum computer and the Rigetti QVM is a noiseless classical simulator, in this case simulating the 19Q. The Rigetti QPU and simulator both run the quantum approximate optimization algorithm (QAOA). The D-Wave 2000Q runs the quantum annealing algorithm (QA). The success probabilities from QAOA are at least 4 orders of magnitude lower than those from the D-Wave quantum annealing system.

This stack consists multiple layers. The bottom layers are quantum chip together with quantum to classical interface. These two layers are technology dependent and many implementations focus on improving the fragility of qubits. Different technologies have investigated like superconductors, trapped ions and nitrogen-vacancy centers. Among these technologies, superconducting qubits give the biggest potential for scalability. Besides physical qubits, logical qubits also draw attention with performance improved by quantum error correction. The middle layers are the control microarchitecture and Quantum ISA. QISA provides an interface to communicate SW and HW just like in classical architectures. QISA usually includes both quantum instructions and classical instructions since a quantum computer will always consist of both computing components.

These instructions are fed into control microarchitecture and decoded into required control signals with precise timing. These control signals processed by quantum to classical interface based on the specific quantum technology. These signals translated into required pulses and sent to quantum chip. The top layer represents quantum algorithms that described by high-level quantum programming languages like Scaffold and LIQ $Ui|\ \rangle$. These algorithms lack the low-level hardware information and it assumed that all operations can executed perfectly. Compilers help to provide this information so the quantum algorithms can execute by the potential quantum hardware. In compilation layer, these algorithms are compiled into a series of instructions that defined by QISA.

Controller microarchitecture is required to solve the control challenges in full stack quantum computer. There are many control difficulties need to be solved for quantum computer.

The first version of QuMA proposed and the diagram showed in Fig. 10.

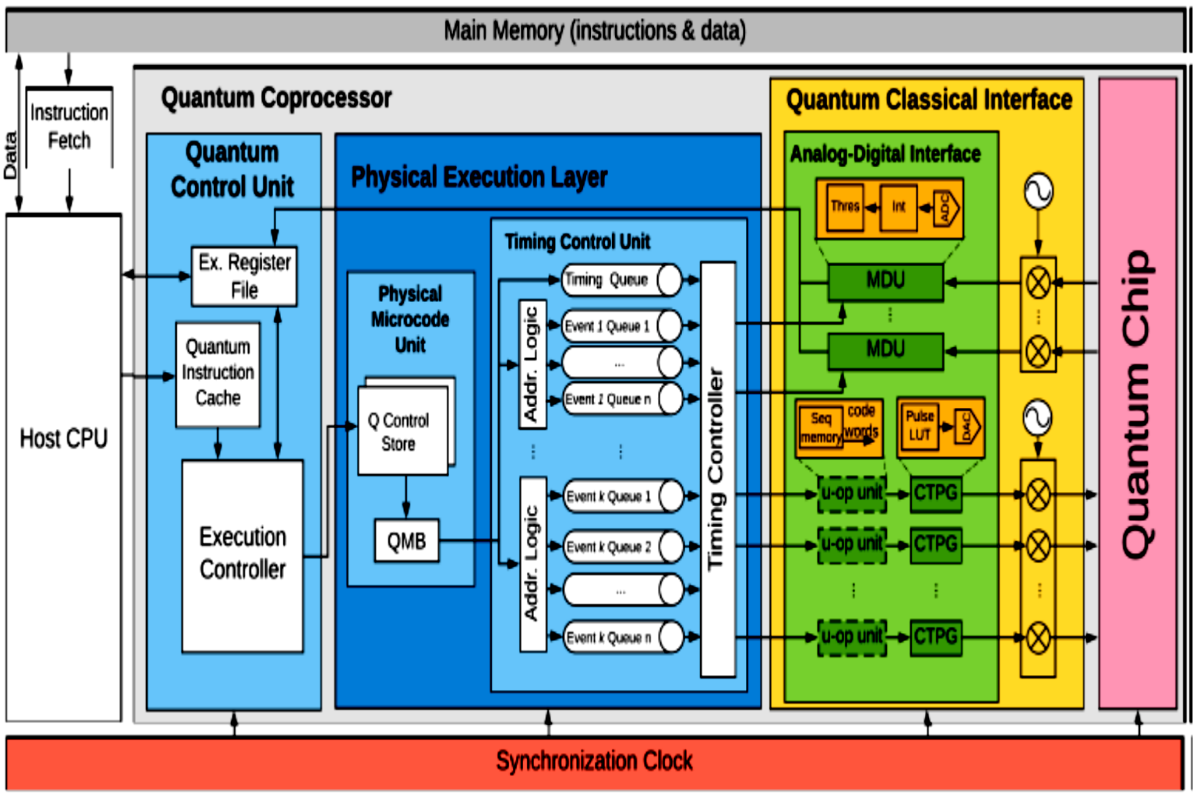


Figure 10. Overview of the structure of QuMA

QuMA is a heterogeneous architecture, which includes a classical CPU as a host and a quantum co-processor as an accelerator (Fig. 11).

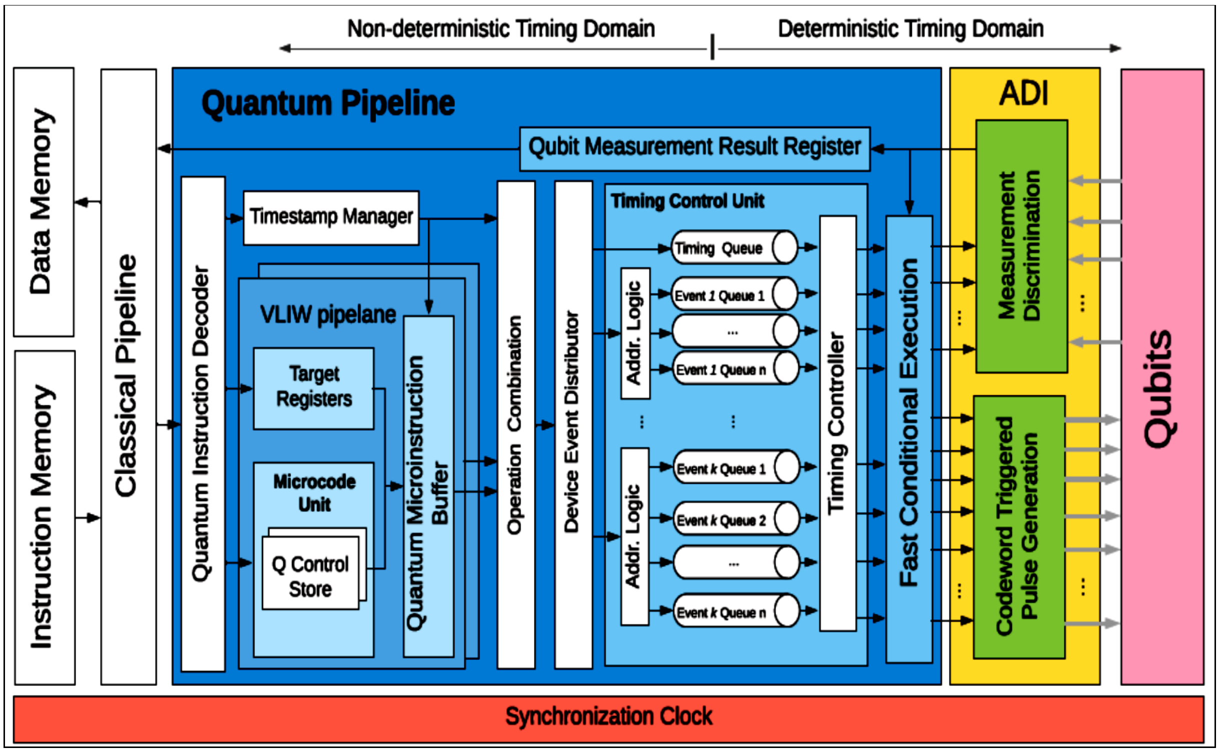


Figure 11. Quantum microarchitecture implementing the instantiated eQASM for the seven-qubit superconducting quantum processor.

QuMA accepts a binary file generated by the OpenQL compiler infrastructure. These instructions processed and generated micro operations at a deterministic timing. The analog-digital interface converts digital signals into corresponding analog pulses, which will perform quantum operations on qubits.

2. Hardware

Nowadays, there are various potential quantum technologies being pursued to implement qubits, such as trapped ions, quantum state in superconducting circuits, nuclear spins or electron spins in silicon, and nitrogen-vacancy centers. Other candidates like Majorana fermion based topological qubits (not yet built) are actively being researched as well. The currently most promising technologies are trapped ions and superconducting qubits, both of which have demonstrated to satisfy the DiVincenzo criteria.

Superconducting qubits Superconducting integrated circuits are Josephson junction based harmonic oscillators coherently controllable and measurable by magnetic flux pulses and microwaves. The performance of superconducting qubits benefits from the non-linearity of Josephson junctions and surrounding microwave circuitry. These systems have potentially excellent scalability since this technology have not encountered any hard-physical limits. Figure 12 show superconducting phase qubit.

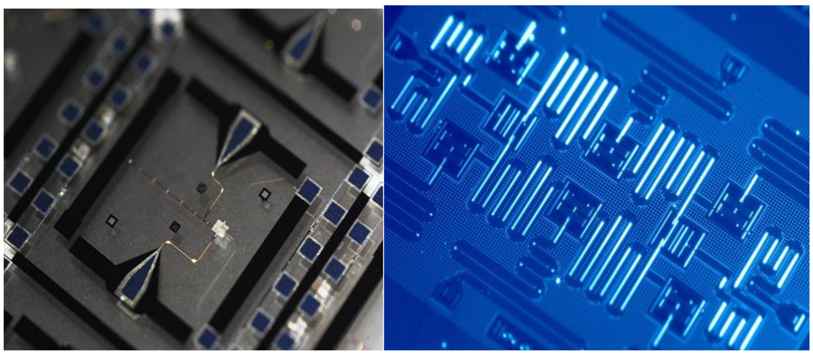

(a) *(b)*

Figure 12. Superconducting phase qubit from [3].

Superconducting qubits usually fabricated with well-established fabrication techniques such as photo and electron-beam lithography. However, superconducting qubits also suffer from drawbacks like low coherence times. This also makes quantum error correction very important for this type of system. Transmon is a promising technology, which achieves error rates lower than the fault-tolerance threshold for surface code. A universal gate set which comprised of single-qubit gates (mainly *x* and *y* rotations) and the CZ gate is used at here. The transmon is a lumped-element nonlinear *LC* resonator. The first-excited state used as the qubit. The transition frequency between these states can tuned by controlling the flux through the loop between the two Josephson junctions.

Figure 13 shows a prototype seven-port transmon developed in Dicarlo lab, Delft University of Technology.

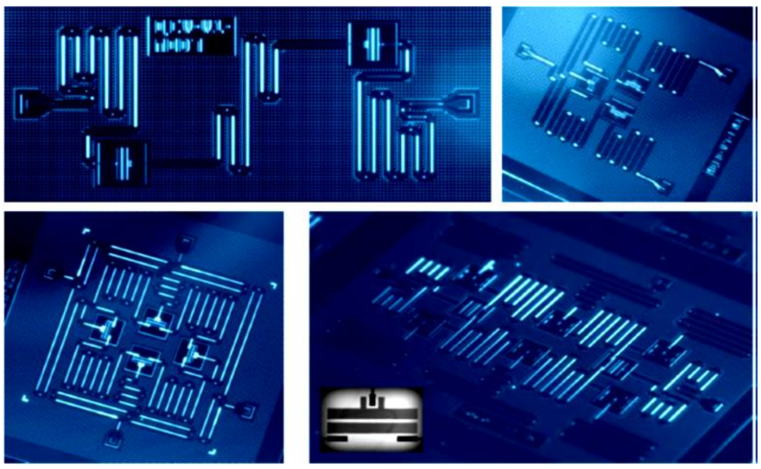


Figure 13. The prototype seven-port transmon.

