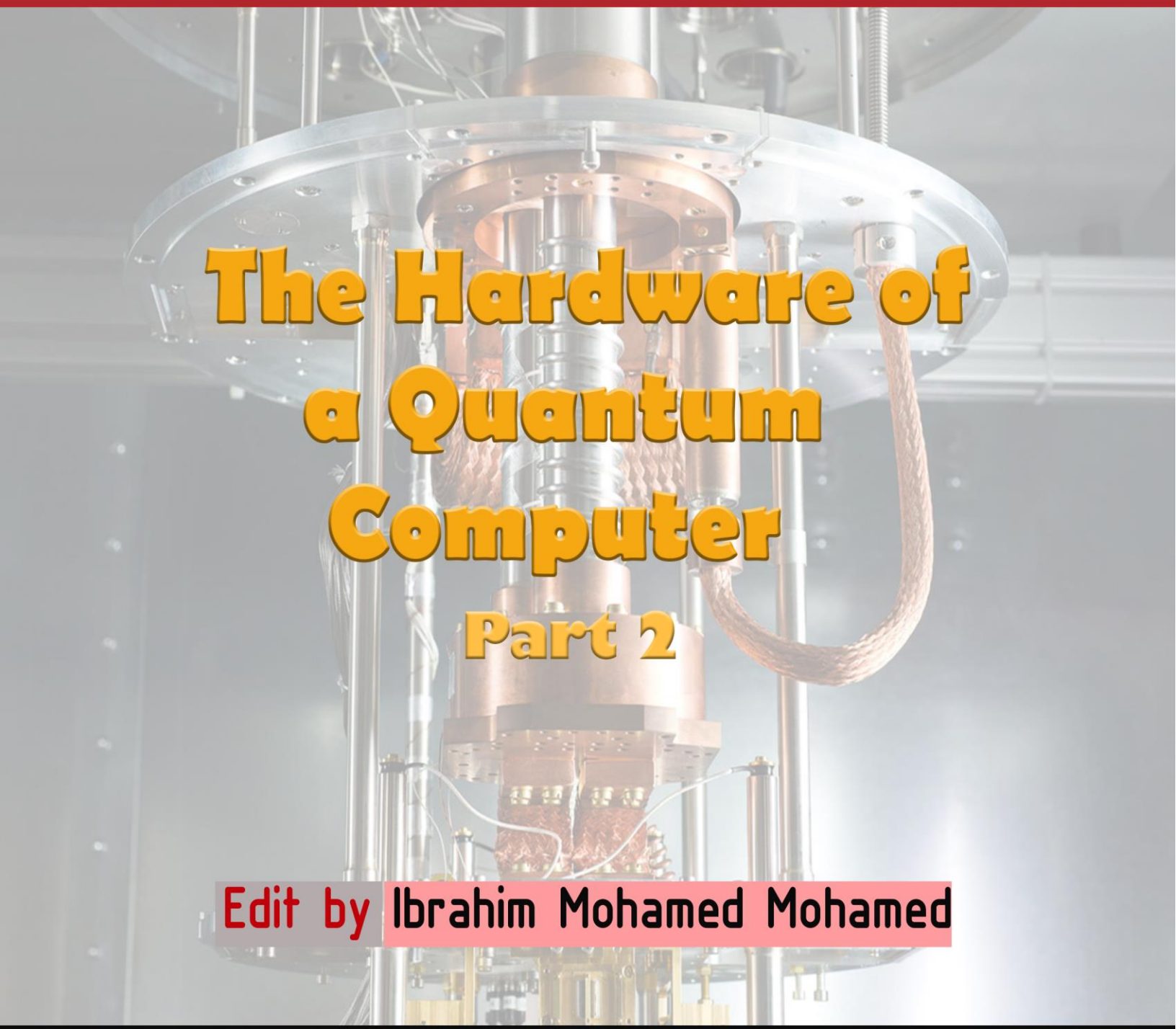


The Hardware of a Quantum Computer

Part 2

Edit by Ibrahim Mohamed Mohamed

Architecture, Algorithms, and Protocols of a Quantum Computer and Quantum Internet


بسم الله الرحمن الرحيم
وقل رب زدني علما
١٤٢٩

Contents

About The Editor:

Ibrahim Mohamed

- *Advanced Technical School for Information Technology .*
- *Electrical Engineering. Suze Canal University .*
- *"Self Study" Aerospace Engineering .Tu Delft university and NPTEL*
- *"Self Study".....More*

Welcome

Welcome to this course

Dear learners,

Welcome to this second part of the Professional Certificate in Quantum 101: Quantum Computing & Quantum Internet.Usually, lectures, articles and Videos about quantum computation mostly talk about qubits themselves, and all the quantum phenomena that make qubits exciting, futuristic, and almost impossible to fully understand.Qubits on their own, however, don't form a working quantum computer. The qubit might be the most spectacular element of the computer, but many more layers are needed to actually work with the quantum phenomena that give the quantum computer its great powers. This Professional Certificate will touch on all of these layers: from the devices which bridge the gap between the quantum chip and the classical control hardware, to the mathematical aspects of some quantum algorithms. This is the second part of the program, "Architecture, Algorithms, and Protocols of a Quantum Computer and Quantum Internet": in the first part, we discussed the physical materials, four different types of qubit, and operations on these qubits. We encourage you to follow this course as well, to get a full overview of the quantum computer. This second “MOOC”, part 2, will dive deeper into the other layers of quantum computers and a quantum internet, such as the classical electronics/hardware and the different types of classical and quantum software involved. We hope you enjoy the course with us!

Kind Regards,
The QuTech Team

About the course

In this course, you will learn:

- The functionalities of the building blocks of a quantum computer, including micro-architecture, compilers, quantum error correction and quantum algorithms, and the challenges in realising these systems and interconnections.
- Deeper information on the building blocks of a quantum internet, and the protocols and networks needed to bring the quantum internet into reality.
- The scientific principles behind the quantum computer that make quantum technologies possible.

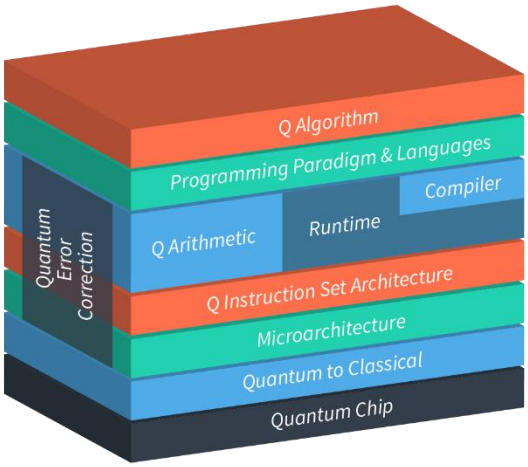
Get the most out of this course!

This course aims to give you an understanding of the scientific basis behind the quantum computer and a quantum internet. Our challenge to you is to really understand the working principles of the different layers of a quantum computer and a quantum internet. This course is about getting an insight into the electronics, hardware and software that are essential for a practical quantum computer and a quantum internet. We will inspire each other and we ask that you bring in your experiences, your insights and your thoughts via the forums.

Module 1: Introduction to Architecture, Algorithms, and Protocols of a Quantum Computer and Quantum Internet

Introduction to Module 1

In the first part of the course, we separated the execution of a quantum algorithm into a set of layers:



The topmost of these layers represents the abstract description of the algorithm, which may ignore various aspects of the device we wish to run it on. In order to execute it on a device of our choosing, we have to translate it into a series of instructions that our device will support, and subsequently into physical signals which can be used to control qubits 'on the chip'. In this module, Carmen G. Almudever begins by reviewing this 'quantum stack', as well as its lowest layer, the physical qubits themselves. In the subsequent lectures, Fabio Sebastiano describes the challenges in constructing the "Quantum-to-Classical" layer, second from the bottom.

These three lectures (and two quizzes) begin our journey upwards through the stack. We hope you find them exciting and enlightening.

Introducing the architecture, algorithms, and protocols of a quantum computer and quantum internet

In this first lecture of the course, Carmen will set out the scope of this “MOOC” and we will quickly look back at the information covered in the first course of this Program: The Hardware of a Quantum Computer (formerly entitled "The Building Blocks of the Quantum Computer - Part 1"). Koen will set out the different layers of the quantum computer and Menno will summarize and re-introduce the four different types of qubits that have been discussed in The Hardware of a Quantum Computer.

Note: the original lectures by Koen Bertels (introduction to the building blocks of the quantum computer - part 1) and Menno Veldhorst (closing lecture - part 1) can be found under this introduction Section for your reference.

