Quantum Computer Architecture: Towards Full-Stack Quantum Accelerators

K. Bertels, A. Sarkar, T. Hubregtsen, M. Serrao,
A.A. Mouedenne, A. Yadav, A. Krol, I. Ashraf
Quantum Computer Architecture lab
Delft University of Technology, Netherlands

Abstract—**This paper presents the definition and implementation of a quantum computer architecture to enable creating a new computational device - a quantum computer as an accelerator. A key question addressed is what such a quantum computer is and how it relates to the classical processor that controls the entire execution process. In this paper, we present explicitly the idea of a quantum accelerator which contains the full stack of the layers of an accelerator. Such a stack starts at the highest level describing the target application of the accelerator. The next layer abstracts the quantum logic outlining the algorithm that is to be executed on the quantum accelerator. In our case, the logic is expressed in the universal quantum-classical hybrid computation language developed in the group, called OpenQL, which visualised the quantum processor as a computational accelerator. The OpenQL compiler translates the program to a common assembly language, called cQASM, which can be executed on a quantum simulator. The cQASM represents the instruction set that can be executed by the micro-architecture implemented in the quantum accelerator. We propose that the industrial and societal application developers use perfect qubits that have no decoherence or error-rates. The perfect qubits offers facilities to the quantum application developer and they are not blocked by issues such as decoherence.**

I. Introduction

The history of computer architecture dates back various decades and has been very evolving. An important extension is the emergence of accelerators [1] as specialised processing units to which the host processor offloads suitable computational tasks. Recently, computer architecture research is getting more focused on quantum computing. In the next 5 to 10 years of quantum computer development, it does not makes sense to talk about quantum computing in the sense of a universal, Turing computer that can be applied in any kind of application domain. Given the recent insights leading to e.g. Noisy Intermediate-Scale Quantum (NISQ) technology as expressed in [2], we are much more inclined to believe that the first industry-based and societal relevant application will be a hybrid combination of a classical computer and a quantum accelerator. It is based on the idea that any end-application contains multiple computational kernels and the properties of these parts are better executed by a particular accelerator which can be, as shown in Figure 1, either field-programmable gate arrays (FPGA), graphics-processing units (GPU), neural processing units (NPU) like Google's tensor processing unit, etc. The formal definition of an accelerator is indeed a co-processor linked to the central processor that is capable of accelerating the execution of specific computational intensive kernels, as to speed up the overall execution according to Amdahl's law. We now add two classes of quantum accelerator as additional co-processors. The first one is based on quantum gates and the second is based on quantum annealing. The classical host processor keeps the control over the total system and delegates the execution of certain parts to the available accelerators.

Computer architectures have evolved quite dramatically over the last couple of decades. The first computers that were built did not have a clear separation between compute logic and memory. It was only with von Neumann's idea to separate and develop these distinctly that the famous von Neumann architecture was born. This architecture had for a long time a single processor and was driven forward by the ever increasing number of transistors on the chip, which doubled every 18 months. In the beginning of the 21st century, the single cores became too complex and did not provide any substantial processing improvement. This led to the incorporation of multiple cores. The homogeneous multi-core processor dominated the processor development for a couple of years but companies such as IBM and Intel started understanding that heterogeneity is the right way forward to improve the compute power. GPUs and FPGAs are seen as natural extensions of the computer architecture, implying that the quantum accelerator would be a logical next step.

In the quantum computing world, there exist two important challenges. The **first** is to have enough numbers of good quality qubits in the experimental quantum processor. The current competiting qubit technologies include ion traps, majoranas, semi-conducting and superconducting qubits, NV-centers and even graphene.