The vertical I/O will be realized either using through-silicon through-silicon-visa or bump bonding in a flip-chip arrangement (Fig. 14).

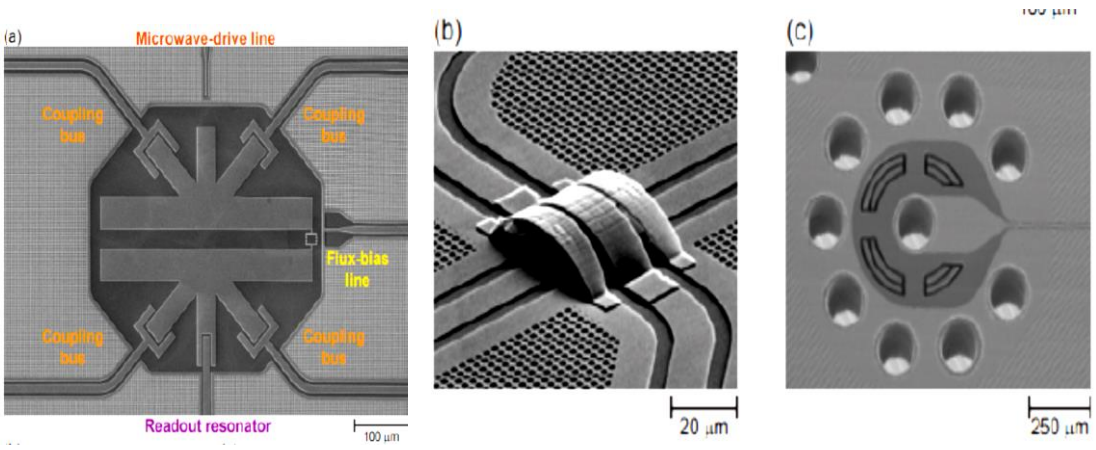
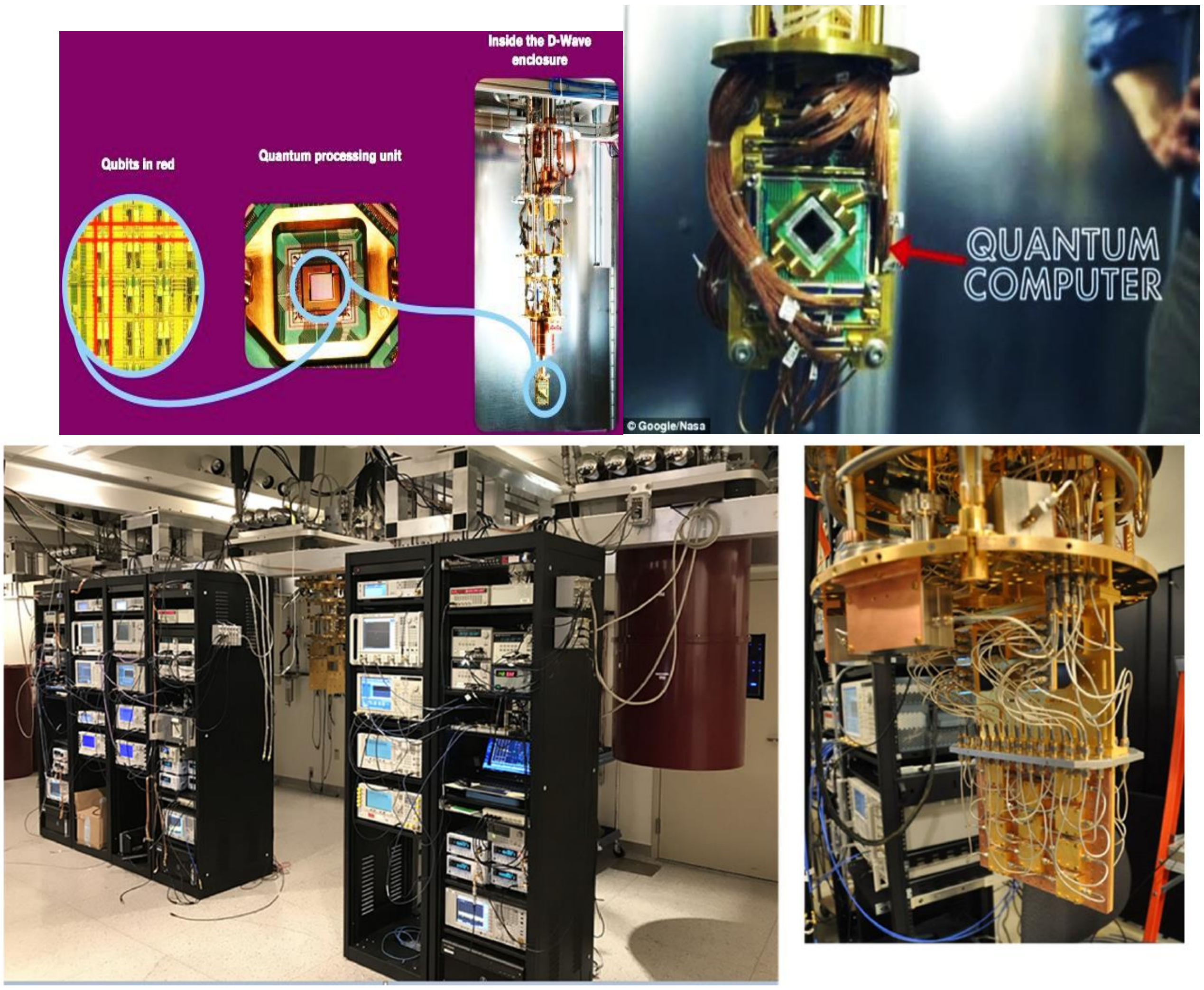
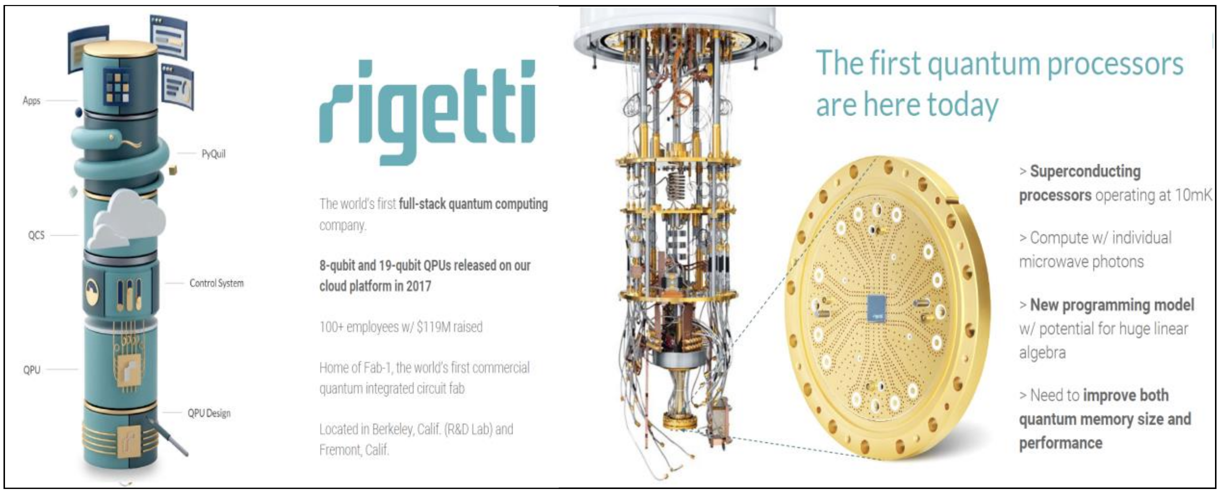


Figure 14. Images at various scales of implementation of surface-code fabric. (a) Starmon qubit. (b) Transmission-line crossover. (c) Vertical I/O.

Quantum computer of D-Wave Co. shown on Fig. 15.





rigetti

The world's first **full-stack quantum computing** company.

8-qubit and 19-qubit QPUs released on our cloud platform in 2017

100+ employees w/ $119M raised

Home of Fab-1, the world's first commercial quantum integrated circuit fab

Located in Berkeley, Calif. (R&D Lab) and Fremont, Calif.

The first quantum processors are here today

> **Superconducting processors** operating at 10mK

> Compute w/ individual microwave photons

> **New programming model** w/ potential for huge linear algebra

> Need to **improve both quantum memory size and performance**

Figure 15. D-Wave quantum computer and Rigetti quantum processor

On Fig. 16 an imaging objective collects ion fluorescence along the *Y*-axis and maps each ion onto a multichannel photo-multiplier tube (PMT) for measurement of individual qubits.

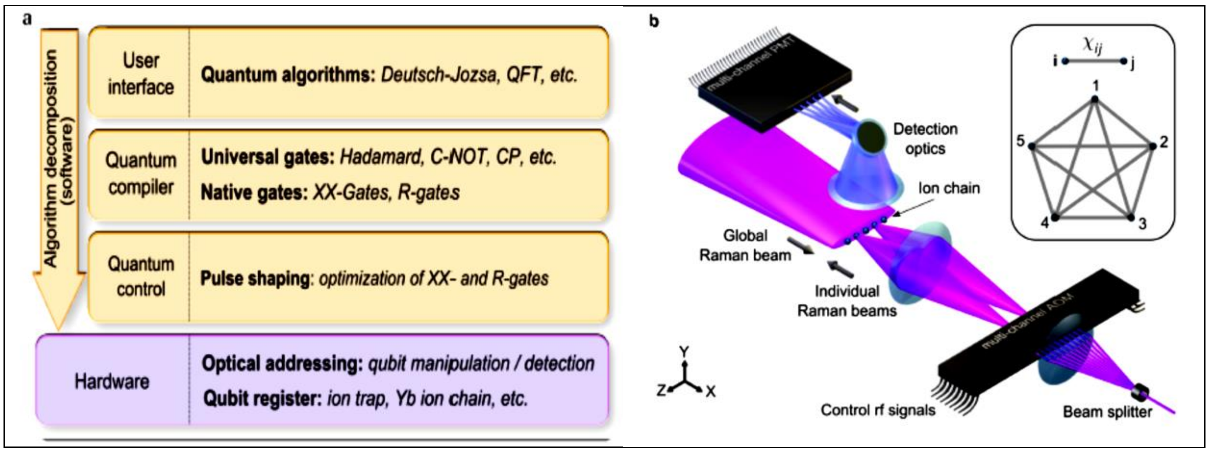

a
Algorithm decomposition (software)
User interface
Quantum algorithms: Deutsch-Jozsa, QFT, etc.
Quantum compiler
Universal gates: Hadamard, C-NOT, CP, etc.
Native gates: XX-Gates, R-gates
Quantum control
Pulse shaping: optimization of XX- and R-gates
Hardware
Optical addressing: qubit manipulation / detection
Qubit register: ion trap, Yb ion chain, etc.
b
multi-channel PMT
Detection optics
Ion chain
Global Raman beam
Individual Raman beams
multi-channel AOM
Control rf signals
Beam splitter
Y
Z
X
X_{ij}
i
j
1
2
3
4
5

Figure 16. Computation architecture. (a) Decomposition of algorithms from the user interface and software operations to the physical hardware. (b) Hardware setup. A linear chain of trapped ion qubits along the Z-axis is shown at the center of the figure.

Counter propagating Raman beams along the *X*-axis perform qubit operations. A diffractive beam splitter creates an array of static Raman beams that individually switched using a multi-channel acousto-optic modulator (AOM) to perform qubit-selective gates. By modulating appropriate addressing beams, any single-qubit rotation or two-qubit Ising (*XX*) gate can be realized (Fig. 17).