Introduction to the building blocks of a quantum computer - part 2

Welcome to the second part of the course about the building blocks of a quantum computer and a quantum internet. The first person who formulated the idea of a ‘quantum computer’ was the physicist Richard Feynman in 1981. In his talk, Simulating Physics with Computers, he postulated that to simulate quantum systems you would need to build a quantum computer. At that time, the quantum computer was only a theoretical nice idea but nowadays it is a reality. Here, at QuTech, physicists and engineers are working together with industrial partners, such as Intel and Microsoft, towards the goal of creating a practical quantum computer. In part 1 of this course, you have been introduced to the layers of a quantum computer by Koen Bertels. Although we commonly speak of the quantum computer, this term is actually a bit misleading; one could better speak of a quantum accelerator. When designing a quantum computer, we try to make use of the knowledge in classical computing. We are basically adopting the same kind of layer view of what a quantum accelerator or a quantum computer would be. Let’s shortly look back at Koens introduction about the building blocks of a quantum computer. 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. That’s 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 layered 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 takes 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 actually 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 device. One layer lower is that when you have an application that you need to program. You do that classically. Again, you have a programming language such as C++, Fortran, Cobol or any other 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 accelerator, the quantum accelerator. There are a couple of languages that have been developed so far. There is ScaffCC, there is ProjectQ. And here at QuTech we are developing our own programming language called OpenQL, which is 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. Again here at QuTech, we are working on a quantum assembly language which we call QASM, which was originally developed for a book 'Quantum computation and quantum 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 everbody 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 with 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, 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. And they can also be zero or one, 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 each qubit, two different states; namely 4 states at the same time. If I have 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 really comes for free. 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 also 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 again her 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 nevertheless 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, maybe 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 out by our micro-architecture and that is what we can report back to the end user. So that is

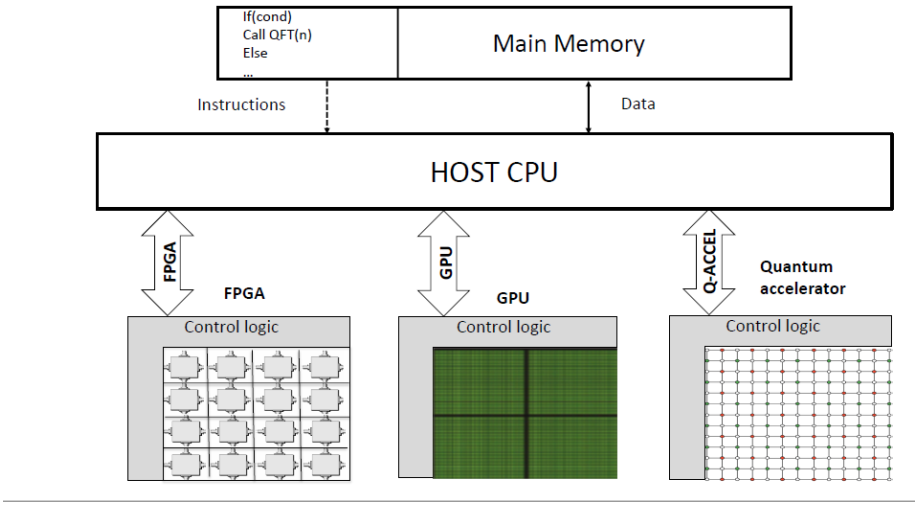
something that you should not forget. A quantum device is a very powerful device, it gives massive parallelism in principle. But we need multiple runs and average out what those calculations of those results are. And the one with the highest frequency is the most likely result of your quantum device. In the first part of this course, we took some time to introduce you to quantum materials and ket notation. We talked about the basic unit of information in quantum computing and quantum internet, the qubit. Let's have a look at the 4 types of quantum platforms that have been discussed: In the first part of this MOOC we mainly focused on the lowest level, the quantum chip that forms the foundation. The quantum chip is the quantum circuitry hosting the physical qubits. Physical qubits can be controlled to a superposition state and coupled together to create entanglement. These principles are on the basis of a quantum computer and quantum internet. The quantum mechanical state of a qubit is usually fragile and can suffer from environmental interactions and decohere over time. However, in order to do operations on these qubits it is essential that many operations can be done within the qubit coherence time. Theoretical predictions state that for practical fault tolerant quantum computing it will be necessary to perform thousands of operations within the qubit coherence time. While more sophisticated quantum error correction protocols are actively being studied, it is clear that good qubits are essential and that qubits need to be of high-fidelity. Today, various qubits are being explored and investigated to reach this goal. In this series of lectures we have introduced to you several of the most promising qubit systems. Qubits based on the spin states of electrons or nuclei associated with defects or donors in silicon or diamond can reach very long coherence times. In materials with low net nuclear spins, detrimental magnetic interactions with the environment are small. In addition, the strong confinement leads to small overlap with other states. NV centers in diamond are particularly interesting, as they can be coupled to photons, providing an optical link between spin qubits that are distant from each other. Spin qubits in quantum dots also exhibit very long coherence times, but they also offer the advantage of being manmade. This allows to realize qubits at a predefined location, which is clearly beneficial for scaling up the number of qubits. Superconducting transmons offer this advantage as well. In addition, they are larger and so, at least in the few-qubit regime, it is even easier to fabricate these types of qubits. Large efforts are devoted to improving the qubit environment leading to qubits that become more and more isolated and thus to qubits with extended coherence. This is achieved by, for example, removing magnetic noise from nearby nuclear spins or electric noise stemming from charge defects in the substrate. In addition, clever qubit designs enable to decrease a qubit sensitivity to noise. For spin and superconducting qubits, sweet spots exist where to first order qubits are insensitive to certain noise. A special class of qubits exists that can be made intrinsically insensitive to some noise. These are called topological qubits and are qubits such as the ones based on emergent Majorana fermion states. These qubits could become exponentially less sensitive to local noise with increasing system size. That is, by increasing the separation between Majorana fermions, the qubits become more and more protected against local noise and can thus hold their coherence for a longer time. Today, we do not know what the best qubit platform will be and so there is an active race going on between all these different platforms. But how do we go from quantum hardware to a quantum computer? In this second part of the course, we will introduce you to the other layers of the quantum system stack. We will also talk about quantum internet. At QuTech, the Quantum Internet and Networked Computing group, is working on realising a quantum network that connects four different cities in The Netherlands. I hope that you will enjoy this course, and it becomes clear that building a Quantum Computer and a Quantum Internet requires different expertises. Such as physics, mathematics, computer and electrical engineering. Good luck!

Main takeaways

- The building blocks for a quantum computer are a description of the quantum algorithm to be executed, a quantum language, a compiler, arithmetic, an instruction set, a micro-architecture, a quantum-to-classical conversion system, and a quantum chip.
- Quantum computing is not like classical computing where you can run the same algorithm thousands of times and get exactly the same result in all of them. In quantum computing, we average the results of multiple runs. The result with the highest frequency is the most likely result of the quantum computation.
- Several of the most promising qubit types that are being researched are the NV-centre qubit, the spin quantum dot qubit, the superconducting transmon qubit, and the topological qubit, based on Majorana fermions. What the best qubit platform will be, is still unknown.

Extra Section : lectures of Building Blocks part 1

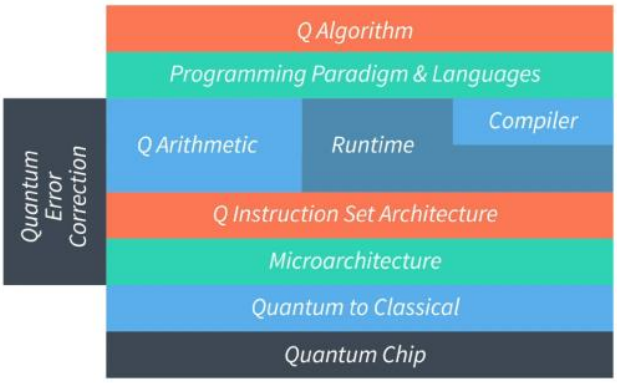
In case you have missed part 1 of this course, we include the two lectures that briefly summarize the content of the first course, which was mainly about the the lowest layer of the quantum computer: the qubits.Building Blocks of a Quantum Computer - part 1 is also open as a self-paced course on EdX (enrol here), we recommend you to take this course as well to get a full overview of the layers of a quantum computer. It is not compulsory, but the content is of great help in fully grasping the content of Building Blocks part 2. In Building Blocks of a Quantum Computer - part 1, we introduce you to quantum materials and ket notation, and the four types of qubit systems that QuTech researchers are investigating.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.



Koen Bertels ,Building Blocks of a Quantum Computer

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.