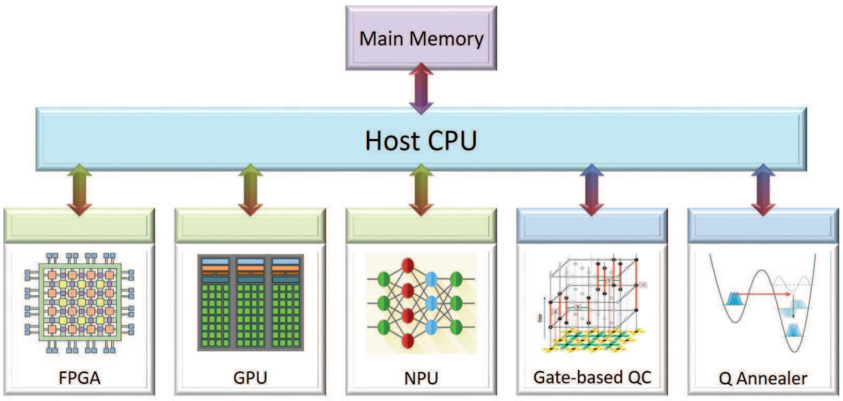


Fig. 1: System architecture with heterogeneous accelerators

Improving the overall status of the qubits is challenging as these suffer from decoherence that introduces errors when performing quantum gate operation. It is only when the quantum physical community overcomes those challenges that the quantum accelerator will be a widespread adopted solution. This direction is shown in the left picture of Figure **??** where different quantum technologies are depicted in the lowest layer. The **second** challenge is to formulate at a high level the quantum logic that companies and other organisations need to be able to use high-performance accelerators for certain computations that can only run on the quantum device. This requires a long-term investment in terms of people and technical know-how from companies that want to pursue this direction and reap the benefits. The right part of Figure **??** shows the industrial commitment to think about the required quantum logic that can be executed using the full-stack, evaluated and tested on a quantum simulator. It is important to emphasise that the qubits are called perfect qubits that do not decohere or have any other kind of errors generated by them. With the emergence of huge amounts of data, commonly called big data, it is understood that this paradigm is not scalable to super-large data sets. The key factor is the huge amount of data that needs to be processed by multiple computing cores which is a very tough problem to solve. The data communication between the cores is a very difficult programming problem and the data management problem is substantially slowing down the overall performance.

As shown in Figure **??**, an important concept that we have been implementing in the quantum computing world is the implementation of a full stack for a quantum accelerator as will be described later in this paper. The basic philosophy of any accelerator is that a full stack needs to be defined and implemented. The last 10 to 15 years have shown a large number of accelerators that were developed as part of any modern computer architecture. It always consists of the same following layers: it starts at the highest level describing the logic that needs to be mapped on the accelerator. Examples are video processing, security, matrix computation, etc. These application-specific algorithms can be defined in various languages such as C++ or Fortran. In the case of FPGAs, these algorithms are translated into VHDL or Verilog. In the case of GPUs, the language is often formulated using mathematics or other libraries and translated by the compiler to an assembly language that can be mapped on the GPU-architecture. Especially in the case of FPGAs, there is no standard micro-architecture on which the VHDL or Verilog can be executed. Such an architecture needs to be developed for every application that needs to be accelerated. The final layer is a chip based implementation of the micro-architecture combined with the hardware accelerator blocks that are needed.

II. The Quantum Full-Stack

In the context of quantum accelerator development, the same full-stack approach is adopted for either perfect or realistic qubits. The execution can be either on an experimental quantum chip or on the QX simulator. The highest level starts at the end-user application for which a part of that application is developed in a quantum language, such as OpenQL. The quantum part of any industrial or societal application can be executed on any kind of available quantum prototype. For any quantum logic that is specified, a specific and target-related micro-architecture needs to be defined and used. We present the considerations for the various layers in this section. Besides gate-based quantum computing approach, we