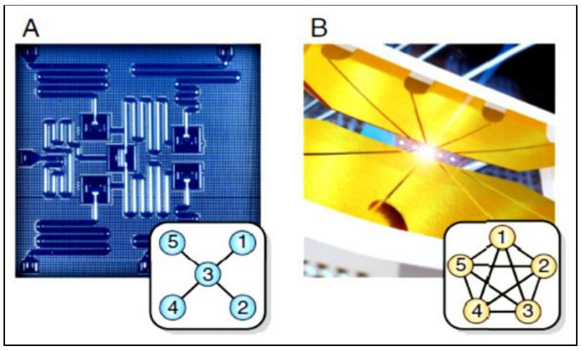


Figure 17. Two qubit Ising (XX) gate [4].

The architecture on Fig. 18 involves only one classical computer, which is the host of the digital image (original and classical), the cloud accessed via Internet, and the corresponding QPU. It is clear that for this architecture it is not necessary that the classical computer and the QPU are in the same location. The key lies inside the classical computer.

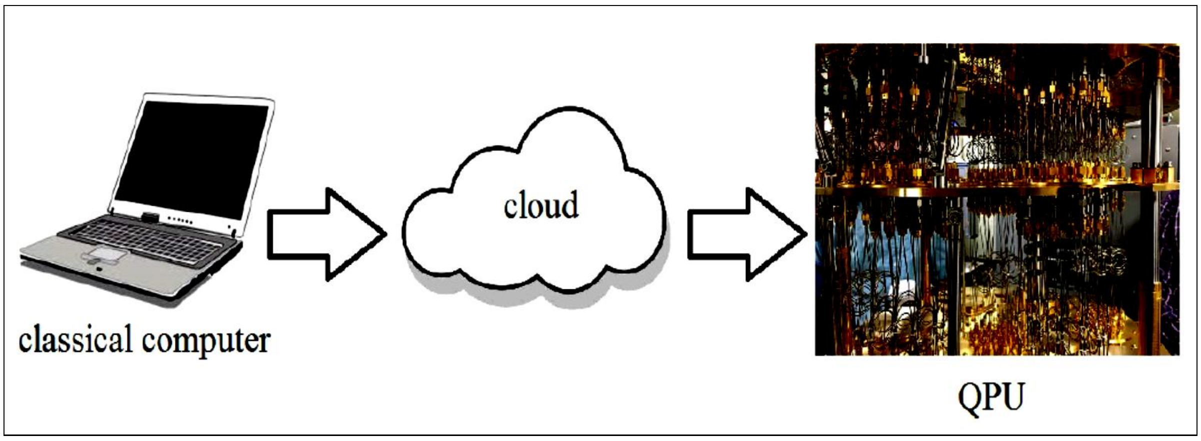


Figure 18. Architecture: classical computer + cloud + QPU of IBM Q or Rigetti

Example: *Quantum Image Processing*. Two sets of implementations on quantum platforms will be presented:

a) the first set consists of a strict comparison of performance between FRQI, NEQR and QBIP on the Rigetti platform, specifically on its Quantum Virtual Machine (QVM), and which consists of taking an image of the classical world, representing it with the technique, and then returning it again to the classical world to evaluate the impact that the technique had on the morphological integrity of the image, i.e, without quantum algorithm, only the technique; b) while the second set will consist of exclusive QBIP implementations for several quantum algorithms, since it is the only technique with which efficient implementations can be made on IBM Q, Rigetti, Quantum Programming Studio, Quirk, among many others. Three techniques of internal image-representation in a quantum computer are compared: Flexible Representation of Quantum Images (FRQI), Novel Enhanced Quantum Representation of digital images (NEQR), and Quantum Boolean Image Processing (QBIP). All conspicuous technical items are considered in this comparison for a complete analysis:

i) performance as Classical-to-Quantum (Cl2Qu) interface,

ii) characteristics of the employed qubits,

iii) sparsity of the used internal registers,

iv) number and size of the required registers,

v) quality in the outcomes recovering,

vi) number of required gates and its consequent accumulated noise,

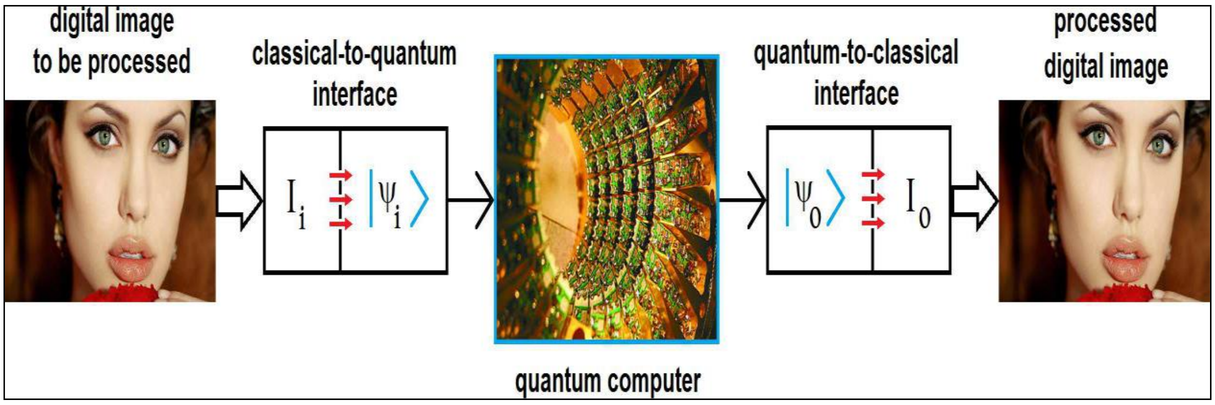
vii) decoherence, and

viii) fidelity.

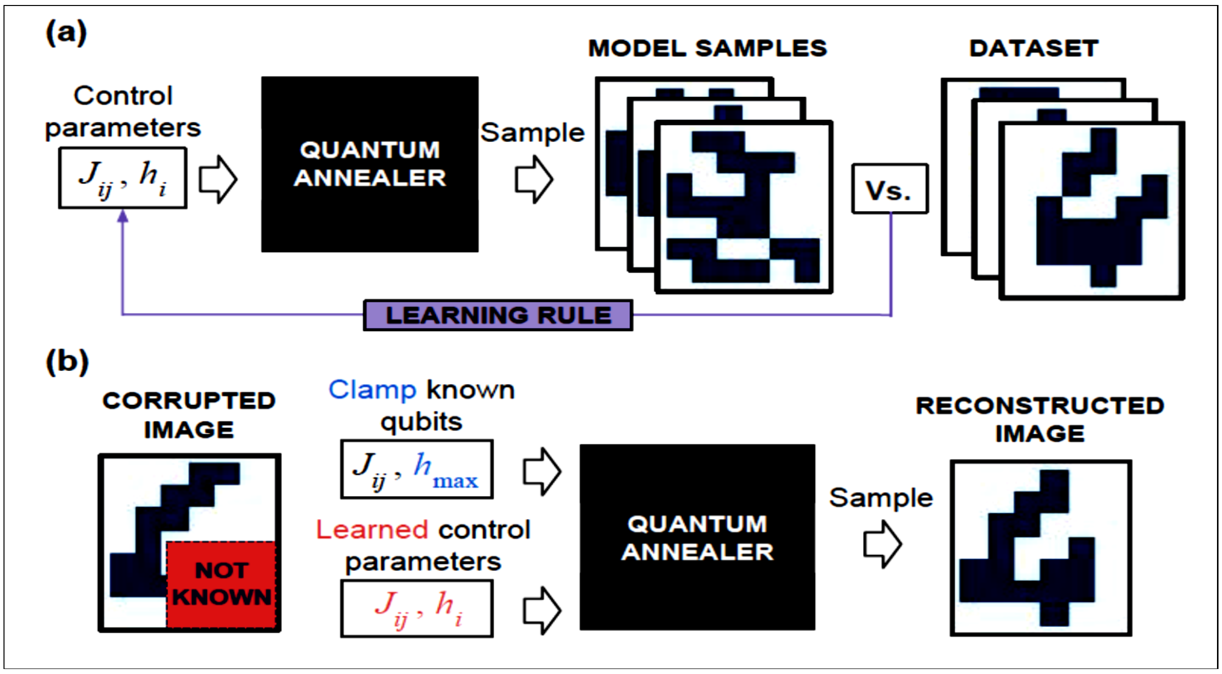
These analyses and demonstrations are automatically extended to all variants of FRQI and NEQR.

This study demonstrated the practical infeasibility in the implementation of FRQI and NEQR on a physical quantum computer (QPU), while QBIP has proven to be extremely successful on: a) the four main quantum simulators on the cloud, b) two QPUs, and c) optical circuits from three labs. Moreover, QBIP also demonstrated its economy regarding the required resources needed for its proper function and its great robustness (immunity to noise), among other advantages, in fact, without any exceptions.

A typical scheme of QImP necessarily implies an architecture like the one presented in Fig. 19.


digital image to be processed
classical-to-quantum interface
I_i $|\psi_i\rangle$
quantum computer
quantum-to-classical interface
$|\psi_0\rangle$ I_0
processed digital image

(A)



(B)

Figure 19. (A) Scheme of Quantum Image Processing (QImP) [6]. (B) Example of annealing-based generative models for binary images. (a) Learning consists of adjusting control parameters so that the generated samples become similar to those in the dataset. (b) Inference consists of using the learned model to sample parts of a corrupted image and obtain a reconstruction.

Evidently, this scheme does not differ much from that used by quantum computing for any other task. Figure 19 shows that the classical image (digital) must be introduced into the quantum computer for further processing, which is the responsibility of a quantum algorithm allocated inside a quantum computer. Image processing consists in the filtering of the noise that the digital image brings from its classical origin, e.g., a camera, the channel through which it was received, etc.

Consequently, the quantum algorithm will have the function of filtering said noise. The problem is to introduce a quantum version of the original classical image into the quantum computer. There are two possibilities to achieve this: a) by hand, i.e., preparing qubit by qubit in an artisanal way, or b) automatically, using a Classical-to-Quantum interface (Cl2Qu).

Now, suppose that the camera takes color pictures at a resolution of 1920x1080 pixels (i.e., a common camera), so, each digital image will be composed of 1920x1080x3x8 bits (approximately 50 Million bits), which should be converted into approximately 50 Million of qubits inside the quantum computer. If we analyze this real case in order, option (a) is not viable, since no laboratory in the world is able to prepare 50 Million qubits, let alone there are no humans who would do this artisanal imitation work, i.e., convert each one of the 50 Million of bits in its quantum counterpart. This is obviously unthinkable. Regarding (b), we find the following options according to the QImP literature, i.e., via:

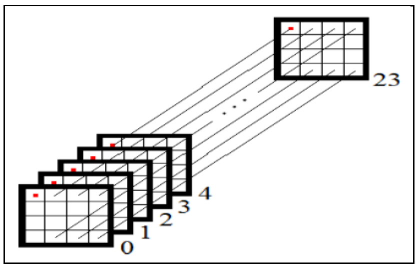
I. FRQI (and its variants),

II. NEQR (and its variants) [3], and a third option,

III. which are constituted by a family of proven Cl2Qu interface, and that are used by QBIP.

Next, postulate the essential and inexcusable conditions that must be met by any Cl2Qu interface in order to be used on a QPU like Rigetti or IBM Q Experience. Therefore, with this information, we can evaluate the performance as Cl2Qu interface of the three techniques suggested above. What is clear even from this precise moment is that any effort to implement a QImP scheme is absolutely unfeasible without a real and practical Cl2Qu interface.