Koen Bertels ,Building Blocks of a Quantum Computer

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

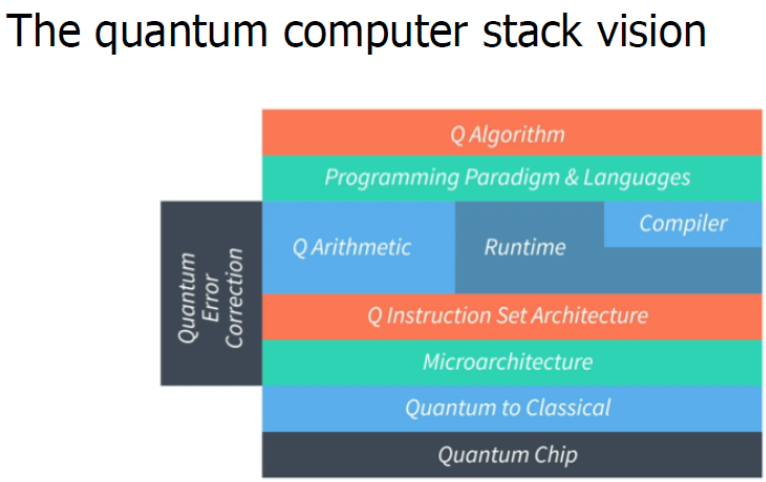
using qubits, quantum bits. These can also be zero or one too, but they can also be zero and one at the same time. And that is the famous superposition that we are exploiting in a quantum device. We can also combine two qubits so we don't have two different states but we have a combination of those states; 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.

Main takeaways

- Current efforts are put into building a quantum accelerator, a computational add-on to the classical computers to exceed current performance.
- Heterogeneous architectures include special purpose devices, such as Field Programmable Gate Arrays, Graphics Programming Units and, soon, quantum co-processors.
- The building blocks for a quantum computer are quantum algorithms, quantum programming languages, quantum compilers, quantum arithmetic, quantum instruction sets, quantum micro-architectures, quantum-to-classical conversion systems and quantum chips.

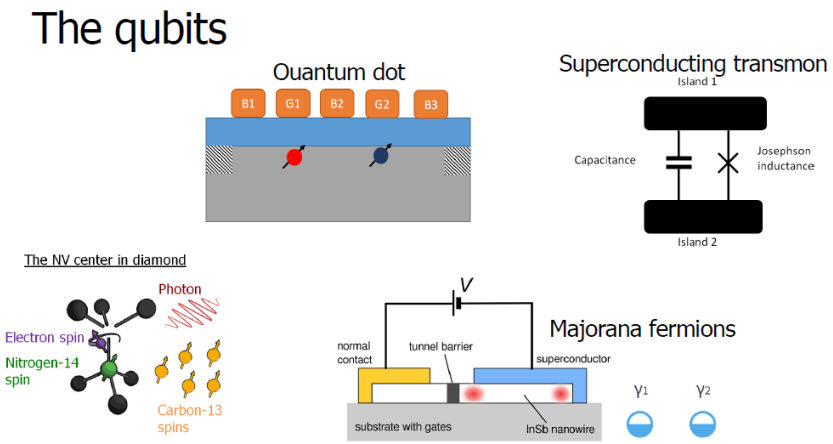
Extra Section : Closing lecture - part 1

This is the closing lecture of the first part of our series of lectures on the building blocks of a future quantum computer. We started by introducing a stacked layer approach toward a future quantum computer. In part 2 of this MOOC we will focus on the higher layers of this stack.



Menno Veldhorst,Closing lecture

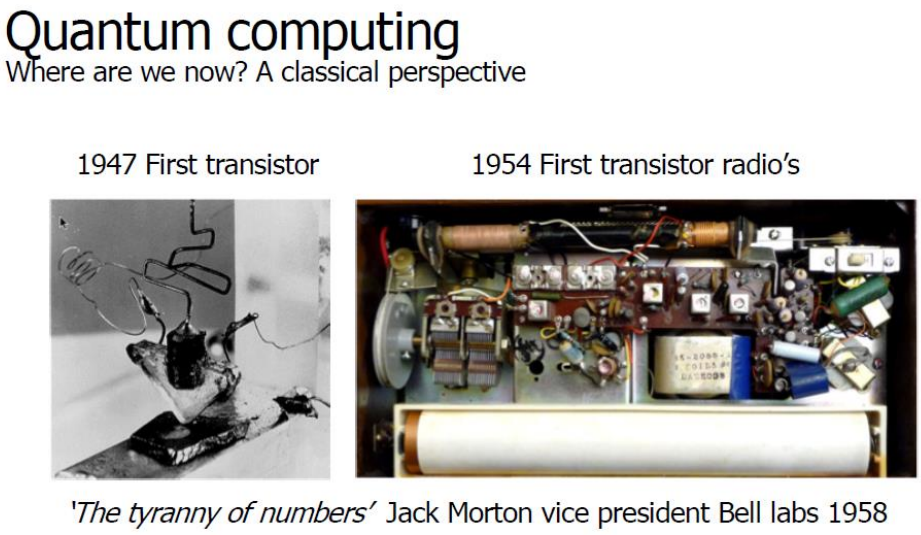
In addition we will also discuss the path toward a quantum internet. A quantum internet allows for communication between nodes where the security of the communication is dictated by the laws of quantum mechanics. In the first part of this MOOC, we mainly focussed on the lowest level, the quantum chip that forms the foundation. The quantum chip is the quantum circuitry hosting the physical qubits. Physical qubits can be controlled to a superposition state and coupled together to create entanglement. These principles are on the basis of a quantum computer and quantum internet. The quantum mechanical state of a qubit is usually fragile and can suffer from environmental interactions and decohere over time. However, in order to do operations on these qubits it is essential that many operations can be done within the qubit coherence time. Theoretical predictions state that for practical fault tolerant quantum computing it will be necessary to perform thousands of operations within the qubit coherence time. While more sophisticated quantum error correction protocols are actively being studied, it is clear that good qubits are essential and that qubits need to be of high-fidelity. Today, various qubits are being explored and investigated to reach this goal. In this series of lectures we have introduced to you several of the most promising qubit systems.



Menno Veldhorst,Closing lecture

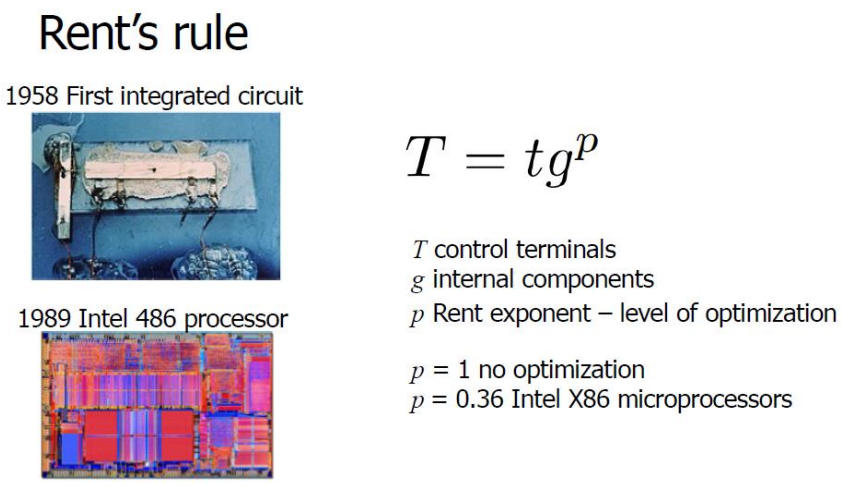
Qubits based on the spin states of electrons or nuclei associated with defects or donors in silicon or diamond can reach very long coherence times. In materials with low net nuclear spins, detrimental magnetic interactions with the

environment are small. In addition, the strong confinement leads to small overlap with other states. NV centers in diamond are particularly interesting, as they can be coupled to photons, providing an optical link between spin qubits that are distant from each other. Spin qubits in quantum dots also exhibit very long coherence times, but they also offer the advantage of being manmade. This allows to realize qubits at a predefined location, which is clearly beneficial for scaling up the number of qubits. Superconducting transmons offer this advantage as well. In addition, they are larger and so, at least in the few-qubit regime, it is even easier to fabricate these types of qubits. Large efforts are devoted to improving the qubit environment leading to qubits that become more and more isolated and thus to qubits with extended coherence. This is achieved by for example removing magnetic noise from nearby nuclear spins or electric noise stemming from charge defects in the substrate. In addition, clever qubit designs enable to decrease a qubit sensitivity to noise. For spin and superconducting qubits, sweet spots exist where to first order qubits are insensitive to certain noise. A special class of qubits exists that can be made intrinsically insensitive to some noise. These are called topological qubits and are qubits such as the ones based on emergent Majorana fermion states. These qubits could become exponentially less sensitive to local noise with increasing system size. That is, by increasing the separation between Majorana fermions, the qubits become more and more protected against local noise and can thus hold their coherence for a longer time. Today, we do not know what the best qubit platform will be and so there is an active race going on between all these different platforms. We also often find out that developments made in one platform can be implemented into other platforms and so all these activities strongly benefit from each other. Where are we now with quantum computing? To answer this question, let's take a classical perspective. In the 1950s, electrical systems contained many components, each requiring to solder to numerous others.