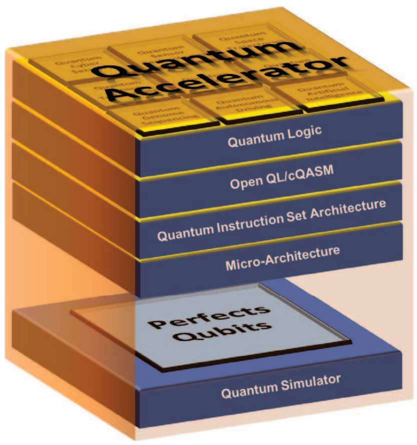


Fig. 2: Simulated full-stack with perfect qubits

also include the quantum annealer based system/simulator as we currently investigate the components of all types of architectures currently in the market. We first introduce here the different kinds of qubit models that we support at this state of research in the quantum computer engineering field. The real, realistic and perfect qubits are presented here, that can be used for either purely experimental or purely application development perspective.

A. Perfect qubits

An important concept that is introduced for our line of research is the use of perfect qubits. Companies, governments and other organisations interested in building a quantum accelerator need to evaluate the availability of quantum computing resources in terms of quantum algorithms and have a way to test the correctness of the quantum logic. To serve these needs, we use perfect qubit, such that any of the erroneous behaviour arising due to qubit quality can be avoided during application development phase. These qubit modelled in the simulator do not decohere and stay in ideal state required for the algorithm. Using these perfect qubits guarantees that the end-users can verify and check the algorithm that they are working on and test if the computed results have a meaning that can be easily interpreted. We are not the only ones who use this but it is a very clear concept that separates the two directions that we are investigating in the Quantum Computer Architecture lab. As explained above, we introduce in OpenQL, a datatype which represents the perfect qubit which has a more stable behaviour than the realistic qubits. Whether or not the nearest-neighbour constraint applies, is a discretion of the designer. The compiler may or may not compute a route for the qubits. These decisions are based on the requirement and maturity of the application development stage before translating to realistic experimental testing.

B. Industrial and societal quantum application logic

The highest layer in the full-stack focuses on the application that needs to be developed for any organisation. On current, modern architectures, there are a large number of initiatives developed that run on either the FPGA, the GPU or the TPU as the accelerator platform. When envisioning the quantum accelerator idea, many similar topics are well suited, such as security, artificial intelligence, autonomous driving, genome sequencing, sensors and trajectories for aeroplanes and rockets. For the two application examples that we are currently developing, we assume the use of perfect qubits such that the focus can be completely given to the algorithm logic and the micro-architecture design.

1) We research algorithms for accelerating quantum genome sequencing. These are motivated by the application of gene therapy and personalised medication for every single individual on earth. The treatment will be based on every person's DNA-profile that has to be generated by extensive computational processing of the reads from sequencing devices.
2) We are also working on a quantum accelerator model in collaboration with a German car manufacturer focusing on autonomous and electrical cars. For confidentiality agreements, we do not go into any detail of this project.

Given the potential of quantum acceleration, this top-down approach is necessary to understand how investing in the development of quantum computing has the potential to become a world-wide technology that can be used by every country, organisation or individual.

We focus on one such candidate application of genome sequence reconstruction. For instance, quantum computational power would be imperative if we aim want to compute the DNA-profile of every human being in the world, which takes around one week on a large network of very powerful servers for one person's DNA. With the availability of enough qubit capacity, the entire parallel input data-set can be evolved simultaneously as a superposition of a wave function. This particular property makes it possible to perform the computation of the entire data-set in parallel. This kind of computational acceleration provides a promising approach to address the computational challenges of DNA analysis algorithms. The essence of accelerating sequence reconstruction is the ability to run parallel search operations on the short reads obtained from sequencing an individual DNA from a sequencing machine, onto an already available reference of the organism. In recent years, GPU, FPGA and cluster computing frameworks like Hadoop and Spark have been used to reduce the total run-time. Potentially, quantum computation offers a fundamentally different way to address the enormous volume of data by employing superposition of reads in the search process, thereby reducing the memory requirement maybe even exponentially. The quantum search primitive (Grover's search) itself is provably optimal [3] over any other classical or quantum unstructured search algorithm. The rather modest quadratic speedup in cycles however becomes extremely relevant for industrial application due to the total CPU run-time involved in the big data manipulation (in order of 1000s of CPU hours [4] for a single human DNA sequence reconstruction).

