Issue Date:	**2020-12-01**	**CEN-CLC/FGQT N050a**
Deadline:	**n/a**	**Supersedes: N050**
Status:	FOR APPROVAL	

TITLE	**Use case: Using a Quantum Computer as Secondary Processor in the Qloud**
PROJECT	FGQT Roadmap
REFERRING TO	N020a, N050
SOURCE	Niels Neumann (TNO)
CONTACT	niels.neumann@tno.nl

Supported by:

- Rob van den Brink (Delft Circuits, The Netherlands),
  - Rob.vandenBrink@Delft-Circuits.com,
  - https://delft-circuits.com/
- Giovanni Frattini and Antonello Corsi (Engineering Ingegneria Informatica S.p.A., Italy),
  - Giovanni.Frattini@eng.it and Antonello.Corsi@eng.it
  - http://www.eng.it

ABSTRACT

Explanation of changes

We propose this contribution on use-cases for quantum computers. This contribution focusses mainly on the technicalities desired when using quantum computers. We propose to add this contribution to the roadmap document *Standardization Roadmap for Quantum Technologies*, specifically to section 5: *Innovation and use cases*.

Why are the changes needed?

This document provides a use case for using cloud-based quantum computing. This in turn helps in identifying where (additional) standards are desired. The sketched use-case is focused already on the near-term, where specific problems can be solved more efficiently by using quantum computers.

Instructions for editor

Section 4 to be included as-is in section 5.1.2 *Domain-specific use cases* of N020a.

1. Situation sketch

Quantum computing has the potential to solve some computational problems efficiently or provide meaningful gains in other problems. Examples of the former are cryptographic related problems, such as prime factorization and the discrete-log problem, and simulations of chemical processes. An example of the latter is machine learning, as quantum computing can provide meaningful gains in the running time, the number of training steps required or in the capacity of associative networks.

It is expected that, at least for the foreseeable future, quantum computers will mainly be hosted on few locations worldwide, which can be accessed remotely. Currently, this means that you have to log on to some programming environment of the host and program your quantum algorithm there. Another option is that you program the quantum instructions locally and then send everything to the host. Ideally however, users are not bothers with technicalities of implementations and low-level quantum instructions, but instead only with the result of a quantum routine.

An example of a high-level quantum computing software-stack is shown in Figure 1. Currently, the separation between a local user and a remote quantum host is high in the stack. Already on a high level, instructions must be programmed or shared with the quantum computer host. Therefore, users are required to program their algorithms in low-level instructions.

The desired separation is however lower in the stack, such that users can program quantum instructions on a high-level and function libraries can be used when programming the quantum or hybrid algorithm.

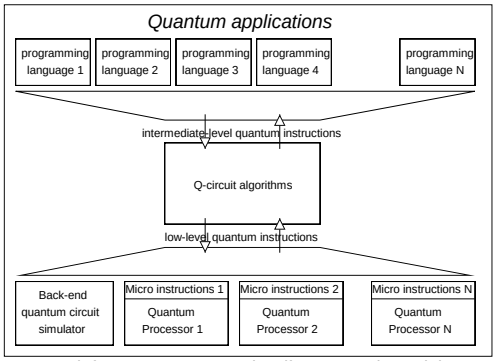

Figure 1: A software stack for quantum computing (from: Van den Brink, Neumann, Phillipson "Vision on Next Level Quantum Software Tooling", 2019).

2. Use case of quantum computing

Suppose a ~~company~~ user has developed an algorithm focussed on pattern-recognition ~~has developed an algorithm has developed a pattern-recognition algorithm~~ with a specific computationally heavy sub-routine. A hybrid algorithm might provide the solution, however this requires the algorithm to be partly run on a classical (super)computer and partly on a quantum computer. As quantum computers are likely to be hosted in the cloud and quantum computing time can be bought.

In this case, there are two points of attention:

1. The ~~company~~ user will likely not want to share their algorithm;
2. The ~~company~~ user is only interested in the result of the sub-routine.

Currently, the user has to program their algorithm in low-level quantum instruction and provide those to the host, or program the algorithm in an environment provided by the quantum host. Instead, the user should be able to program as high-level as possible. Therefore, a quantum library, with high-level quantum instructions is essential. Instructions in this library are then compiled to lower-level instructions. The library can be hosted at the user-side or at the host-side, and similarly for the compilation of the instructions. This quantum library should be callable from most to all often used classical programming languages. The user should be able to opt for running the quantum library locally if the algorithm is confidential or the user is not willing to share the algorithm for other reasons. Both the low-level instructions and those supported by the quantum library should at least support a standard instruction set. Note that these standardized low-level instructions are hardware-agnostic. In a compilation step, these instructions can be translated to hardware-specific operations, based on the used hardware-backend.

