Quantum Computing. II.Quantum Computer Languages

Ivan Bolesta
dept. of Electronics and Computer Technologies
Ivan Franko National university of Lviv
Lviv, Ukraine
bolesta@electronics.lnu.edu.ua

Oleksii Kushnir
dept. of Electronics and Computer Technologies
Ivan Franko National university of Lviv
Lviv, Ukraine
alex.kuschnir@gmail.com

Serhiy Velhosh
dept. of Electronics and Computer Technologies
Ivan Franko National university of Lviv
Lviv, Ukraine
velgosh@electronics.lnu.edu.ua

Yuriy Furgala
dept. of optoelectronics and Information Technologies
Ivan Franko National university of Lviv
Lviv, Ukraine
yuriy.furhala@lnu.edu.ua

Abstract—**The programming languages that can be used to write the program code when using quantum computers are reviewed. The prospects of using such languages from the point of view of integrating them into software complexes that are convenient for the developer are discussed.**

Index Terms—**qubits, quantum logic elements, quantum circuits, quantum languages, quantum computer.**

I. INTRODUCTION

The theoretical foundations and basic principles of quantum computers have been formulated long ago [1]. Therefore, it is not surprising that the development of proper programming languages began to take place at the beginning of the computer era. Therefore, despite the lack of real-world devices that carry out quantum computing, programming languages for quantum computers and emulators for their use have been developed for 30 years. Therefore, during this time many defects in this area have been created that have different properties and principles of use. Due to the peculiarities of the functioning of modern quantum computers, it is impossible for them to directly use conventional classical programming languages. However, the very quantum computer is not enough for common tasks that are now solved easily on ordinary computer systems. Therefore, at this stage of development, it is expedient to use quantum computers as a co-processor to a large data center. This will allow it to be used in a cloud service only for specific specific tasks.

II. QUANTUM COMPUTER LANGUAGES

The programming languages for development under quantum computers should be with a classic syntax, understandable to the modern developer, but with built-in capabilities of setting the initial values of qubits and using a set of gates.

Another important parameter of quantum programming languages is the existence of a full-fledged quantum computer emulator for this language. Testing the code with the help of an emulator allows you to run a real computer already working code, and debug it on the emulator. In addition, emulators allow you to check the states of qubits at any time, which is impossible on a real device. However, of course, the emulator does not have an acceleration of quantum parallelism.

Despite the fact that there are not so many working quantum computers and access to them is limited (with the exception of the IBM Q Experience [2] program, there are already quite a large number of quantum programming languages. There are two groups of quantum programming languages: imperative quantum programming languages and functional quantum programming languages. The paradigm of imperative programming, in which the process of obtaining the results is described as a sequence of instructions for changing the state of the program, is close to the physical work of the quantum computer, in which we can set the initial values of qubits, act on the sequence of gates and obtain the result. That is so-called “pipe-filter” architecture. Therefore, the quantum code can be exclusively consistent, without branching and cycles. The most famous representatives of this group are QCL [3], LanQ [4], and OpenQASM. On the other hand, the paradigm of functional programming is more familiar to modern developers and more user-friendly. This group includes LIQUi, Q#, Quipper, and Python QISKit.