C. Programming language, compiler and run-time support

The quantum algorithms and applications presented in the previous section can be described using a high-level programming language such as Q# [5], Scaffold [6], Quipper [7] or OpenQL [8] and compiled into a series of instructions that belong to the (quantum) instruction set architecture.

As shown in Figure 3, the compiler infrastructure for such a heterogeneous system consists of the classical compiler for the host processor combined with the quantum compiler. It is important to note that the architectural heterogeneity where classical processors are combined with different accelerators such as the quantum accelerator, imposes a specific compiler structure where each compiler part can target the different instruction sets and ultimately generates one binary file which can be executed on different instruction set architectures. For the computer architecture envisioned in our research, any high-level implementation of the system application will consist of two interleaved types of logic: the classical logic which will be executed by the micro-architecture of the controlling processor and the quantum logic which will be mapped onto the quantum processor. The quantum logic can be encapsulated by classical language structures such as decision and loop constructs. The micro-architecture extracts the quantum part and send it to the quantum processor.

As we adopt the quantum circuit model as a computational model, the quantum compiler translates the quantum logic into quantum circuits for which reversible circuit design, quantum gate decomposition and circuit mapping are needed. The output of this compiler is a series of instructions, expressed in a quantum assembly language, such as cQASM, that belongs to the defined instruction set architecture. [1] The definition of a shared quantum assembly language is a key challenge such that there is uniformity in the algorithmic descriptions of different research groups. **Perfect qubits:** The compiler can also target the use of perfect qubits. As defined above, that implies that these qubits live as long as they are needed and have principally no error-rates in the quantum gates that are executed. Depending on the state of the execution platform, connectivity constraints can be imposed for mapping and routing. When we generate everything in terms of perfect qubits, that also implies that there is no separation anymore between logical and physical qubits as there is no requirement for error coding.

[1] QASM is one candidate for such a language and was originally produced by Nielsen and Chuang to generate the LaTeX figures for the quantum circuits for their book.

D. Quantum micro-architecture

Any computer has a series of instructions which can be executed on the dominant processor. To this purpose, any kind of processor has a particular architecture capable of executing any sequence of the legitimate instructions. This also holds for the quantum processor, which also has a series of instructions that it can execute, some of which are classical logic and others are the quantum instructions that will be executed on the quantum chip. So the quantum accelerator will consist of two components: the classical and digital micro-architecture part that has a classical processor to execute part of the accelerator logic and the quantum chip that contains the qubits that need to be executed in an analogue way.

Essential to any kind of computational device is the presence of one or multiple computer architectures that are responsible for executing the instructions that are delegated to the co-processor. The architecture of a machine connects the physical hardware to the applications that can run (on that hardware) and dictates how instructions are executed. This is also true for the case of a quantum accelerator. For the quantum algorithms to be understood by the quantum accelerator, a low level representation of the quantum instructions is required that the classical control hardware of the quantum chip can understand. This is known as the Quantum Instruction Set Architecture (QISA). The content of the QISA can be modified for each accelerator logic that needs to be implemented. For any micro-architecture, there are a number of properties that we have to estimate, such as the appropriate instruction-length, pipeline depth (for parallel quantum gates) and targeting multiple control channels per single instruction. Based on these principles, the basic blocks are constructed, such as timing control unit and the microcode instruction set of the overall micro-architecture. We do not yet have a full implementation of the micro-architecture for logic expressed in terms of the perfect qubits.

