Introduction to the building blocks of a quantum computer

When we build a computer, we have to integrate different kinds of technologies such as those for the memory, bus interconnect, processor chips but also the peripheral devices such as a keyboard or a even screen to be able to interact with the machine. This lesson discusses the big picture of a quantum computer: namely from how to program it, to reading out the result.

Now the first word we want to emphasize is the word "compute". Even though QuTech is created to build a quantum computer together, we are actually not really building a quantum computer, but we are building a quantum accelerator. Namely, a computational devise that can be connected to a classical processor that will provide the performance for a series of applications that we can never reach classically.

Here you see the global view of what we currently understand as a heterogeneous multicore architecture. Heterogeneous because we have different kind of accelerator technologies. We have an FPGA which stands for the field programmable gate array. We have GPU's, for Graphic Programming Unit. These are the vector processors we are using to produce graphics computation. The third alternative accelerator technology will be a quantum co-processor which has the quantum properties that will provide a substantial increase in the compute power.

This is basically what we see. That also means that when you write any application, you will most likely end up using different kinds of accelerators including the FPGA, the GPU and also the quantum accelerator and therefore your application has to be compiled for four different instruction sets; namely, your Intel processor for instance on the classical machine, your FPGA instruction set, the GPU instruction set, and also the quantum instruction set.

That is very important to understand, so whatever we are going to discuss in the following talks is only going to focus on the quantum accelerator device or the application processes that we need classically as well as quantumly to be provided.

Whenever you talk about a computer, classically we have divided that in different layers; the lowest layer is always let's say the hardware, the chip that has been designed. It is never a single chip or a single processor. No, it has memory, it has interconnections, so a bus that allows the processor to communicate with the memory to retrieve instructions and to retrieve or produce the data.

This is shown here. And then it goes up to the microarchitecture that we need, up to the application level.

I will now go in detail on each of those layers in a quantum context.

Because we are basically adopting the same kind of layer view of what a quantum accelerator or a quantum computer would be, and we simply put the Q of quantum in front of every layer. And that is our research and working program.

The highest level is the quantum algorithm that we know of. We don't even know yet what they will be. We have examples such as factorization that is used in cryptography in securing communication between machines. But quantum algorithms can as well be designing a new molecule for personalized medicine. It can be that you might want to have a climate module that you want to run on your quantum accelerator that take all kind of mechanisms into account that we currently are unable to compute or even model on a classical machine. So that is the quantum algorithm layer, and that is where the biggest opportunity lies worldwide. Where many companies and organizations can start developing their own quantum application. Because every company or every end-user can think of how they can use that computational aspects of such a quantum device.

One layer lower is that when you have an application that you need to program. You do that classically. You have a programming language like C++, Fortran, Cobol or any language that you can think of. Those languages can produce the code for a classical processor. But we also have to develop our own quantum language for the quantum accelerator.

There are a couple of languages that have been developed so far. There is ScaffCC, and ProjectQ. And here at QuTech we are developing our own programming language called OpenQL, inspired by OpenCL, which is a language developed for GPU programming. We are now shifting it to the quantum infrastructure, so making the changes to that language.

So that is the programming language layer.

For every programming language we need a compiler. A compiler takes the input of your algorithmic logic and compiles it into a lower level language that is classically called an assembly language. Here we are working on a quantum assembly language which we call QASM, which was originally developed for a book 'Quantum computation and information' by Nielsen and Chuang. To generate the figures in their book of the quantum circuits. We just expanded that language into a full-fledged quantum assembly that is being generated by our OpenQL compiler, which is the programming language that we also developed. So that you can express your quantum logic in such a way.

We are internationally collaborating with other partners working on similar things such that we standardize this quantum assembly language. Because for now everybody has its own local version, and that is not very good that everybody has his own variation, but if we all agree that this is what we assume to be QASM, then progress will come much faster.

The next layer is quantum arithmetic's because the mathematics of what you need to do is completely different than classically. The quantum gates operate quite differently, that's why you need to develop the quantum arithmetic; how to do a quantum operation.