Quantum Boolean Image Processing (QBIP) This technique consists of a pair of interfaces, i.e., Cl2Qu: $bit \rightarrow [C12Qu] \rightarrow |bit\rangle \equiv qubit$, and Qu2Cl: $qubit \equiv |bit\rangle \rightarrow [Qu2Cl] \rightarrow bit$, with a quantum algorithm between them. QBIP strictly respects the configuration stablish in Fig. 19, where the three mentioned elements work strictly and exclusively with CBS $\{|0\rangle, |1\rangle\}$, not being altered by the quantum measurement. If we have a Lena's color image like that of Fig. 20 (a), and discompose it in its 24 bitplanes, then, we will obtain Fig. 20 (a), which shows a simplified detail of this procedure for a tile of only 4-by-4 pixels of Fig. 20 (b) (for the purpose of not complicating the drawing), which is carried out by a *bit-slicer*() function implemented on a classic computer.



(a) (b)

Figure 20. a) Lena's color image of 128-by-128-by-3-by-8 bits. b) Slicing of 4-by-4 pixels of the original image in its 24 bit-planes.

The action of the mentioned *bit-slicer*() function has a counterpart, it is the *bit-reassembler*() function, which allows us to reconstruct the image from its bitplanes.

The joint action of both functions can be seen in Fig.21.

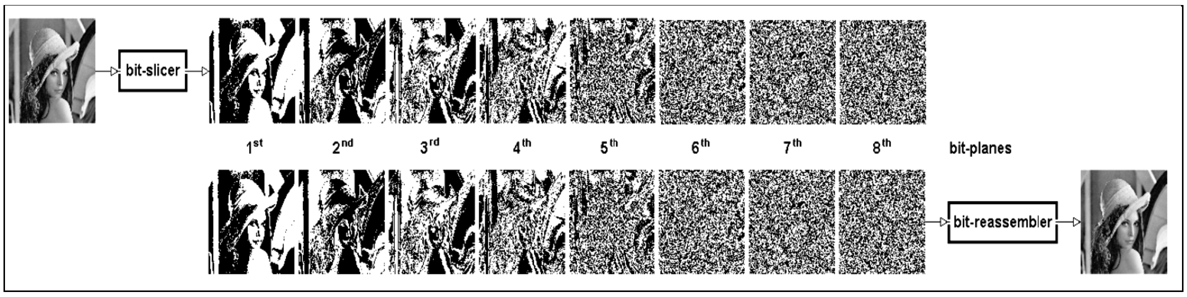


Figure 21. The combined action of the bit-slicer() and bit-reassembler() functions on Lena's red channel.

However, in order not to complicate said figure, we only represent the 8 bit-planes of the red channel. QBIP only works with the most significant bit (MSB) of each color channel of the original image, as we can see in Fig. 22 (where only it is represented the MSB of the red channel), which for an image of *R*Ч*C*Ч*CoC*Ч*BpPpC* bits (where, *R* is the number of rows, *C* the number of columns, *CoC* means *channels-of-color*, which they always are 3, and *BpPpC* is the number of *bit-per-pixel-per-channel*, which are generally 8) we can only work with *R*Ч*C*Ч*CoC* bits, dramatically lowering the storage respect to FRQI and NEQR at least 8 times.

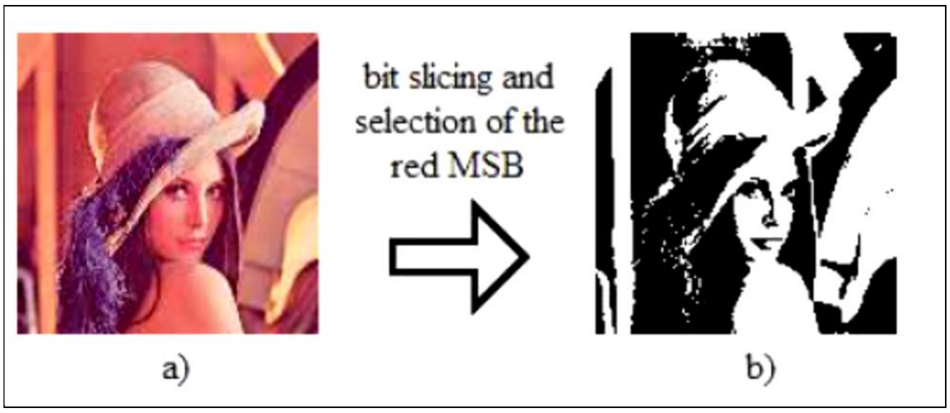


Figure 22. a) Lena's color image of 128-by-128-by-3-by-8 bits. b) Lena's MSB bitplane of the red channel with 128-by-128-by-1 bits.

It is precisely and exclusively on the mentioned MSBs (of the respective three-color channels) that the quantum algorithm acts.

Figure 23 shows, from left to right:

- the original image to be treated,

- the FRQI version after applying this technique and the posterior quantum measurement of them respective outcomes. It is evident that the quantum measurement eliminates the gray levels, converting each pixel to a strict CBS {0,1}, i.e., destroying the original image,

- the NEQR version, with three types of outcomes (and all its possible combinations), which destroys the morphology of the image as a result of the devastating effect of entanglement coupling.

Finally,

- the QuBoIP version, where we have applied the technique to the 8 bitplanes of the image, i.e., not just to the first bitplane (MSB) as it is usual in this technique. The outcomes clearly show the total absence of entanglement between them, as well as, the complete immunity to alterations related to quantum measurement and entanglement coupling.

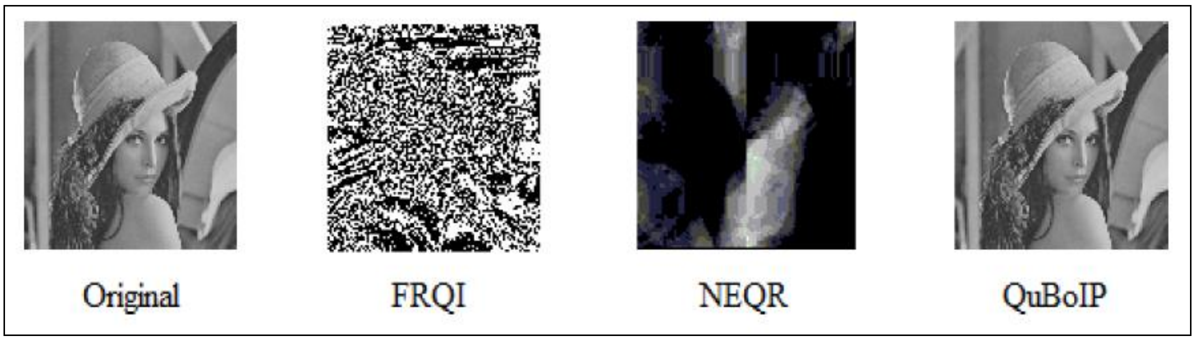


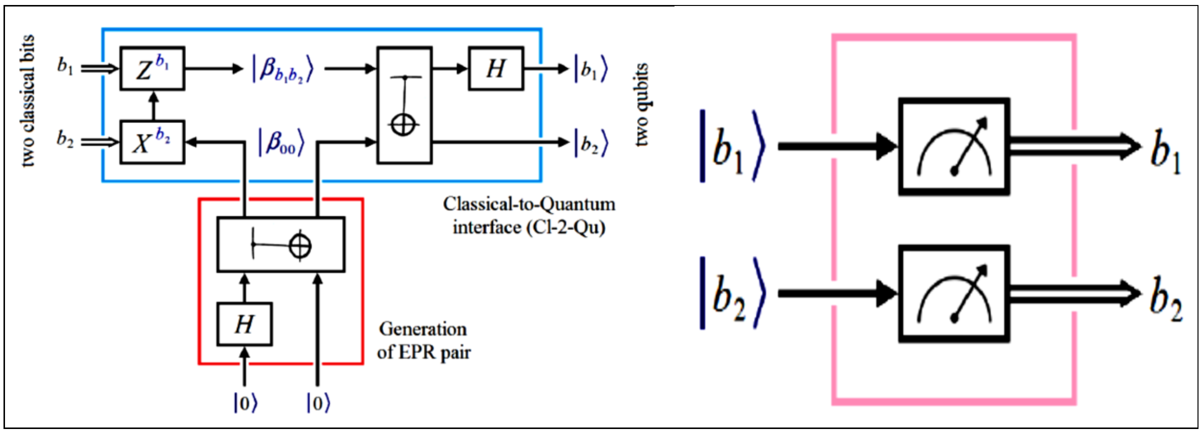
Figure 23. Results of the first experiment. From left to right, in order, we have: the original image, the FRQI version after quantum measurement, the NEQR version in terms of the entanglement between some of their outcomes and their respective allocation, and the QuBoIP version after its treatment and quantum measurement.

Finally, we show another important result of the experiment for the three techniques: the elapsed time each one takes,

- FRQI → 819 minutes,

- NEQR → 714 minutes, and

- QuBoIP → 68 minutes, taking into account all the bitplanes of the image + a Classical-to-Quantum interface + a simple quantum measurement module as a Quantum-to-Classical interface, Fig. 24.



(a) *(b)*

Figure 24. Example of a possible pair of Cl2Qu and Qu2Cl interfaces for QBIP extracted from the superdense coding protocol (a); Quantum-to-Classical (Qu2Cl) interface (b).

This result was reached coding in pyquil 2.6 on the QVM of Rigetti on an Intel Core i7-4702 MQ

CPU @ 2.20 GHZ/2.20 GHZ with 8.00 GB RAM on a 64-bits operating system, Windows 7 Ultimate. This demonstrates that when working exclusively with CBSs, virtually anything can be used as a Cl2Qu interface.

3. Software

SW is a simulation platform to aid in the exploration of quantum computing. Currently, there are three classes of simulators built into the system representing different levels of abstraction:

1. Physical Modeling: This is the Hamiltonian simulator, which attempts to model some of the actual physics in a quantum system. It differs from the other simulators in that it has the concept of the time it takes for an operation to be performed (since it is numerically solving a differential equation) and can only operate on a small number of qubits (around 30). It is also (by its very nature) slow.

2. Universal Modeling: This is the most flexible of the simulators. It allows a wide range of operations to be performed (including ones defined by the user). It can handle millions of operations (gates) to be executed, is highly optimized for parallel execution and is highly efficient in memory usage. Its main limitation is the number of qubits (~30) that can be entangled at one time.

3. Stabilizer Modeling: This simulator has the virtue of allowing large circuits (millions of operations) on massive numbers of qubits (tens of thousands). The main limitation is the types of gates, which may be included in the circuit. They are fixed in the system and come from the “stabilizer” class (e.g., Clifford group). This limits the usefulness of the types of algorithms that can be implemented and tested. However, it does allow the design and test of Quantum Error Correction Codes (QECC) which requires large numbers of qubits per logical qubits.