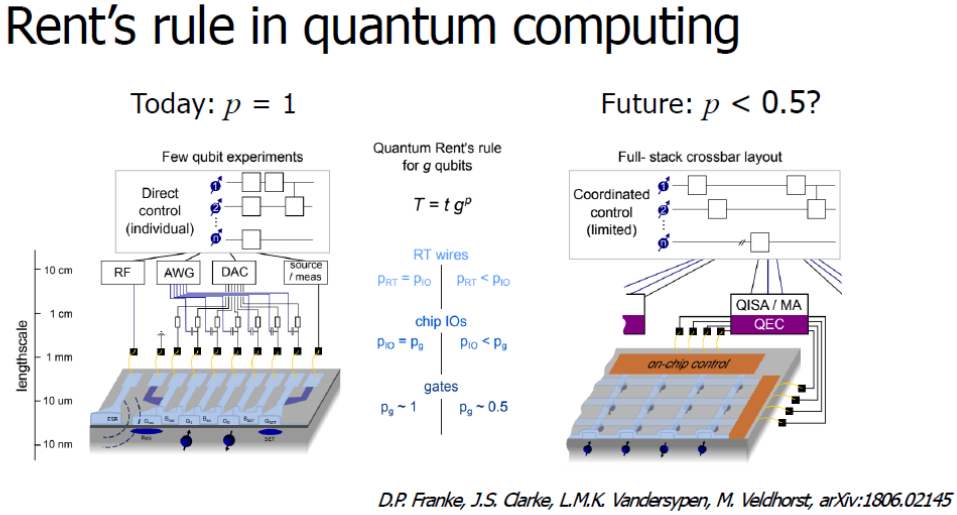
Menno Veldhorst,Closing lecture

This clearly prevented to go to large numbers. Jack Morton, at the time vice president of device development at Bell labs, referred to this situation as the 'tyranny of numbers'. Quantum devices are now at a similar stage of maturity. For classical electronics the solutions was the invention of the integrated circuit. The power and beauty of these systems become apparent if we compare the numbers of connectors at the outside of the chip to the number of active components inside the chip. Large grid arrays have a couple of thousand connectors, but these can address billions of transistors on a classical processor. This huge ratio between components and connectors is described by Rent's rule and this rule has been one of the main drivers behind Moore's law.



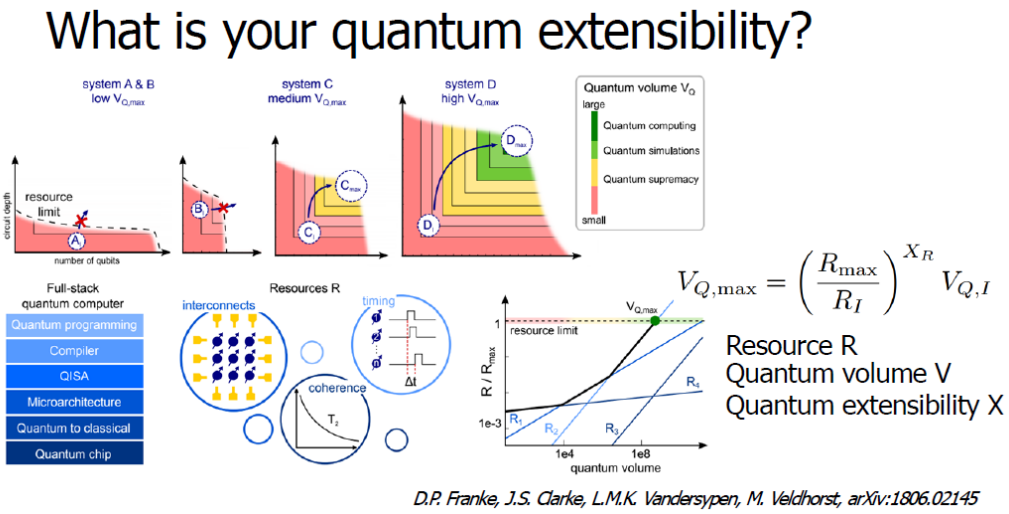
Menno Veldhorst,Closing lecture

It is thus most likely that we will require for practical quantum information applications the development of a 'quantum integrated circuit'. Today's quantum devices require individual electronics and connects for each and every qubit.



Menno Veldhorst,Closing lecture

Increasing these systems to large numbers will clearly require a different scaling law. Concepts from classical memory technology, such as crossbar layouts where signals only have to come from the sides to address a large array, are now being proposed as an efficient way forward for quantum systems. These approaches come with their own challenges, but do provide a powerful method to avoid an interconnect bottleneck. Interconnects are not the only challenge that need to be addressed. Every critical resource must be evaluated and optimized. In the figure here you see some fictitious qubit platforms. The functionality of each qubit platform is described according to its quantum volume. The usefulness of a platform will depend on the number of qubits available, but also on the amount of operations that can be executed on them. A system with only a few qubits is not powerful, even if they are of very high-fidelity. Likewise, a system containing many qubits of very low quality is also practically useless. The quantum volume takes this into account and is defined as the square of the lowest number of these two quantities.



D.P. Franke, J.S. Clarke, L.M.K. Vandersypen, M. Veldhorst, arXiv:1806.02145

Menno Veldhorst,Closing lecture

Increasing the quantum volume will enable to reach quantum supremacy, the era where quantum systems become too complex to be efficiently simulated with classical computers. Further increasing the quantum volume will enable to perform useful practical quantum simulations and finally to reach practical quantum computing. Nonetheless, while this is certainly a useful metric, the quantum volume does not provide the information whether a system is capable of reaching these applications. This scalability aspect is described by the quantum extensibility of a system. Depending on all the resources, required at every position of the stack, a platform has the capability to scale up toward practical quantum information. Defining these resources is a crucial aspect for quantum information research today. We have learned the aspects of some of these resources in the first part of this course, such as qubit decoherence. In part 2 of this course we will introduce the other building blocks of a quantum computer and show you the vision of a full-stack future quantum computer. In addition, we will also introduce to you the foundations of a quantum internet. What is required and where are we now? While building a quantum computer and quantum internet is perhaps the greatest scientific challenge of this century, you will also learn in part 2 why it is completely worth it and you will hear all the opportunities that arise once this becomes a reality.

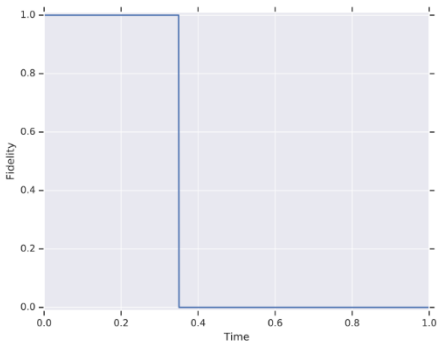
Main takeaways

- Physical qubits can be placed into superposition states, and coupled together to create entanglement. These principles are the basis of a quantum computer and quantum internet.
- The qubit implementations you have been taught in this course are the spin quantum dot, the NV-centre, the transmon, and Majorana fermions. Which qubit platform is the best is not yet clear.
- Today's quantum devices require individual electronics and connections for each and every qubit. Increasing these systems to large numbers will clearly require a different scaling law.
- This scalability aspect is described by the quantum extensibility of a system. Depending on all the resources required at every layer of the stack, a platform has the capability to scale up toward practical quantum information.

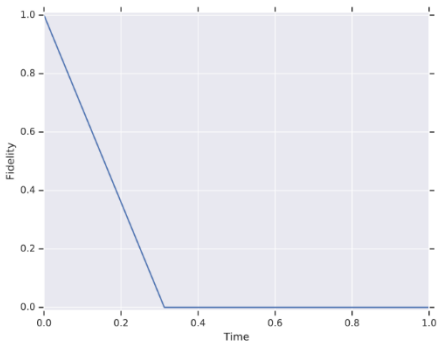
Practice Questions

In the Section , Menno noted that the state of a qubit is fragile, and suffers decoherence over time. We measure decoherence using a quantity called *fidelity*, which is always between 0 and 1. Fidelity measures the similarity between two quantum states. Two pure qubit states which are orthogonal have a fidelity of 0, and two states which are identical have a fidelity of 1. To visualise time-dependent decoherence, we use the fidelity between an initial state at t=0 and an evolved state at some later time t.