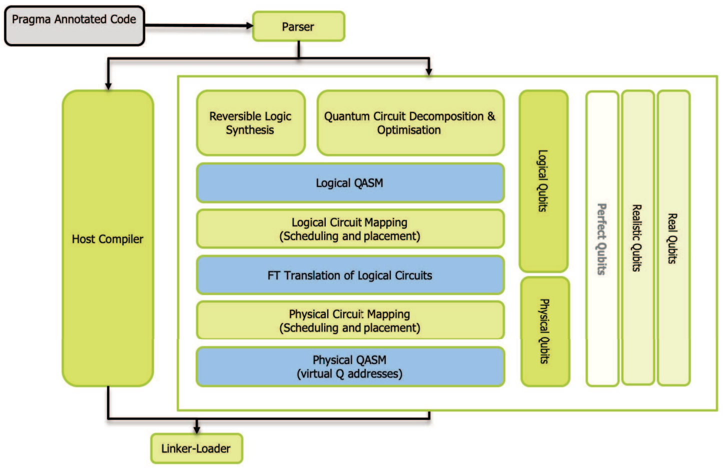

Pragma Annotated Code
Parser
Host Compiler
Reversible Logic Synthesis
Quantum Circuit Decomposition & Optimisation
Logical QASM
Logical Circuit Mapping (Scheduling and placement)
FT Translation of Logical Circuits
Physical Circuit Mapping (Scheduling and placement)
Physical QASM (virtual Q addresses)
Logical Qubits
Physical Qubits
Perfect Qubits
Realistic Qubits
Real Qubits
Linker-Loader

Fig. 3: Compiler infrastructure

E. *Mapping of quantum circuits*

Mapping of quantum circuits is considered in two different contexts: the first is when applied on small real quantum processors and the second one targets a simulation engine that addresses larger number of qubits. Depending on the test objective, we can either take into account large number of qubits or stay at a small scale and closer to the experimental state-of-the-art.

Perfect qubits: When the algorithmic behaviour and content is not yet defined, which is the case in most of the situations, it is important to be able to use perfect qubits that are more reliable and predictable than the experimental ones, as that have no decoherence and execute reliably the quantum gates of the quantum circuit.

- **Scheduling of operations:** With perfect qubits, we have the freedom to impose or relax similar kind of restrictive scheduling instructions on their behaviour.
- **Placement and routing of qubits:** Also for this feature, it depends on how much freedom the algorithm designers needs to experiment with the algorithm they are designing. The more restrictive we are in the placement and routing, the more difficult is becomes. In a more relaxed situation, the designer enjoys more possibilities to experiment and test the algorithm.

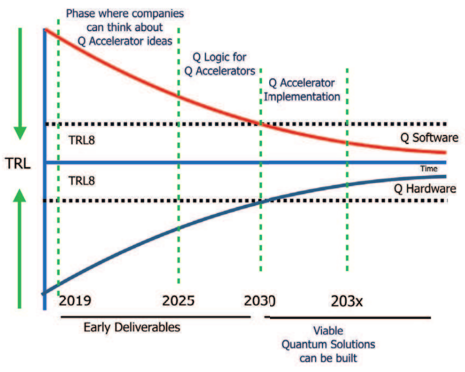
F. *QX simulator*

The QX simulator was developed in our group as a platform to simulate quantum operations on either realistic or perfect qubits. The QX engine can execute any quantum logic expressed in OpenQL and translated by the compiler to cQASM, the common quantum assembly language. The assumed micro-architectural layer encapsulating the QX simulator executes the cQASM instructions by sending the quantum instruction to QX, which then executes it, measures the qubit states and sends back the results to the micro-architecture. The QX simulator is scalable based on the underlying host processor, and is capable of simulating with up to 35 fully-entangled qubits on a laptop PC, which are either perfect or realistic. The main advantage of a platform like QX is to provide application developers, computer scientists and computer engineers the tools to model and test designs before experimental implementation on quantum processors. A order of 50 fully entangled qubits already give a lot of possibilities to test the application in a proof-of-concept simulation. We can also use the different kinds of qubits that we presented in this paper.