Simulations can accomplish in several ways:

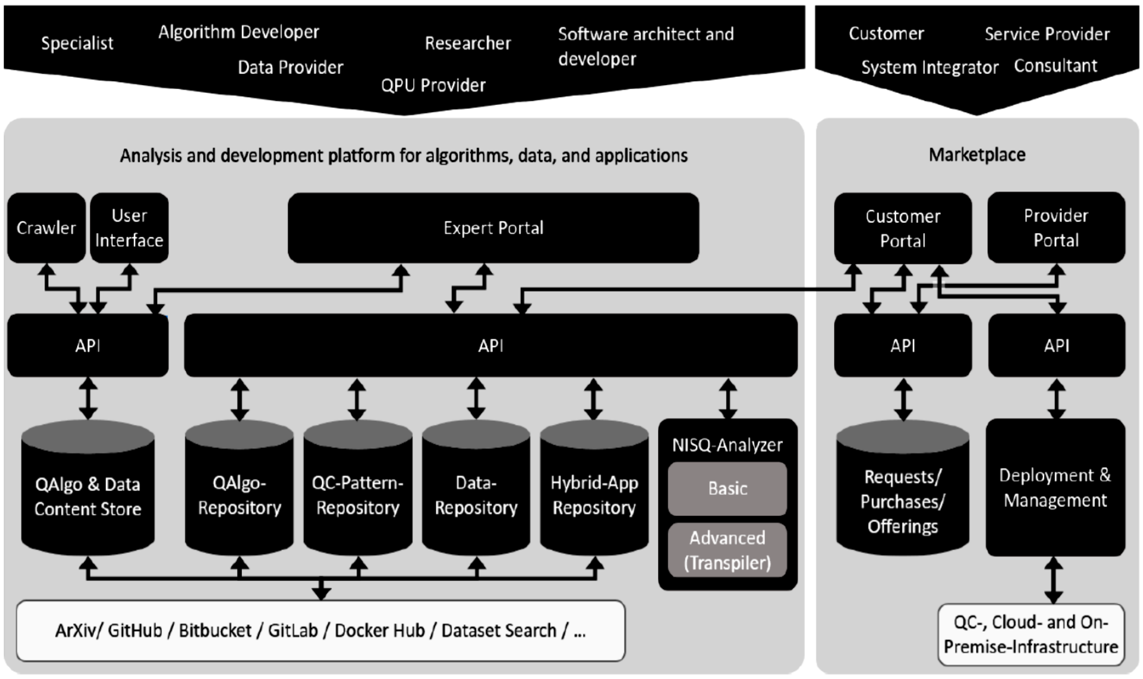
1. **Test mode**: Several built-in tests of the system can be invoked from the command line and are useful demonstrations.

2. **Script mode:** The system can be run directly from an F# text script (.fsx file). This allows the simulator to be operated by simply running the executable (no separate language compilation required). The entire simulator is available from this mode, but interactive debugging is difficult. Script mode allows users to experiment (with fast turn-around time) as well as being able to “kick the tires” without having to install a complete development environment.

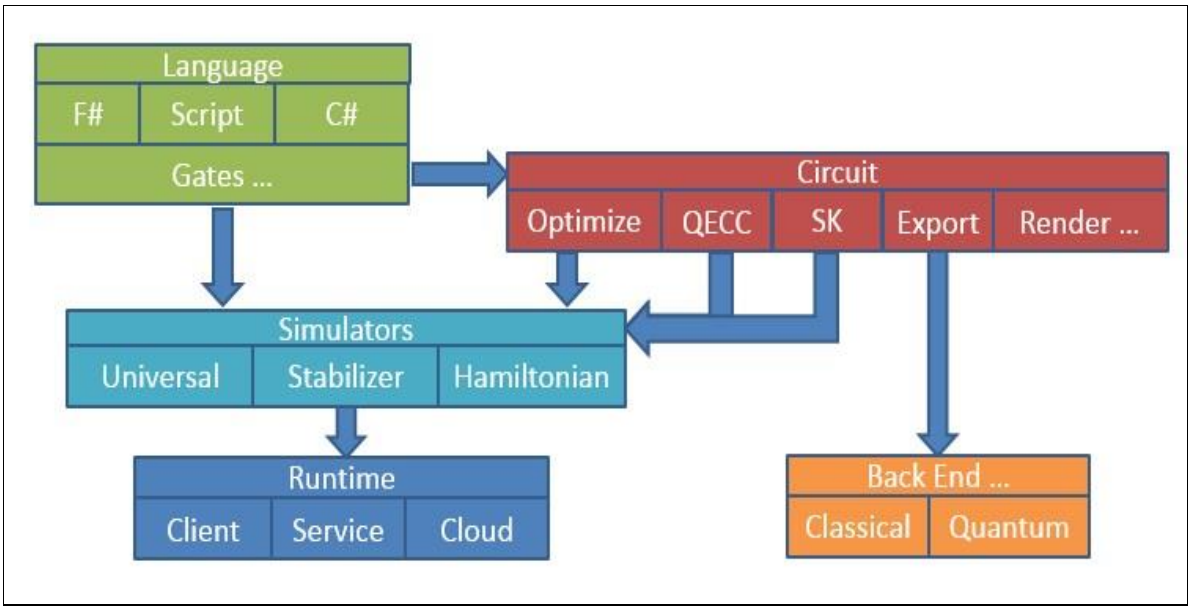
3. **Function mode:** This is the normal development mode. It requires a compilation environment (e.g., Visual Studio) and the use of a .Net language (typically F#). The user has the full range of APIs at their disposal and can extend the environment in many ways as well as building their own complete applications.

4. **Circuit mode:** Function mode can be compiled into a circuit data structure that is extremely general. This data structure can be manipulated by the user, run through built-in optimizers, have quantum error correction added, rendered as drawings, exported for use in other environments and may be run directly by all the simulation engines.

The entire architecture summarized in Fig. 25.



Architecture for a collaborative quantum software platform



The |LIQU⟩ Platform Architecture

Figure 25. The entire architecture

Here are each of the major sections.

Any methods of quantum global optimization and of quantum learning based on quantum neural networks can be described as modified models of quantum search algorithm that by L. Grover was developed. This approach can also use for design optimisation of robust intelligent control based on Quantum Soft Computing. The idea of Quantum Soft Computing application for design of robust intelligent control introduced. The structure of quantum soft computing includes the Quantum Genetic Search Algorithm (for global optimisation of control laws) and Quantum Neural Network (for robust approximation of teaching control signals) based on mutual applications of quantum and genetic search algorithm's operators. The background of this structure is Grover's quantum search algorithm.

We describe how to build a classical hardware (HW) device, which accelerates the simulation of quantum algorithms (QA) on classical computer. The usual approach for so doing consists in the simulation of the either QAs and their underlying quantum systems (see Figs 26, 27 and 28). The main aim of this Part 2 is not to work on real quantum HW (as quantum dots, ion traps, NMR etc.) but to take quantum computing as a computation paradigm (alternative to classical computing and soft computing).

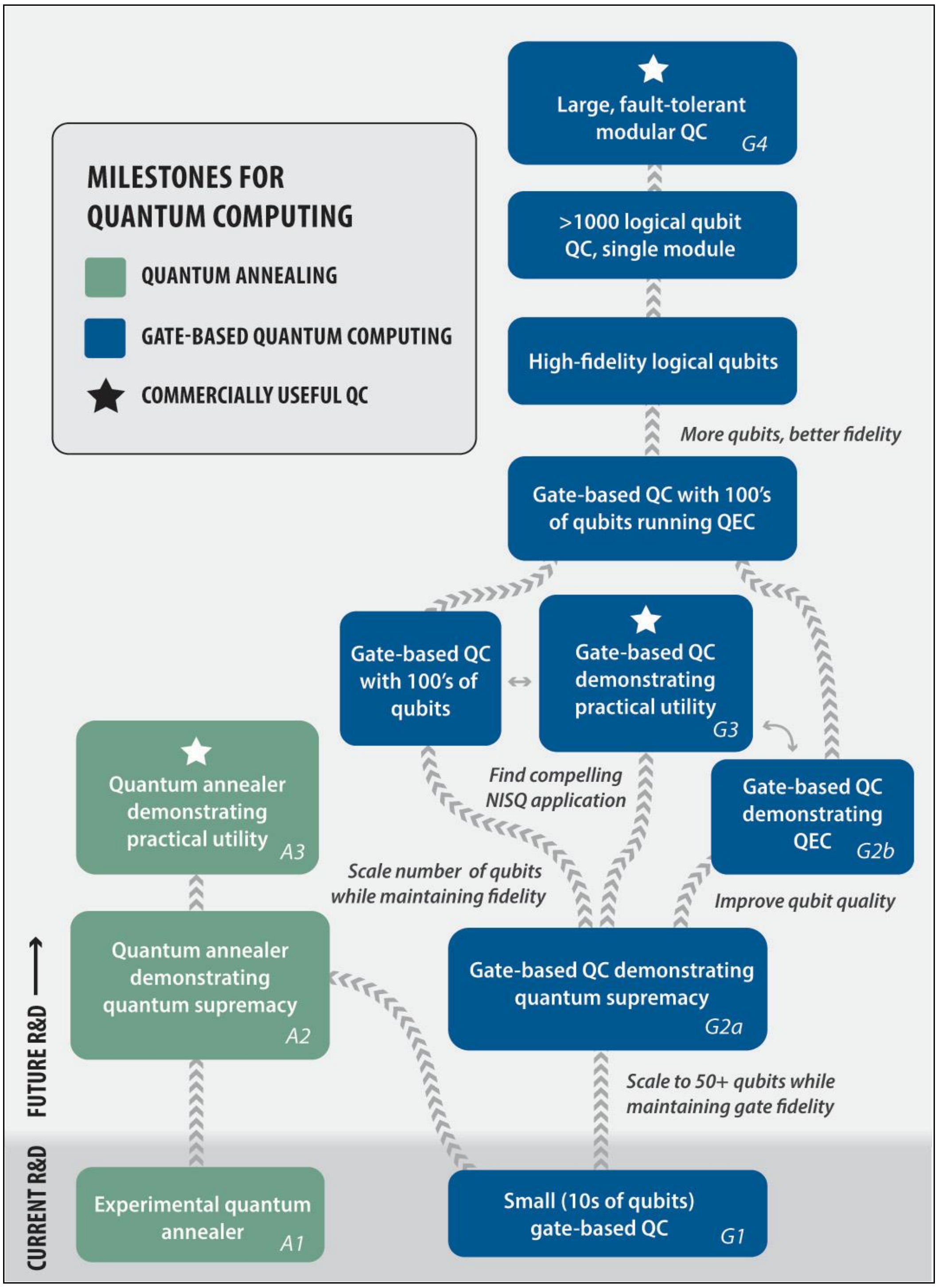

MILESTONES FOR QUANTUM COMPUTING
QUANTUM ANNEALING
GATE-BASED QUANTUM COMPUTING
COMMERCIALLY USEFUL QC
Large, fault-tolerant modular QC G4
>1000 logical qubit QC, single module
High-fidelity logical qubits
More qubits, better fidelity
Gate-based QC with 100's of qubits running QEC
Gate-based QC with 100's of qubits
Gate-based QC demonstrating practical utility G3
Gate-based QC demonstrating QEC G2b
Find compelling NISQ application
Scale number of qubits while maintaining fidelity
Improve qubit quality
Quantum annealer demonstrating practical utility A3
Quantum annealer demonstrating quantum supremacy A2
Gate-based QC demonstrating quantum supremacy G2a
Scale to 50+ qubits while maintaining gate fidelity
Experimental quantum annealer A1
Small (10s of qubits) gate-based QC G1
FUTURE R&D
CURRENT R&D

Figure 26. An illustration of potential milestones of progress in quantum computing. The arrangement of milestones corresponds to the order in which the committee thinks they are likely to be achieved; however, it is possible that some will not be achieved, or that they will not be achieved in the order indicated.

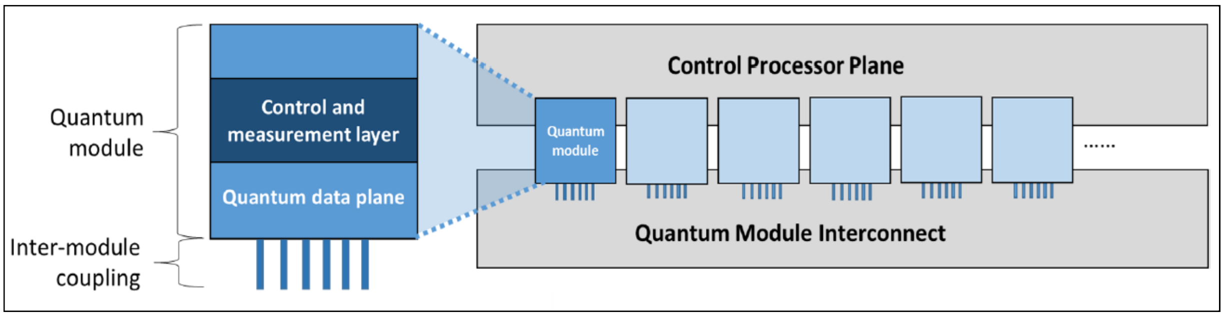


Figure 27. Schematic of a modular design approach to a large-scale, fault-tolerant quantum computer.