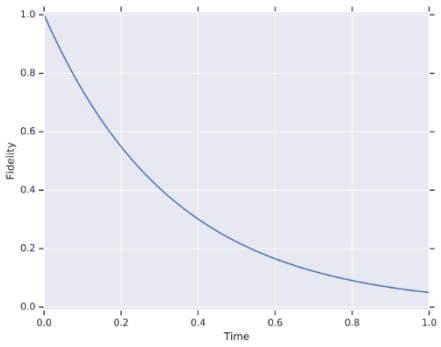
Here are four potential curves showing the decrease in fidelity over time:



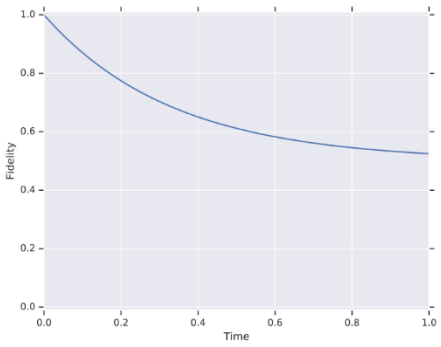
A



B



C



D

Time-Dependent Decoherence

Which of the curves above is the most realistic?

Assume that the decoherence process maps all states to the maximally mixed state, rather than any pure state.

- A
- B
- C
- D

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Compiling Simple Operations

Let's imagine that I have a quantum device with one qubit, called 0, and two instructions, H and T, corresponding to the following matrices:

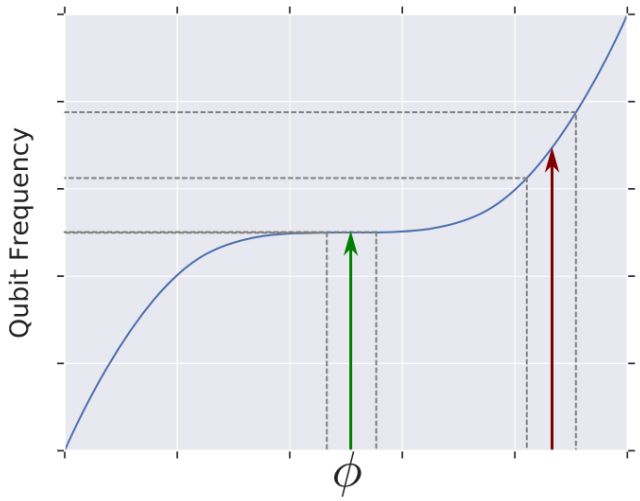
H=12√[111−1] T=[100i√]

Which of the following QASM programs implements X=[0110]?

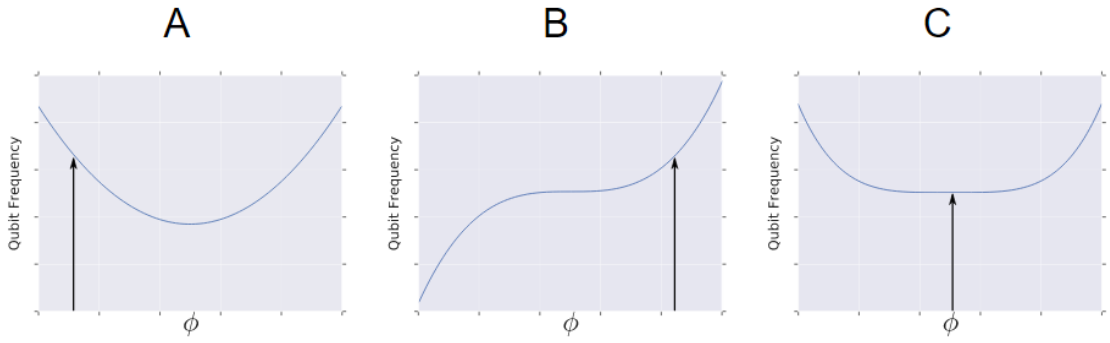
Notation used here is discussed in the 'Ket Notation' section of the Quantum Library.

- H 0; T 0; H 0
- H 0; T 0; T 0; T 0; T 0; H 0
- H 0; T 0; H 0; T 0; H 0

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.In the Section , Menno mentioned 'the sweet spot', which is a simple feature of engineered quantum devices used to decrease the effects of noise in quantum computers. In order to control a qubit, it is sometimes necessary to tune its frequency using an external parameter, which we'll call φ. We'd like the frequency not to vary too much when φ changes slightly, since there can be imprecision in our control of the parameter, or small natural perturbations in its value. A 'sweet spot' is a setting for the parameter where the frequency dependence is flat, so that if the parameter varies slightly, the frequency will not vary.



For example, in the figure above, the setting for φ marked with the green arrow is a sweet spot, since there is little variation in qubit frequency when φ varies. The setting marked with the red arrow is not a sweet spot, since a small variation in φ around that setting produces a large variation in the qubit frequency. Consider the following operating points:



The Sweet Spot

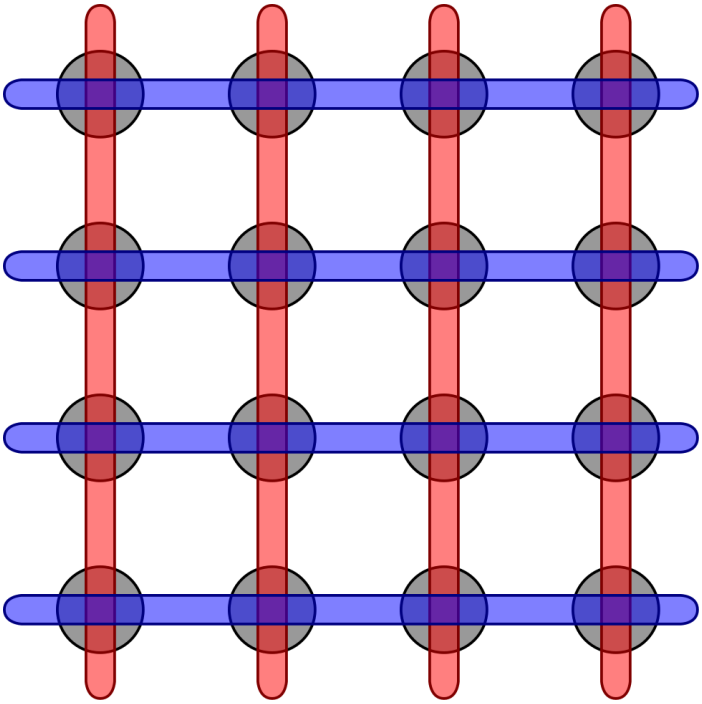
Which of the operating points above is a sweet spot?

- A
- B
- C

Quiz 1: Introducing the architecture, algorithms, and protocols of a quantum computer and quantum internet

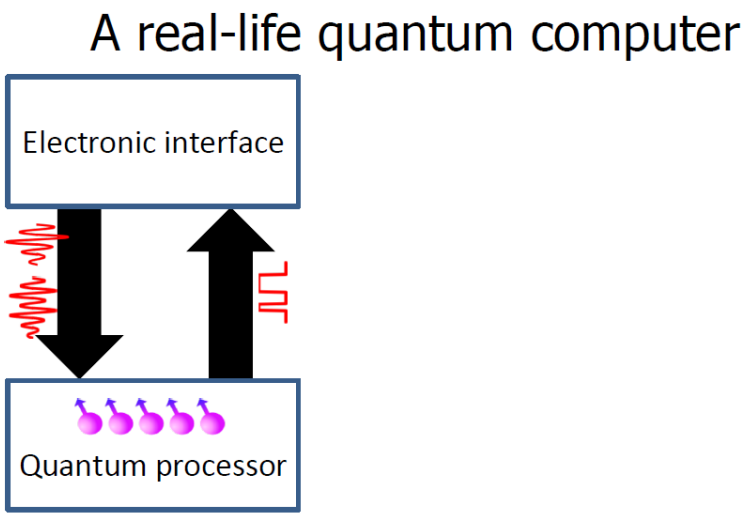
Crossbar-based architecture for quantum computing

In the Section, Menno mentioned crossbars as a way to reduce the number of classical connectors that need to be attached to the quantum chip. See, for example, a simplified diagram of a crossbar-based computer, with 16 qubits and 8 connectors:



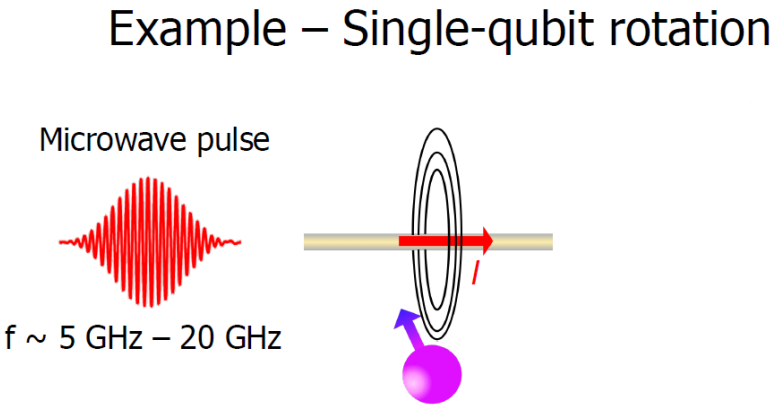
Interfacing qubits with classical systems

In the next two Section lectures, Fabio will talk about the layer that is closest to the qubits: the quantum-to-classical interface. How do we realise the readout of qubits and what are the associated challenges?



Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

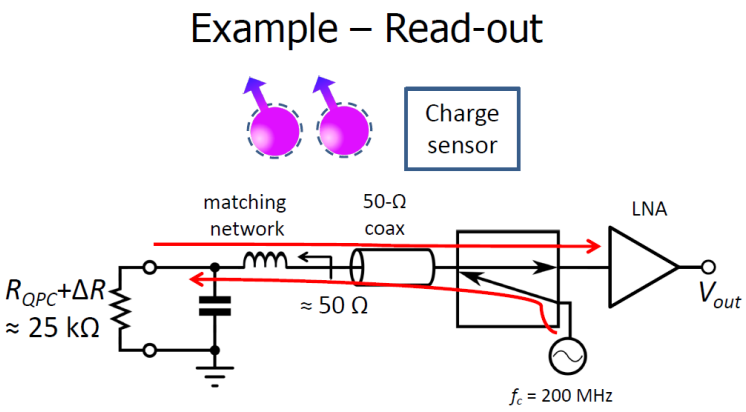
Today I will be talking about how we can interface qubits with classical systems. With classical I mean systems which are non-quantum, so not exploiting quantum effects. In order to understand how we do it, let's look at how a real life quantum computer looks like. Basically, there is the quantum processor where the qubits are and you've seen implementations of qubits for a quantum processor in other Sections. In order to control these qubits, you need an electronic interface to drive the operations on the qubits, for example single-qubit or two-qubit operations. At the same time, you also need to read out the state of the qubits, and this is done also by the electronic interface. In order to be a little bit clearer about what this entails, let's look at a number of examples. For example, let's assume we have a qubit implemented as a spin qubit using a single electron, and you would like to rotate this qubit.



Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

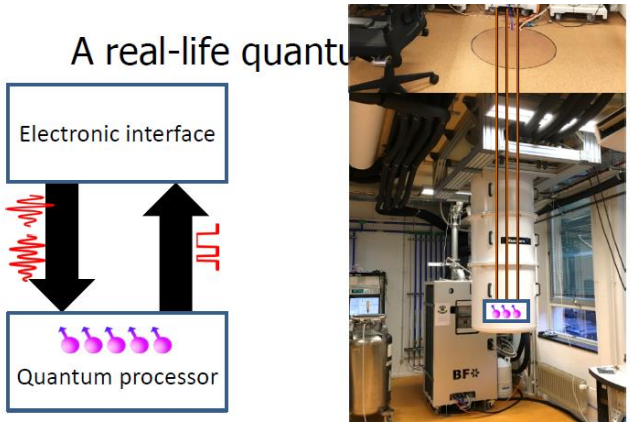
What we need to do is generate a magnetic field which can interact with the magnetic dipole of the electron. To do that, we can think of putting a wire next to the qubit and let a current flow through the wire so that a magnetic field is generated that can interact with the electron. To do that we need to generate a certain microwave pulse with a certain amplitude and duration, at a frequency tuned to the resonance of the electron, so that the qubit can rotate as we want. Now, the amplitude and the shape of this pulse will determine how the qubit exactly rotates. So you can see, by applying a purely electrical signal, we can apply a single qubit rotation in the quantum processor. So that's an example of control. Let's look

now at an example for the read-out. Again, let's talk about a spin qubit using a single electron. This time we want to read the magnetic dipole of a single electron. That is very difficult. So what people do, is that they make sure that the information in the magnetic domain is translated into information about the position of the electrons. So by appropriately timing the quantum processor we can make sure that, depending on the quantum state, the qubit moves to a different position or not. That change in position can be sensed by charge sensors next to the qubit. Now how do we read this charge sensor? Basically, we read the charge sensor by reading its resistance. The main idea is to connect an electrical circuit to such resistor to read its value. Here is a typical example of what is used in a typical setup: you connect the charge sensor to a coaxial line and you inject power at a precise frequency in that coaxial line. If the resistance has a certain defined value, this power will be absorbed by the charge sensor. Otherwise, if the resistance is different, this power will be reflected back and detected by a low noise amplifier. So what we see here is that a quantum phenomenon, the state of the qubit, is completely translated to an electrical signal that can be read out by using classical electrical circuits. So, all of this is embedded in such electronic interface that you see here in the whole system. But how does this look in real life? If you come to one of our labs at QuTech, you will see something like this: we have a big refrigerator, this big cylinder hanging from the ceiling, 2 meter tall. This is used to cool down the quantum processor very close to the absolute zero, because that is the temperature where the qubits work best. The quantum processor is here, at the bottom of the fridge. And to find the electronic interface, we should follow the wires that go through the fridge to the top floor and finally reach the electronic equipment on the higher floor, which implements the electronic interface.

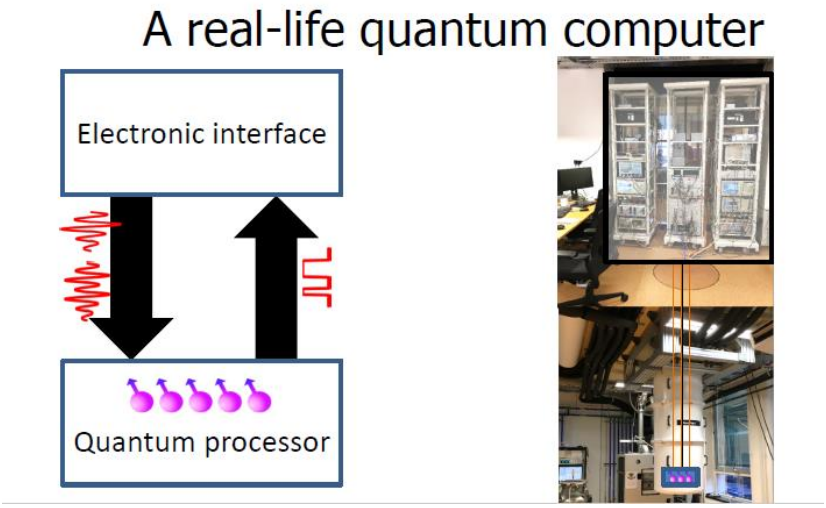


Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

A quantum computer today looks like this: you have the quantum processor at very low temperature and an electronic interface at room temperature composed by very bulky equipment. The question we want to ask ourselves is: can we scale up such a system to a very large number of qubits? Now, you can understand that it is not so simple, especially because of the wiring. If you want to build a quantum computer with practical applications you need millions of qubits and thus millions of wires.