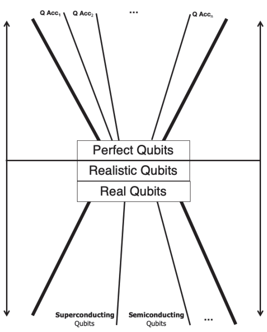
Perfect qubits: For application development, there is the need to execute the quantum logic to verify the computed results of the algorithm in the functional sense. The QX simulator is capable of assuming the non-emergence of errors. The current stage of research on quantum genome sequencing algorithm uses the QX simulator in this mode of development. In principle, any universal quantum logic can be executed on the simulator, the result can be measured and fed back to the micro-architecture.

III. Future Prospects and Conclusion

It is very important that companies and other organisations start investing as soon as possible in Quantum Technology. Figure 4 shows a projection of when different parts of software and hardware development will



(a) Development time frame



(b) Structural division between perfect and realistic qubits

Fig. 4: Quantum computer development future projections

be required, to create an efficient quantum computer. The distinction is made between the use of quantum accelerators and that of manufacturing a quantum chip. In general, any commercial or other organisation is interested in new technology if the Technology Readiness Level (TRL) is high enough. If we adopt the same levels as for classical technology, the TRL needs to have reached level 8 and that is sketched in the red and black line that are shown in Figure 4(a). There are 4 different moments. Phase I focuses on the reflection by the organisation on the concrete need that exists and for which a quantum accelerator logic can be developed. Phase II resembles the team members brainstorming on the logic for the quantum accelerator. Phase III then focuses exclusively on the actual implementation and execution of the Quantum Accelerator logic, whether on an experimental quantum chip or on the QX simulator. This is the moment when the top and low curves can be combined in a real quantum prototype of the accelerator. Figure 4(b) represents the way that the two lines of research are currently separated and which will be joined in maybe over the next decade.

Over the last couple of decades, quantum computing has been a one-dimensional research effort focusing on understanding how to make coherent qubits and how to implement the different universal quantum gate sets on any of the multiple quantum approaches. As far as computer architectural choices were made, the community has been focused very much on the von-Neumann computer architecture and defined qubits in terms of memory and processing qubits. Two approaches seem to be very promising: the first comes from the accelerator community and involves the full stack integration of the different layers that are needed to build the quantum accelerator. The use of perfect qubits in that context makes sense as the end-users of any quantum accelerator can focus their reasoning on the quantum logic of the application and verify it through some implementation of the micro-architecture and the execution of the quantum instructions on the quantum simulator. The second option is to use the full-stack for the control of, for instance, superconducting and semiconducting qubits with a micro-code layer where we translate any kind of common QASM into an operational set of micro-instructions, for a meaningful adoption of existing computer technology. It is very difficult as that also depends on the quantum application that is being looked at and the way the qubits are manufactured.

References

[1] Vassiliadis, S. *et al.* The molen polymorphic processor. *IEEE Transactions on Computers* **53**, 1363–1375 (2004).

[2] Preskill, J. Quantum computing in the NISQ era and beyond. *arXiv:1801.00862* (2018).

[3] Zalka, C. Grover's quantum searching algorithm is optimal. *Physical Review A* **60**, 2746 (1999).

[4] Houtgast, E. J., Sima, V.-M., Bertels, K. & Al-Ars, Z. Hardware acceleration of bwa-mem genomic short read mapping for longer read lengths. *Computational biology and chemistry* **75**, 54–64 (2018).

[5] Svore, K. *et al.* Q#: Enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the Real World Domain Specific Languages Workshop 2018*, 7 (ACM, 2018).

[6] Abhari, A. J. *et al.* Scaffold: Quantum programming language. Tech. Rep., Princeton University (2012).

[7] Green, A. S., Lumsdaine, P. L., Ross, N. J., Selinger, P. & Valiron, B. An introduction to quantum programming in quipper. In *International Conference on Reversible Computation*, 110–124 (Springer, 2013).

[8] Khammassi, N. *et al.* Openql 1.0: A quantum programming language for quantum accelerators,. *QCA Technical Report* 8 (2018).