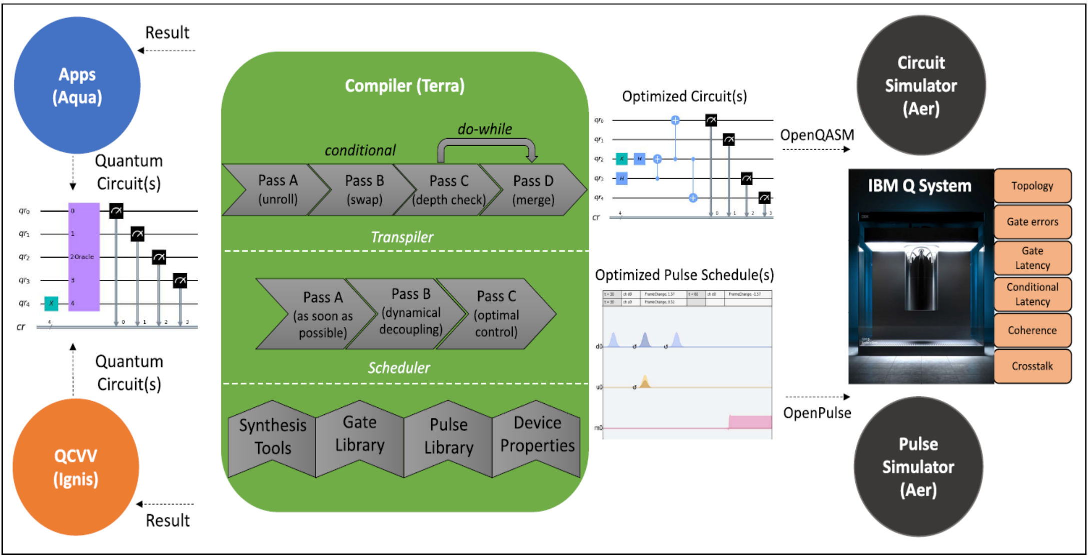


Figure 28. Architecture of Qiskit. Aqua and Ignis produce circuits for different tasks (algorithms and applications, or device QCVV, respectively).

The IBM Q systems and Aer simulators are backends that execute quantum circuits or pulse schedules. The Terra compiler is the bridge that translates and optimizes for a given backend and comprises modular pass-based circuit optimizers (Transpiler) and pulse optimizers (Scheduler). Some example passes are shown. Efficient high-level synthesis methods, access to a library of precomputed gate and pulse equivalents, and information about device constraints and properties all increase compilation quality.

The diagram represents device abstractions and is not intended to imply any particular physical device layout, which will depend on the specific technology and implementation. Each quantum module consists of its own data plane and control and measurement layer and intersects with the control processor plane.

Quantum computing is not only a beautiful way of exploiting HW devices governed by quantum mechanics, but also a new approach for information processing which may (and is) interesting and useful. The ideas introduced by quantum computing, like the use of reversible operators, have applications even without the disposability of real quantum computer. In fact, in the rest of this report the new methodologies, which hybridize quantum computing and soft computing (referred to as methodologies) are introduced.

These new methodologies widen the range of applicability of soft computing techniques, keeping their actual advantages. Since we are interested in applying the quantum computation paradigm, our approach consists first in the analysis of the input-output relations of each block and then in the simulation of these relations. In particular, we present a new circuit implementation of quantum search algorithm with information criteria (minimum of Shannon entropy) for search termination process that is the background for optimisation of control processes. We focus our attention on *superposition*, *entanglement*, and *interference* quantum operators (there are the fundamental quantum operations of QSA) and propose a new HW accelerator structure for Grover's QSA.

The TU Delft superconducting quantum computing stack (QuantumInfinity) along with Octobox-2 is schematically represented in Fig. 29.

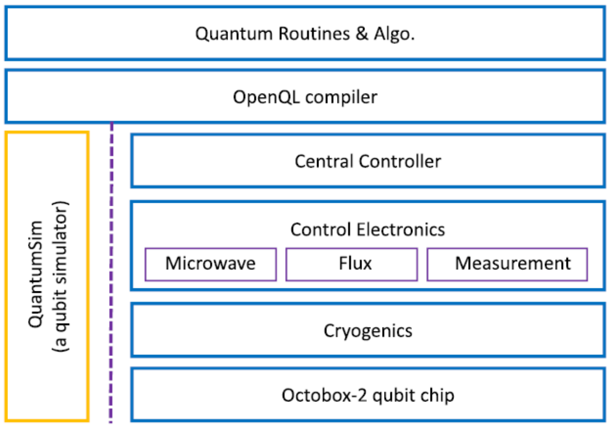
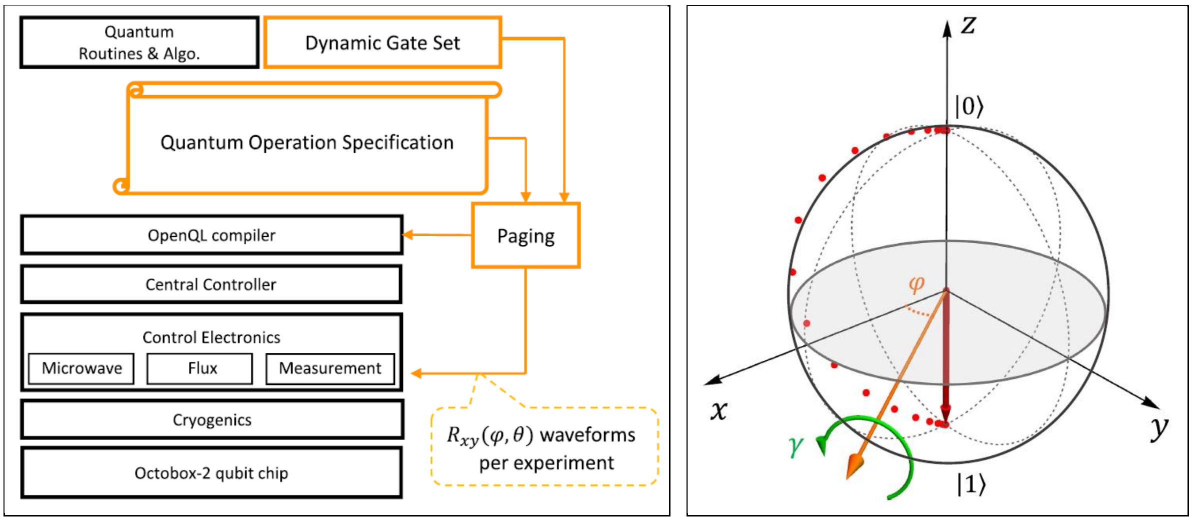


Figure 29. Simplified view of the bQI stack. The blue boxes label the main components with OpenQL compiler running on the host computer. QuantumInfinity also includes a density-matrix-based qubit simulator to simulate quantum experiments using parameters extracted from transmon qubit characterization.

QuantumInfinity translates the quantum instructions into physical MW pulses and dc flux pulses in order to execute the quantum algorithms on qubits. The baseline QuantumInfinity (bQI) system stack supported the following native instructions (or basis gate set) for qubit characterization and benchmarking. Arbitrary waveform generators (AWGs) stored the pulses corresponding to these single qubit and two-qubit operations. The bQI stack employed OpenQL as the compiler to generate executable code for the central controller (CC). The bQI system executed the single-qubit instructions similar to any programmable processing unit. The AWG stored a single pulse for each of the four instructions. Every instance of an instruction in a quantum program triggered the CC to command the AWG to drive the corresponding instruction's waveform via a codeword. This method is distinctly different from the conventional technique of qubit control, where all required qubit operations are concatenated into a monolithic waveform and played out once.
The techniques introduced in the eQI system to support arbitrary rotation instructions are shown in Fig. 30.



(a) (b)

Figure 30. (a) simplified viewb of the eQI stack. The quantum operation specification, a dynamic gate set, and paging are used to implement single-qubit arbitary-rotation instructions eQI; (b) Evolution of the qubit state during a R_{xy} ($\phi = 0.2\pi$, $\gamma = \pi$) operation is shown. The axis defined by $\varphi = 0.2\pi$ is represented by the orange arrow. The red arrow and the green circular arrows represent the final qubit state vector and the sense of rotation, respectively. The red dots on the Bloch sphere denote the most probable qubit state at each 1 ns timestep when the gate time is set to 20 ns.

R$xy(\phi, \gamma)$ is an instruction that effectively specifies an infinite number of rotation operations. This instruction applies a rotation of γ, around an axis oriented measured from the x-axis on the xy plane (see Fig. 30, b). This is challenging to implement, since the AWG has limited memory (to store waveforms) and limited codeword space (to address individual gates).We introduce and implement three mechanisms on the eQI stack to address these hardware constraints: *quantum operation specification* (QOS), *dynamic gate set* (DGS), and *paging* (PG).

Quantum algorithm presented unique architectural and design challenges in requiring a large number of single-qubit arbitrary rotations and two-qubit entangling gates. We implemented the two-qubit version of the algorithm with 60 different random disorder realizations, each of which contained 40 two-qubit instructions and 104 single-qubit instructions. The results validated the capability of our enhanced quantum computing system to run small-scale algorithms targeting real-world applications.
The results from state-of-the-art, few qubit devices are still limited in accuracy. The aim of this work is to implement necessary architectural features to maximize the potential of near-term systems. By utilizing larger quantum computers, we believe that there is potential to gain more accurate information about such materials systems, even with noisy qubits.

Figure 31 shows examples from quantum deep machine learning category.

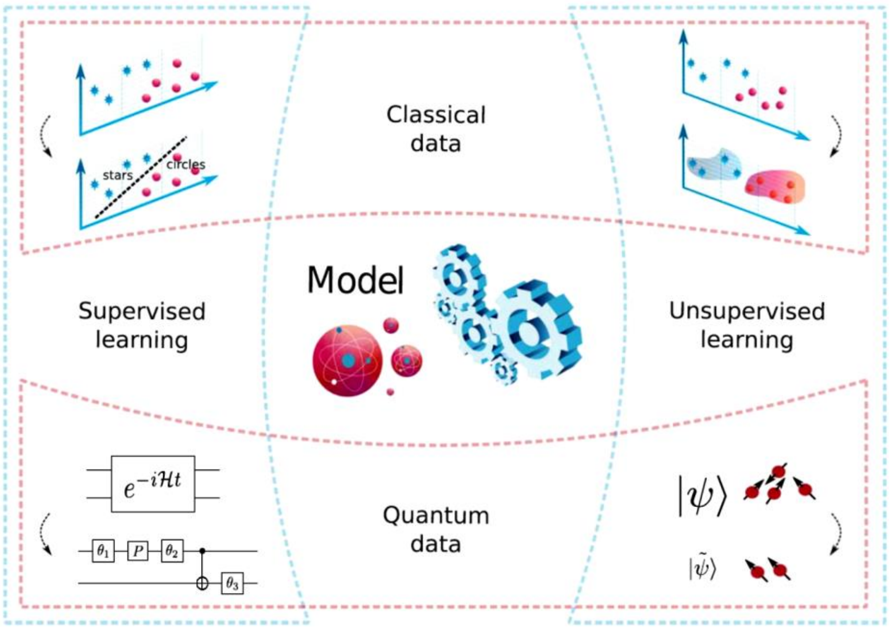


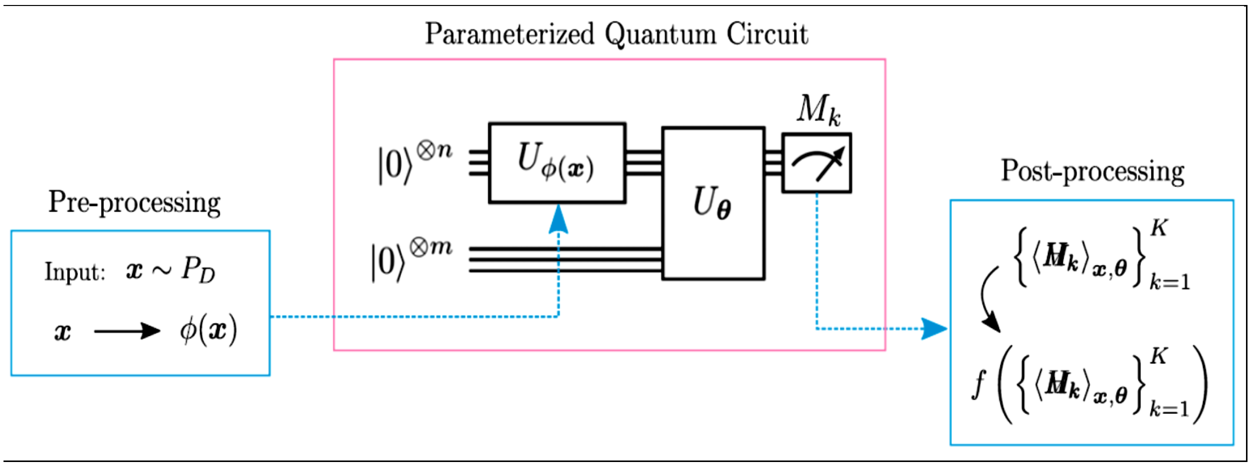
Figure 31. Parameterized quantum circuit models can be trained for a variety of machine learning tasks, such as supervised and unsupervised learning, on both classical and quantum data.