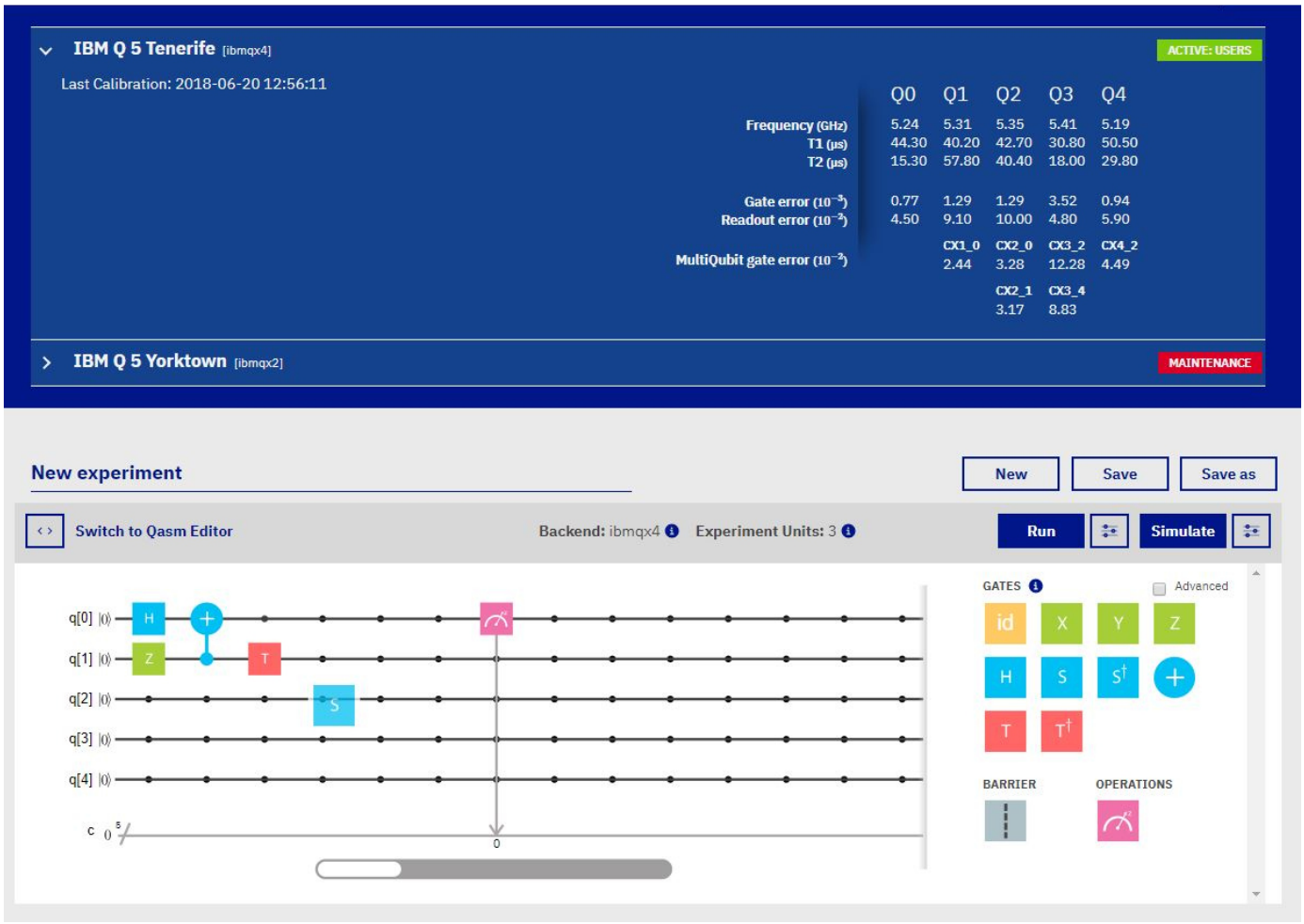

Fig. 1. A main view of the editor of the quantum scheme of IBM Q Experience project.

Quantum Computation Language (QCL) was created by Bernhard Omer in 1998, although language development continues to this day. Language has an emulator that allows you to run quantum programs on a classic computer.

LanQ is a quantum computer science research project. This quantum programming language designed to support the parallel execution of several processes. The source code is distributed under an open GNU GPL license. This program uses a Java machine and a required Java version, at least 1.5.0. All required libraries are distributed in the lanq.jar file. The syntax of the LanQ language is almost entirely taken from C language.

Open Quantum Assembly Language (OpenQASM). The OpenQASM source code was released as part of the IBM Quantum Information Software Kit (QISKit) software for use with quantum computing platform Quantum Experience. Therefore it has not only a quantum emulator (up to 32 qubits) but also the ability to launch the real-world platform of the IBM Q. OpenQASM has common features with specialized programming languages (such as Verilog) used to describe the structure and behavior of electronic circuits.

LIQUi (Language-Integrated Quantum Operations) The LIQUi platform was created by Quantum Architectures and Computation Group in the Microsoft Research project and includes a programming language, optimization algorithms, and several quantum simulators. LIQUi can be used to convert a quantum algorithm written in the form of a high-level program in the language of F# from the .NET Framework family in low-level commands for a quantum computer.

LIQUi allows you to simulate up to 30 qubits per station with 32 GB of RAM. The platform can be used to define, execute and display in different graphic formats of quantum circuits. With LIQUi you can simulate simple quantum teleportation, the Schore factorization algorithm, quantum associative memory, quantum linear algebra.

Quipper. This language was created by a group of authors led by Peter Selinger. Quipper is designed for the same programming tasks as the QCL, but has a different structure and appearance. Language is implemented as an extension of Haskell, which uses a functional rather than imperative way of expression.

Q# is a subject-oriented programming language that is used to express quantum algorithms. It was first released by Microsoft as part of the Quantum Development Kit.

During the Ignite conference on September 26, 2017, Microsoft announced plans to release a new programming language specialized in quantum computers. On December 11, 2017, Microsoft released Q# as part of the Quantum Development Kit.