Furthermore, the compiled quantum instructions are not of interest to the user, instead, only the end-result is. Therefore, also the used backend quantum-technology is in general not of interest and could be decided upon automatically, based on requirements imposed by the algorithm. Finally, error-correcting methods should be applied if needed, without users having to worry about them. Note that it should be possible for users to indicate a preference for specific backends, for instances as user agreements may differ between different quantum computer hosts.

As multiple users may want to use the quantum computer simultaneously, there is a need for protocols, billing and job-control to facilitate who runs first and for how long. For a specific run, the user should be able to interact with the cloud-based quantum computer on what instructions to execute, but also when the calculation is finished and what the results are.

Additionally, the classical part of the algorithm is likely to be run on a high performance computer (supercomputer). Therefore, the supercomputer must be able to interact with the quantum computer, sending instructions and retrieving results.

There are two important notes on the above:

1. The standardized low-level gate set should be universal and hardware-agnostic. By high-level compilation to this standardized low-level gate set, it can in a second step be mapped to arbitrary hardware-backends. This mapping constitutes translating the low-level gates to hardware-specific operations, based on the used hardware-backend.

Current quantum-hardware options require users to access the devices from a remote environment. The term cloud in this document relates to how quantum computers should be used in the future and has similarities with the classical cloud.

quantum computing

4.3. Requirements for quantum computing

The example presented above sketches various requirements for quantum computing to be practically used:

- A local hybrid programming environment where classical and quantum instructions can be used;
- Support for high-level quantum instructions;

- Compilation of these high-level quantum instructions to a set of standardized (hardware-agnostic) low-level instructions (e.g. QASM) and some specific higher level instructions (e.g. quantum Fourier transform);
- Both local and remote support for compilation. The user may opt to compile locally, or may have no preference;
- In case of (hardware-agnostic) low-level instructions send by the user, there should be the option for integration of hardware specific constraints;
- In case of high-level instructions send by the user, there should be messaging to facilitate debugging;
- Protocols on how to interact with the quantum computer, send instructions and retrieve results;
- Protocols on billing and job control to determine who runs first and for how long;
- Compilation of quantum instructions to (hardware-agnostic) low-level instructions such that reconstruction of the original algorithm is hard;
- Translation of (hardware-agnostic) low-level instructions to hardware-specific elementary operations;
- Automatically apply error-correcting techniques (the necessity of these may depend on the specific backend);
- An interface between the quantum computer and supercomputers, to send instructions and provide (intermediate) results.

5.4. To be approved text

The following text is to be approved as-is for section 5.1.2: *Domain-specific use cases*.

Use Case: Using a Quantum Computer as Secondary Processor in the Coud

A user has developed a pattern-recognition algorithm with a specific computationally heavy sub-routine. To optimize both the running time and the results, the user wants to run part of its algorithm on a quantum computer without revealing the propriety algorithm. The rest of the algorithm is programmed in an often-used classical programming language and it is run on a classical (super)computer. Only the result of the quantum-subroutine is needed in the rest of the algorithm.

This requires a quantum-host and infrastructure, such that the user can run the quantum-subroutine under the requirements that

- The user can program the algorithm locally in a classical environment using a high-level classical programming language and additional quantum instructions;
- The quantum instructions can be programmed both on a high- and a low-level;
- The high-level quantum instructions are compiled to hardware-agnostic low-level instructions, either locally or at the host. The company should be able to decide on this;
- The low-level quantum instructions should be from a hardware-agnostic standardized gate set;
- If the user decides to compile locally, low-level instructions are send to the cloud-based quantum computer. The user should then also have the option to integrate hardware specific constraints. Additionally, this should be taken care of by the host if needed;
- The hardware-agnostic low-level quantum instructions should then be translated to hardware-specific instructions, corresponding to the backend;
- If the user instead sends high-level instructions to the host, and the instructions are compiled on the host-side, there should be messaging that facilitates debugging. Additionally, the user should be able to integrate specific hardware-constraints. Otherwise, the compiler should take care of this;

- The high-level quantum instructions may in this case be directly compiled to hardware-specific instructions;
- It is hard to reconstruct the original algorithm from the compiled instructions;
- There is a standardized interface allowing for executing (bursts of) quantum instructions, and the algorithm should automatically call the quantum instructions when needed;
- Protocols exist such that the user can send their quantum instructions to the hardware whenever needed, instructions are processed and run and results are send back to the user;
- Protocols exist on billing and job control to determine who runs first and for how long among different users;
- Quantum error-correction routines are applied if needed, the user should have influence on this if desired;
- An interface exists between the quantum computer and (super)computers, potentially hosted in the cloud, to send instructions and provide (intermediate) results;
- The result of the quantum-subroutine can directly be used by the (super)computer running the whole algorithm.