In the top-left panel, the model learns to recognize patterns to classify the classical data. In the top-right panel, the model learns the probability distribution of the training data and can generate new synthetic data accordingly. For supervised learning of quantum data, bottom-left panel, the model assists the compilation of a high-level algorithm to low-level gates.

Parameterized quantum circuits (PQCs) offer a concrete way to implement algorithms and demonstrate quantum supremacy in the NISQ era. PQCs are typically composed of fixed gates, e.g. controlled NOTs, and adjustable gates, e.g. qubit rotations. Even at low circuit depth, some classes of PQCs are capable of generating highly non-trivial outputs. For example, under well-believed complexity-theoretic assumptions, the class of PQCs called instantaneous quantum polynomial-time cannot be efficiently simulated by classical resources (see below for accessible Reviews of quantum supremacy proposals). The demonstration of quantum supremacy is an important milestone in the development of quantum computers. In practice, however, it is highly desirable to demonstrate a quantum advantage on applications.

Finally, for unsupervised learning of quantum data, bottom right panel, the model performs lossy compression of a quantum state. We look at machine learning applications using PQC-models where the goal is to obtain an advantage over classical models. For supervised learning with classical data we give a general overview of how PQC-model can be applied to classification and regression. For unsupervised learning with classical data we focus on generative modeling since this comprises most of the literature. PQC-models can also handle inputs and outputs that are inherently quantum mechanical, i.e. already in superposition. These are often referred to as quantum data. Quantum input data could originate remotely, for example, from other quantum computers transmitting over a quantum Internet. Otherwise, if a preparation recipe is available, one could prepare the input data locally using a suitable encoder circuit. Assuming this data preparation is efficient, one can extend supervised and unsupervised learning to quantum states and quantum information. Figure 32 shows examples for all these cases.

Intuitively each application is a specification of the components outlined in Fig. 32.


Parameterized Quantum Circuit
Pre-processing
Input: $\boldsymbol{x} \sim P_D$
$\boldsymbol{x} \longrightarrow \phi(\boldsymbol{x})$
$|0\rangle^{\otimes n}$
$|0\rangle^{\otimes m}$
$U_{\phi(\boldsymbol{x})}$
$U_{\boldsymbol{\theta}}$
M_k
Post-processing
$\left\{\langle \boldsymbol{M}_k \rangle_{\boldsymbol{x},\boldsymbol{\theta}}\right\}_{k=1}^{K}$
$f\left(\left\{\langle \boldsymbol{M}_k \rangle_{\boldsymbol{x},\boldsymbol{\theta}}\right\}_{k=1}^{K}\right)$

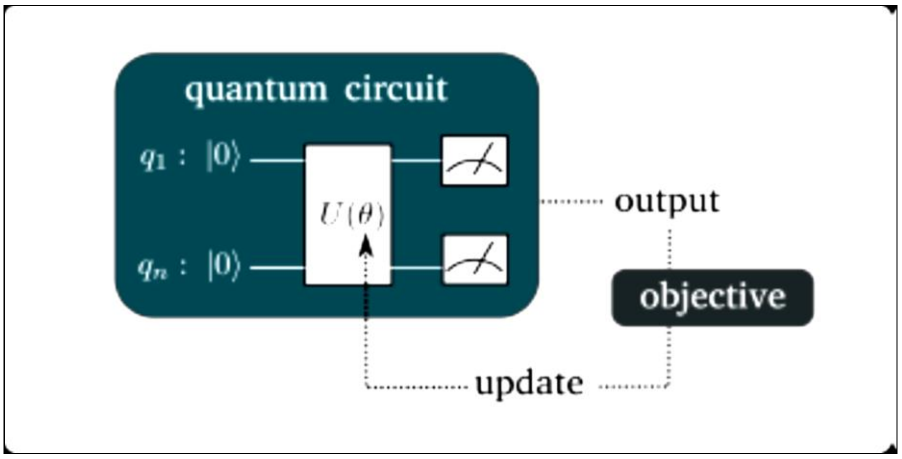


Figure 32. A machine learning model comprised of classical pre/post-processing and parameterized quantum circuit.

Variational circuits (also called *parametrized quantum circuits*) are a family of hybrid quantum-classical algorithms. At the core of the algorithm is a *quantum circuit* which depends on a set of circuit parameters, as well as an *objective function* which defines a scalar score for each set of parameters. The goal is to optimize or train the circuit with respect to the objective. Typically, variational circuits are trained by a classical optimization algorithm that makes queries to the quantum device. The optimization is usually an iterative scheme that finds better candidates for the parameters θ in every step, starting with either random or pre-trained initial parameters.

A data vector is sampled from the dataset distribution, $\boldsymbol{x} \sim P_D$. The pre-processing scheme maps it to the vector $\phi(x)$ that parameterizes the encoder circuit $U_{\phi(x)}$. A variational circuit U_θ, parameterized by a vector θ, acts on the state prepared by the encoder circuit and possibly on an additional register of ancilla qubits, producing the state $U_\theta U_{\phi(x)}|0\rangle$. A set of observable quantities $\left\{\langle M_k \rangle_{k,\theta}\right\}_{k-1}^{K}$ is estimated from the measurements. These estimates are then mapped to the output space through classical post-processing function *f*. For a supervised model, this output is the forecast associated to input ***x***. Generative models can be expressed in this framework with small adaptations.

Forward the quantum generative adversarial network (QGAN) and theoretically examine variants where target data, generator and discriminator are either classical or quantum introduced on Fig. 33.

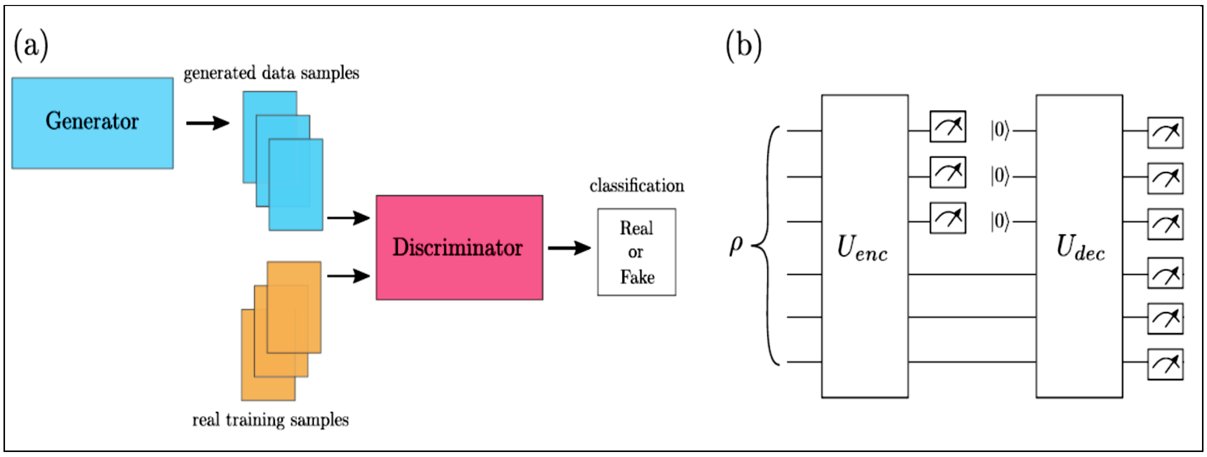


Figure 33. Illustration of quantum generative models. (a) In the quantum generative adversarial network, the generator creates synthetic samples and the discriminator tries to distinguish between the generated and the real samples. The network is trained until the generated samples are indistinguishable from the training samples. In this method the target data, the generator, and the discriminator can all be made quantum or classical. (b) The quantum autoencoder reduces the dimensionality of quantum data by applying an encoder circuit U_{enc}, tracing over a number of qubits and finally reconstructing the state with a decoder circuit U_{dec}. [7].

PQC models can also help in the study of quantum mechanical systems. For systems that exhibit quantum supremacy, a classical model cannot learn to reproduce the statistics unless it uses exponentially scaling resources. Provided that we can efficiently load or prepare quantum data in a qubit register, PQC models will deliver a clear advantage over classical methods for quantum learning tasks.

In many practical decision-making scenarios, there is no available data concerning the best course of action. In this case, the model needs to interact with its environment to obtain information and learn how to perform a task from its own experience. This is known as reinforcement learning. An example would be a video game character that learns a successful strategy by repeatedly playing the game, analyzing results, and improving. Although quantum generalizations and algorithms for reinforcement learning have been proposed, to the best of our knowledge, none of them are based on hybrid systems and PQC-models.

All the proposed HW architectures are modular in the sense that they can generalized by adding similar parts according to the desired number of qubits; furthermore, we do not use multipliers and, by utilizing logic gate in an analogical scheme, we reduce the number of operation and component, providing a substantial in speed-up of computation.

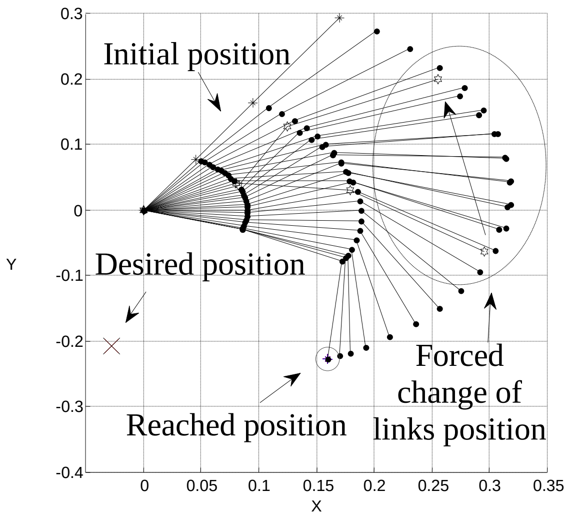
We are concentrating our attention on the description of matrix calculus for the efficient simulation and design methodology of quantum algorithm gates using classical computer technology. We introduced the topic of quantum algorithmic gate (QAG) design, explaining what its objectives are, and describing some of its physical resources, limitations and information bounds.

The system of effective simulation of quantum algorithms on classical computers described. Search quantum algorithms as Grover’s and Shor’s with SW / HW implementations are considered. Examples of speed up of SW and adapted acceleration of HW implementation of quantum computing discussed.