I will not say anything about run-time, which is another part that we need in a quantum accelerator. Because we will need it, but what kind of functionality it should have is a bit unclear right now. That is where there is the tension between the compiler development

because we still can develop a lot of things in the compiler and maybe at a later stage we will develop that in a run-time support.

All of this QASM basically maps very well one on one on with the quantum instruction set.

This quantum instruction set basically describes what the operations are that your quantum device is capable of executing. That is why we have to think of what these instructions are. We know that classically we use an assignment like A = B + C. We should be able to do something similar in a quantum device. But it is not as simple as retrieving data from a memory location and perform the addition and writing back the result, because in quantum we use qubits.

A qubit is a quantum bit. Classically, we reason in terms of bits, zeros and ones. And as you know we are now using qubits, quantum bits. These can also be zero or one too, but they can also be zero and one at the same time. And that is the famous superposition that we are exploiting in a quantum device.

We can also combine two qubits so we don't have two different states but we have a combination of those states; namely 4 states at the same time.

If I combine 3 qubits, I have 2^3 = 8 states.

Now what is very nice about quantum is that the quantum mechanics gives us parallelism for free. Because I can apply quantum gates on those 2^n different states

You will come to understand in the upcoming lectures that nothing comes for free however. There is still a lot of challenges that need to be resolved. But that is ultimately the big challenge and the big opportunity that quantum offers.

That is why you have to understand what this instruction set is and the corresponding architecture should be.

And that immediately brings us to the layer of the micro-architecture.

Just like any classical processors we have also a quantum micro architecture for my quantum device, which contains the processor, the memory and also the interconnects of how the processor will communicate with the qubits. It has local registers in the processor and an ALU, an arithmetic logical unit so that it can compute logical and arithmetical operations, and write back the result to memory so that the user gets an idea of what the algorithm has computed as a result.

So in the quantum case we have a micro-architecture which has a similar kind of functionality. And that is the one we are also currently implementing in a real device that already controls a number of the physical qubits; a superconducting as well as a spin qubit that we are developing at QuTech.

For now we are working on a 17 qubit micro-architecture, so that in principle we can go up to 2^17 parallel executions on the combination of those 17 qubits.

A layer lower is the quantum to classical layer. Because whatever you perform on the quantum level is always an analog phenomenon.

Now you say; "analog? I thought we were building a computer? Which should be digital..". Well, everything up to the micro-architecture is clearly digital, but ultimately what you send down to the quantum chip is a for example a microwave, it can be other things, but let's say it is a microwave and we control an individual electron at the atomic level. The individual electron is important to understand. So if I have 17 qubits, I basically have 17 electrons, not hundred thousands, not millions, but 17. And there are ways to combine those 17 electrons and their spins in the way that they interact and move that indicates that they are doing a particular calculation.

So that means that this layer is necessary for translating all of the logical steps that you need to do in your algorithm into the appropriate microwave or the physical signal that you want to send to this electron and to the qubit.

And then ultimately it enters into the quantum chip. Which consists of these qubits which are connected to each other. And then we hope of course that we get a meaningful result. Now it is never the less important to understand for a quantum accelerator for any computational device is that it is a non-deterministic way of computing. That means that it is not like in a classical machine that you run a thousand times the same algorithm and you will get a thousand times exactly the same result. Quantumly this is absolutely not true, because when you want to read out the result several things happen. The most important is that any entangled superposition that exists, actually is going to get destroyed.

So if I have for example 2^17 possibilities, I am only going to get one of those possibilities back as a result. And all the others will disappear. And that is why you maybe have to do a computation 10 times, a 100 times, we don't even know how many times we need to do that. And then you can make a histogram of what has been computed, and the readout that has the highest frequency of occurrence has a high probability of being read by our micro-architecture and that is what we can report back to the end user.

So that is something that you should not forget. A quantum device is a very powerful device, it gives massive parallelism in principle. But we need multiple runs and average out what those calculations of those results are. And the one with the highest frequency is the most likely result of your quantum device.