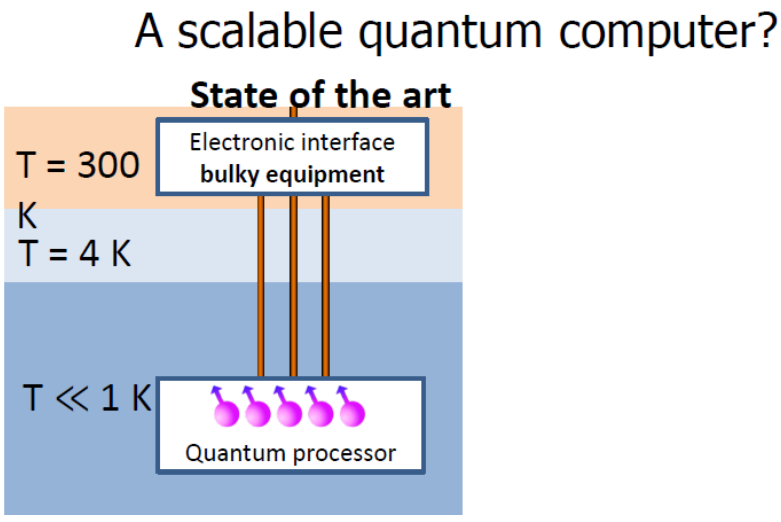


Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

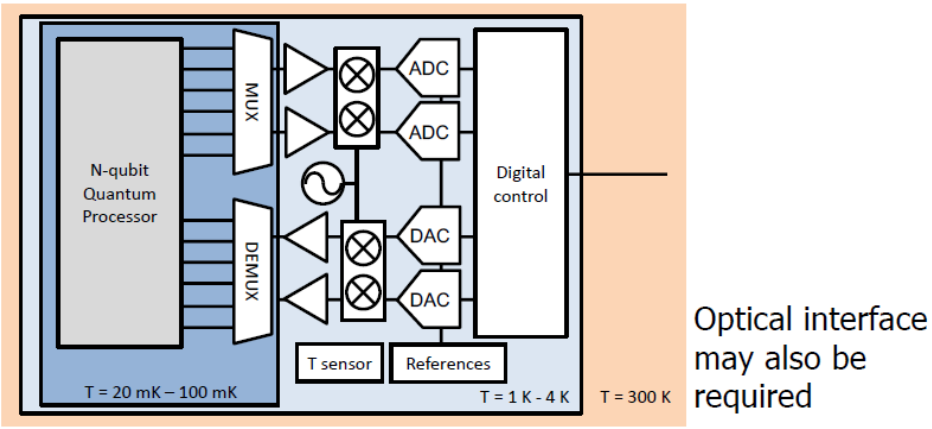


Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

As an analogy, think to the camera in your phone. It has millions of pixels and you try to connect each of those pixels with a wire 2 meters long to its electronic read-out. This is not something we do today, because it is not very practical. The same is true for a quantum computer. This approach here is not practical. So a much better solution is to build electronics on purpose for this application, tailor make it, and bring such electronics also at very low temperature, that we can connect it to very large quantum processor, so that we can build a scalable quantum computer. Our approach is to bring the electronics very close to the quantum processor. Here you see the quantum processor cooled to a temperature between 20 and 100 mK and the electronics placed very close to it. So on the top part of the figure, you see the electronics that we use to read out, so you see some amplification, some frequency down-conversion and then you need to convert this analog signals to the digital domain through analog-to digital-converters, or ADC's. These digital signals are processed by the digital control unit, which determines what to do with this read out states and decides what to feedback into the quantum processor. That's done with the digital-to-analog converters, which brings back the signal into the analog domain, eventually upconverting and amplifying it back into the quantum processor. One thing you may realize is that we may also need to bring some part of the electronics at the same temperature of the qubits, so down to 20 mK. But if we can do that, we can really build a scalable quantum computer.

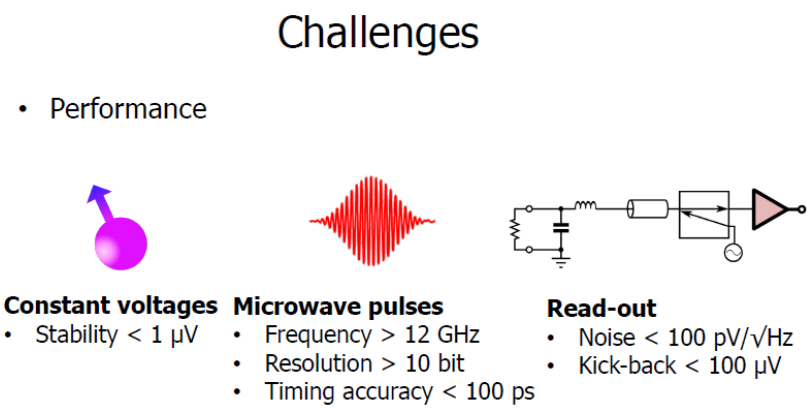


Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems



Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

As a side note: here I am showing a purely electrical interface, but for some qubits, for example NV-centres, we may also need an optical interface, possibly also working at cryogenic temperature. Now, to implement all that, we have to face a number of challenges.

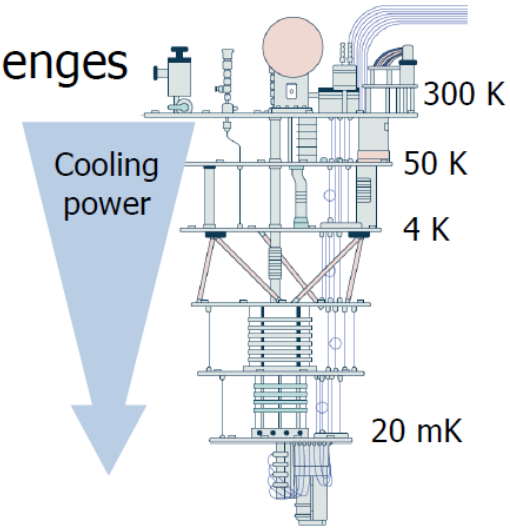


Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

First, all of the electronic controller need to achieve very tight performance. If you think to the example of the spin qubits, just to let the qubit exist, you need to provide constant voltages with a very tight stability well below 1 uV, which means better than 1 ppm. We need also to provide microwave pulses with very tight accuracy in terms of amplitude and timing. And you need to build a read-out with very low noise, much better than what is possible today while at the same time avoiding any kickback. That means avoiding any effect back to the qubit, because you want to avoid spoiling the quantum state. So that's the first challenge: Meeting the performance.

Challenges

- Performance
- Power dissipation
  - ~ 1 W @ 4 K
  - < 1 mW @ 20 mK



Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

The second challenge is that you have to operate this electronics at cryogenic temperature. So typically you want to place it in a refrigerator. You see a simplified diagram on the right. The refrigerator has different stages at different temperatures. The qubits are typically placed at the lowest stage at the lowest temperature, and you want to place the electronics as close as possible. What's the problem? The problem is that the lower you go in temperature, the less cooling power you have available in the fridge. So, for example, at 4K, you only have 1 W of power available, and if we go to 20mK you even have much less than a mW. So the challenge here is to build all these electronics while dissipating the minimum amount of power. Finally, but also very important, we need to build these electronics with a technology that can work at cryogenic temperature as low as 4 K or even 20 mK. You have different options, because you can use superconducting devices or any semiconductor operating at low temperature. Here you see a number of examples. The challenge is to choose the best technology.

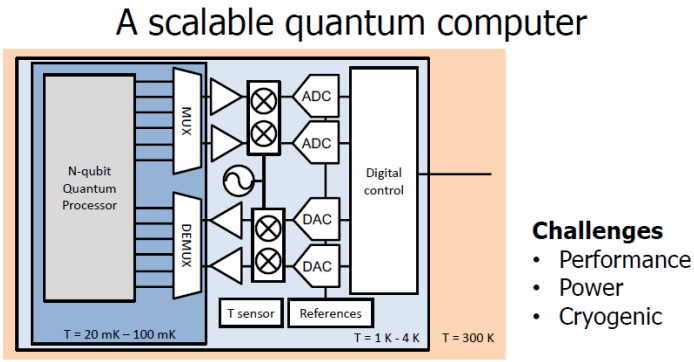
Challenges

- Performance
- Power dissipation
- Cryogenic technology
  - Operate @ 4 K, 20 mK, …
  - Superconducting devices (RSFQ, RQL, SQUID, …)
  - Semiconductors @ low-temperature

	Minimum temperature
Si BJT	100 K
Ge BJT	20 K
SiGe HBT	< 1 K
GaAs MESFET	40 K
CMOS	30 mK or below?

Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

So to conclude this lecture: we have seen that we can build a scalable quantum computer if we can build an electronic interface operating at cryogenic temperatures very close to the quantum processor. However, to make sure that this works, we have to address a number of challenges in terms of performance, power dissipation and cryogenic operation. These are the topics that we are going to discuss in the following Section.