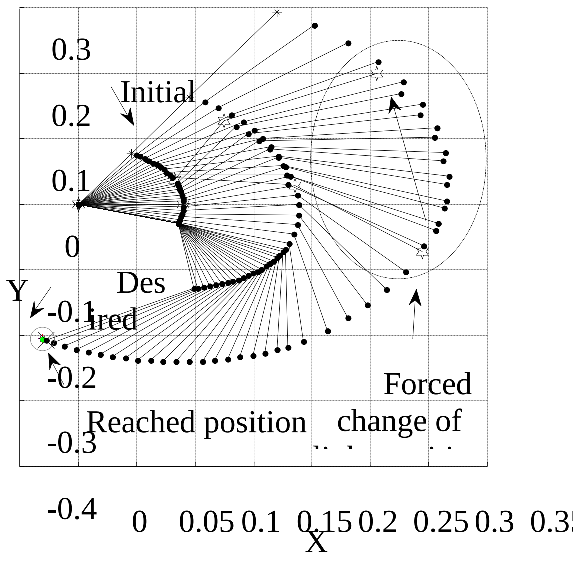
4. *Quantum supremacy of intelligent control*

Using two control objects of varying complexity (3DoF and 7DoF manipulators) as an example, the advantages and limitations of using soft and quantum computing technologies in intelligent control systems were demonstrated. On the example of 3DoF manipulator, the minimal difference between the results of the physical Manipulator Testbench and the MatLab/Simulink model demonstrated.

From Fig. 34, a, b it can see that in the considered unexpected control situation, intelligent control system based on quantum computing decides with the positioning problem with a given accuracy, in contrast to intelligent control system based on soft computing with separated control.



(a)



(b)

Figure 34(a) Intelligent control systems based on soft computing with separated control in unexpected control situation; (b) intelligent control systems based on quantum computing

Figure 35 demonstrate quantum supremacy in the considered unexpected control situation, intelligent control system based on quantum computing decides with the positioning problem of prosthetic robotic arm with a given accuracy, in contrast to intelligent control system based on soft computing with separated control (see Fig. 34 a).

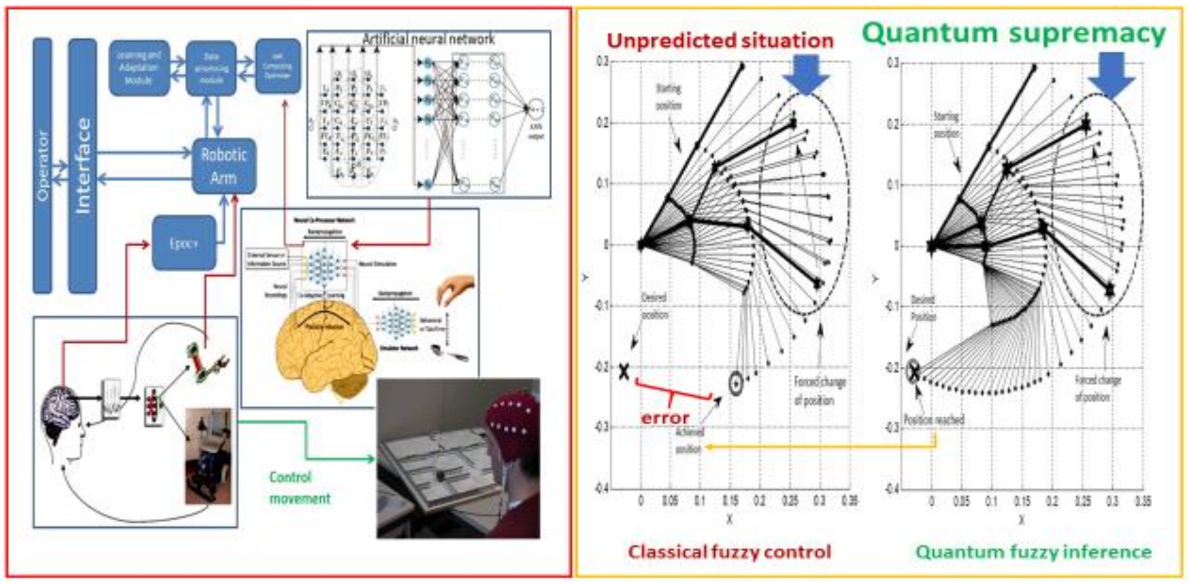


Figure 35. Example of quantum supremacy: Application of quantum computing cognitive control of prosthetic arm with super-accuracy of position in unpredicted situation.

Figure 36 shows a generalized comparison of intelligent control systems based on soft and quantum computing for 3DoF and 7DoF manipulators for standard and unexpected control situations of the considered examples and demonstrate quantum supremacy.

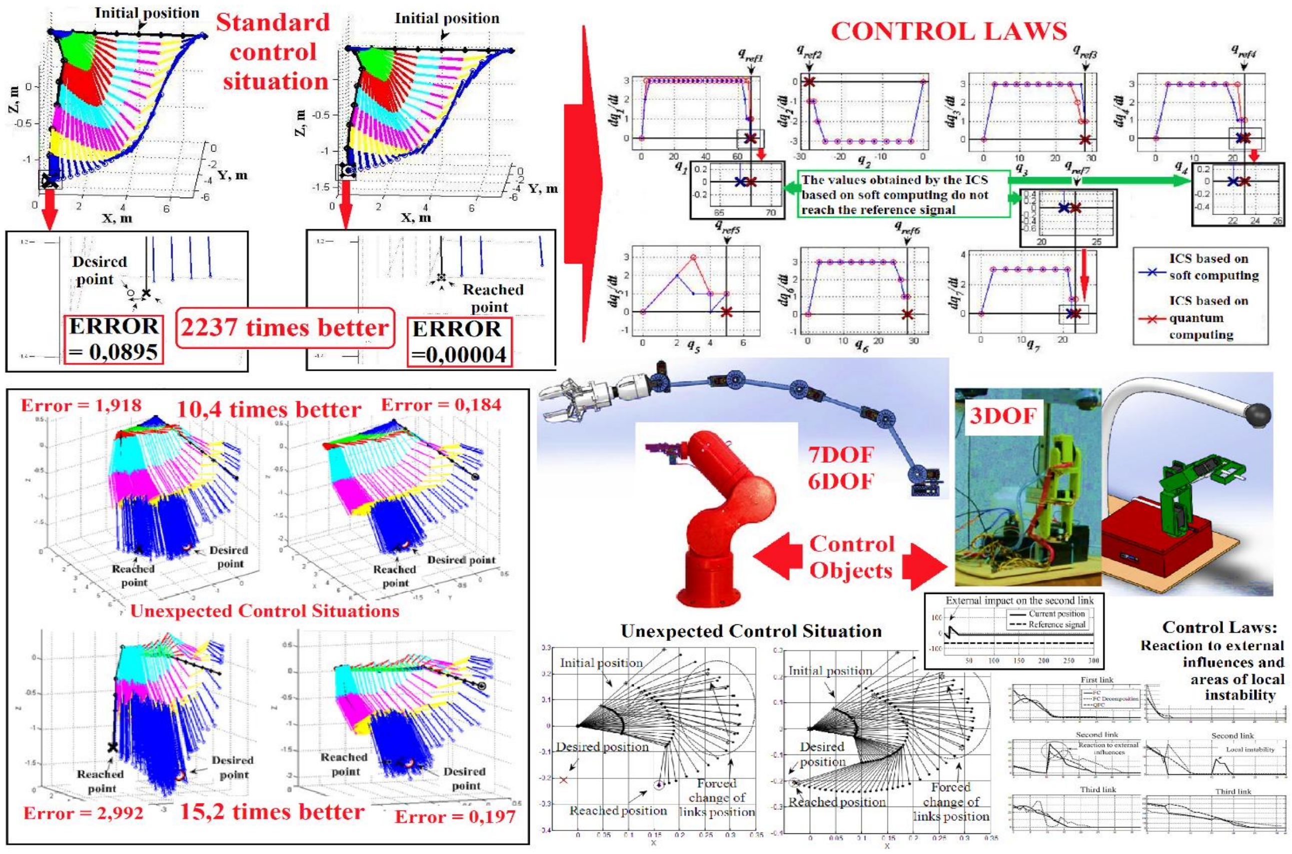


Figure 36. Comparison of intelligent control systems based on soft and quantum computing for 3DoF and 7DoF

Further research focused on the development and analysis of the physical Testbench of Manipulator with 7 DoF, as well as its integration with the mobile platform.

Our main applied domains of QAG - approach: (i) Computer science; (ii) Artificial intelligence (AI) industrial applications; (iii) Fundamental intelligent informatics and general system theory; and (iv) Intelligent robust control system design in next two books are considered.

These domains be in deep interdependence. And included quantum algorithms design (quantum walk models, one-way computing and quantum adiabatic computing) in many applied quantum software engineering: quantum machine learning; quantum game models and decision-making processes in information uncertainty; quantum pattern / face recognition; quantum error correction codes; quantum-inspired neural network structures design (for quantum learning) and quantum-inspired genetic search algorithms (for quantum global optimization) etc.

It is the background of quantum soft computing for robust KB design of intelligent control system and robust quantum control (quantum soft computing applications), etc.

Conclusion

We described briefly any important applications of QAG simulation system (as examples, quantum games, quantum random search walk algorithms and control decision-making processes in classical and quantum situations of information uncertainty) and its developed tools in AI-systems design. Main ideas and peculiarities of Quantum Soft Computing tools as a new paradigm of computational intelligence and simulation processes briefly are considered. Applied Quantum Soft Computing toolkit (as a quantum computational toolkit and background for robust intelligent control design technology) discussed.

References

1. Horowitz M., Labonte F., Olukotun K. and C. Batten. New plot and data collected for 2010-2015 [by K. Rupp].
2. Pino J. M., Demonstration of the QCCD trapped-ion quantum computer architecture // arXiv:2003.01293v1 [quant-ph] 3 Mar 2020. Taken by Erik Lucero (b) IBM five-qubit universal quantum computer http://web.physics.ucsb.edu/~martinisgroup/photos.shtml, (released May 2016).
3. Salm M. A Roadmap for Automating the Selection of Quantum Computers for Quantum Algorithms // arXiv:2003.13409v1 [quant-ph] 30 Mar 2020.
4. Wilhelm F.K. et al. Entwicklungsstand Quantencomputer. — Federal Office for Information Security. — 2017.
5. Benedetti M. Parameterized quantum circuits as machine learning models // Quantum Sci. Technol. — 2019. — Vol. 4. — Pp. 043001.
6. Leymann F., Barzen J., Falkenthal M. Quantum in the Cloud: Application Potentials and Research Opportunities // arXiv: 2003.06256[quant-ph]. 2020.
7. Josephson Junction Quantum Computing at UCSB http://web.physics.ucsb.edu/~martinisgroup/photos.shtml, taken by Erik Lucero (b) IBM five-qubit universal quantum computer (released May 2016).
8. Debnath S., Linke N. M., Figgatt C., Landsman K. A., Wright K. and C. Monroe. Demonstration of a programmable quantum computer module. // arXiv: 1603.04512 v1 [quant-ph], 2016.
9. Quantum Image Processing: the truth, the whole truth, and nothing but the truth about its problems on internal image representation and outcomes recovering // arXiv: 2002.04394 v1 [quant-ph], 2020.
10. Benedetti M. Parameterized quantum circuits as machine learning models // Quantum Sci. Technol. — 2019. — Vol. 4. — Pp. 043001.
11. ZOU X. Enhancing a Near-Term Quantum Accelerator’s Instruction Set Architecture for Materials Science Applications // IEEE Trans. On Quantum Engineering. — 2020. — Vol. 1. — Pp. 4500307 [DOI 10.1109/TQE.2020.2965810].