Q# is only available as a standalone download to Visual Studio. The Quantum Development Kit comes with a quantum simulator that can execute programs written in Q#. To activate the same quantum simulator, you must use the "shell" program in any other programming language of the .NET family.

In Q#, qubits are executed in the form of topological qubits. The Quantum Development Kit quantum simulator is capable of generating up to 32 qubits on local computer or 40 qubits on Azure cloud.

III. Graphic Quantum Design

Another approach to developing programs for quantum computers is graphic design, writing of ready-made schemes in a special editor (fig. 1).

The editor runs in the browser window and also has the ability to edit the text of programing code in the language of OpenQASM.

On the right side of the schema creation area is a set of possible gates (fig. 1). The result of the program can be obtained from both the simulator and the real quantum computer.

The limitation for an ordinary user is the ability to work with only five qubits.

Thus, a mathematician or a physicist who understands the principles of the work of quantum computers (subject area), but can't write a program code independently, can easily develop a project of quantum computing in such an editor. This significantly reduces the required input level for the developer.

IV. Scheme of Quantum Computer use

Quantum computers have greater efficiency in using for some tasks, but for ordinary daily tasks, their use is not feasible. An efficient scheme is the use of a quantum compiler as a cloud service for specific tasks of decryption and simulation. Then there is the following scheme of use (Fig. 2).

Then, any user of the service will be able to launch a quantum code from user's device through the cloud. Such a cloud should consist of a classical computer that manages and accepts quantum data. This allows you to enjoy all the benefits of quantum acceleration in the existing infrastructure.

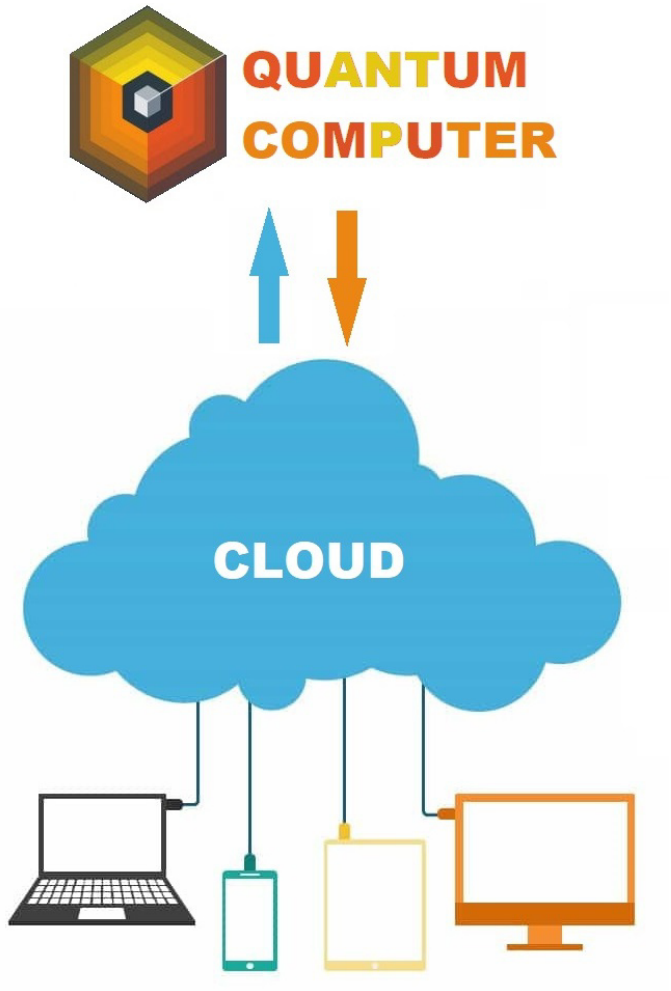

Fig. 2. Scheme of Quantum Computer use.

V. Conclusions

Despite the recent appearance of real quantum computers, there is already a large number of programming languages for them. Some languages come in large complexes, which also have emulators that can help future developers.

The current development of technology dictated the scheme of using a quantum computer as an application of cloud-based service. The first such system already exists and runs in test mode.

REFERENSES

[1] A. Einstein, B. Podolsky, N. Rosen "Can quantum-mechanical description of physical reality be considered complete?" Phys. Rev. V. 47, 1935, pp. 777–780.

[2] "IBM Makes Quantum Computing Available on IBM Cloud to Accelerate Innovation" https://www-03.ibm.com/press/us/en/pressrelease/49661.wss

[3] B. Omer "QCL - A Programming Language for Quantum Computers," http://tph.tuwien.ac.at/~oemer/qcl.html

[4] H. Mlnařík. "LanQ – a quantum imperative programming language," http://lanq.sourceforge.net/