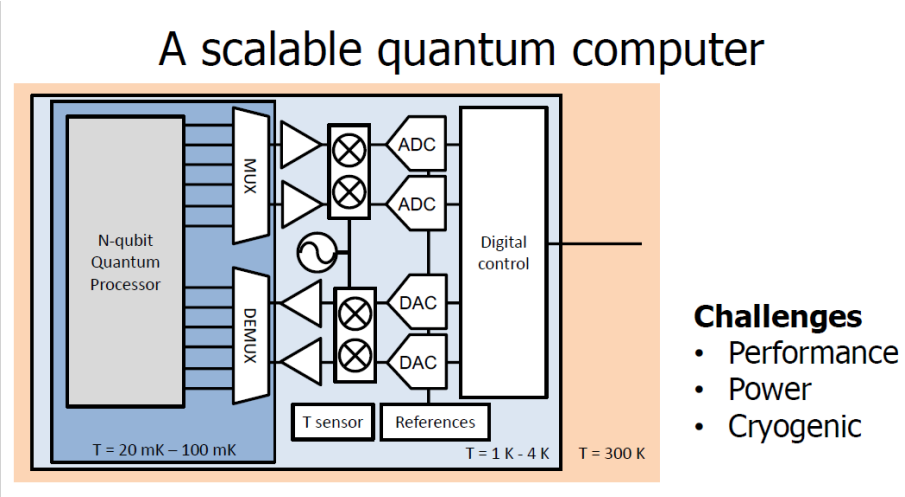
Fabio Sebastiano,Interfacing qubits with classical (non-quantum) systems

Main takeaways

- In order to control the qubits of the quantum processor, we need an electronic interface to perform the operations on the qubits and read out the states of the qubits.
- Custom-built electronic components built to operate at cryogenic temperatures are needed to build a scalable quantum computer.
- The components of the electronic interface need to be placed as close as possible to the quantum processor.
- To make it possible to build a scalable quantum computer we need to take performance, power dissipation, and cryogenic operation into account when building the electronic interface.

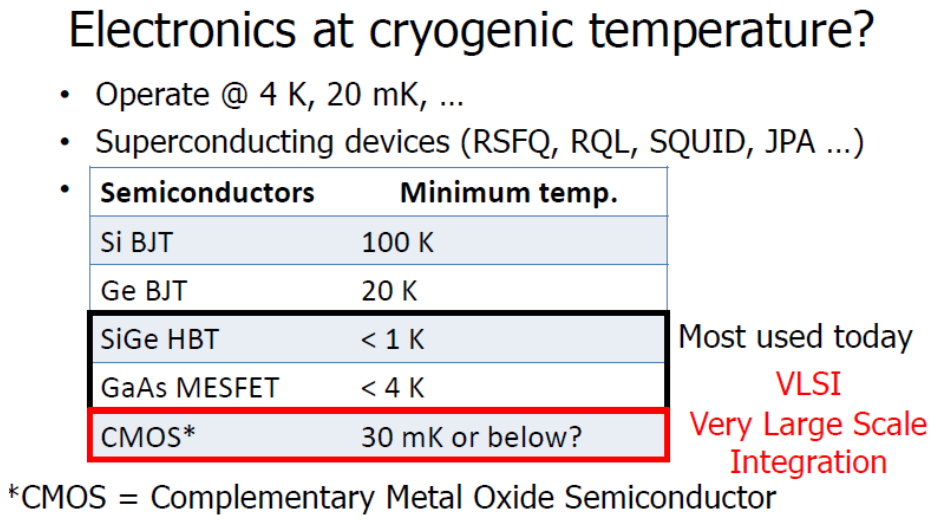
Electronic interfaces for quantum processors

In this second Section, Fabio dives deeper into the challenges that have to be overcome when developing the quantum to classical interface.



Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

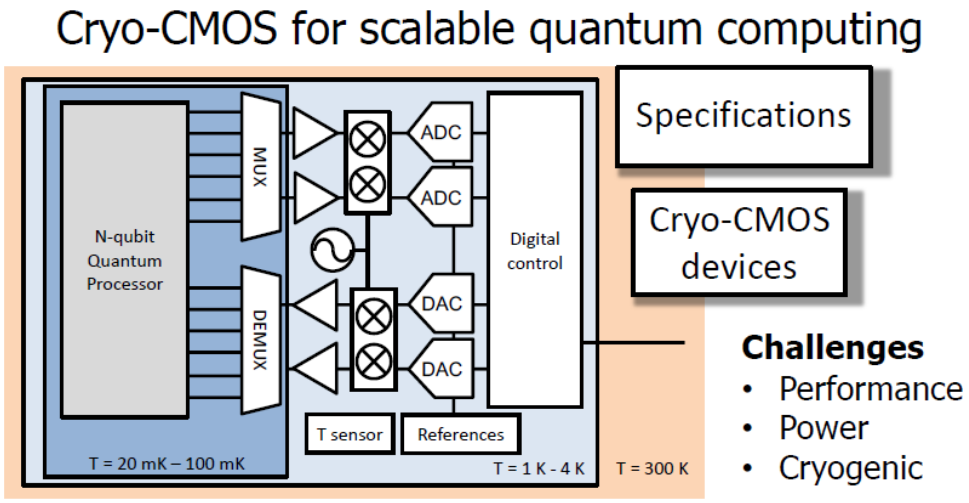
We have seen that to build a scalable quantum computer we need to build a complex electronic interface operating at cryogenic temperature as close as possible to the quantum processor. To do that, we need to meet a number of challenges in terms of performance, power dissipation and of course we must find electronics that can work at cryo temperatures.



Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

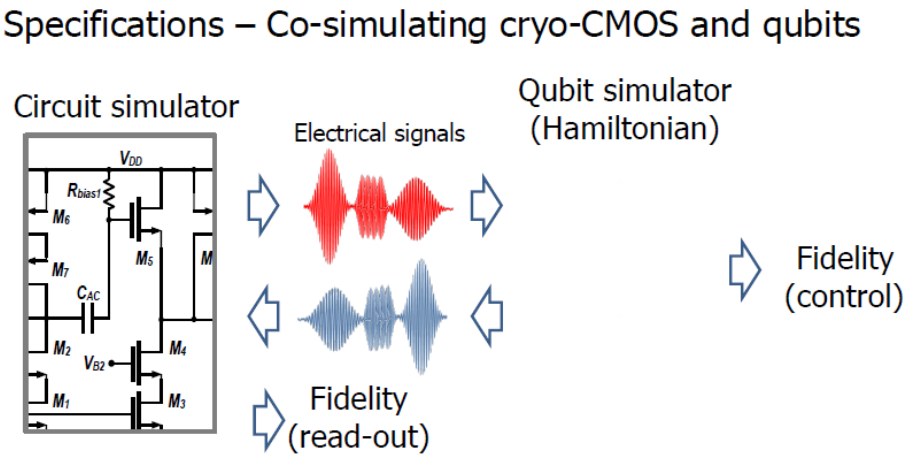
Let's look now to some of these issues in more details. Let's start from the cryogenic operation of the electronics. So which is the best electronic technology that can operate at 4 Kelvin, at 20 milliKelvin or even below? There are different options. First, we can think to use superconducting devices like RFSQ, RQL and so on. These devices operate naturally at very cold temperature because they exploit superconductivity. And one additional advantage is that they have very low power dissipation. Next to that, another option is to take standard semiconductor technologies and look at what is the minimum temperature at which they can operate. One problem is that some of them, like the first line here in this table, cannot operate at the temperature we need, because this device doesn't work below 100K. Other technology really works at 4 Kelvin or even below. For example these two types of devices: Silicon Germanium HBT's or MESFETS are very much used today, also for quantum applications, because they operate at 4 Kelvin or below and can operate with very low power dissipation. However, if we really look closely to all the technologies, there is only one of the technologies here that is very special. That's the last one shown here: the CMOS technology. CMOS stands for Complementary Metal Oxide

Semiconductor and it is the same technology that is used to implement conventional microprocessors. It is the only technology in this list, which can offer Very Large Scale Integration. Which basically means that we can integrate billions of devices on a single silicon chip. But, why is this relevant here? Because we want to build a very large scale quantum computer with millions of qubits and of course also the electronic interface would be very complex.



Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

And CMOS is the only technology today that can offer such complexity, comparable to what is done in conventional microprocessors. So we want to use CMOS, because it also works at 4 Kelvin and is proven to work at least down to 30 mK. In fact, we don't know yet what the minimum temperature is at which the CMOS can still operate. So, if we want to build a scalable quantum computer the best approach, and that is what researchers are doing today also here at QuTech, is to research cryogenic CMOS technology, or cryo CMOS in short. So we want to build all this system here using cryo CMOS.

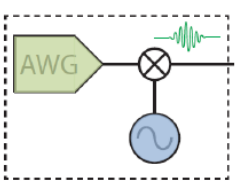


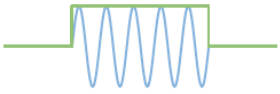
Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

But before we can do that, we have to look at 2 specific problems. First of all, which are the exact specifications that this system must satisfy? And second, how exactly do cryo CMOS device operate at such low temperatures? Let's look at the specifications first. Specifications are very important. Since we have to build a tailor made system, we have to be aware what the impacts are of any choice in the electronics on the qubit performance. What researchers typically do, is to simulate cryo CMOS electronics and qubits together. The main idea is to start from a circuit simulator where you can simulate the electronics. This simulator produces the electrical signals that can be fed to a qubit simulator, which is basically a full physics simulator using the Hamiltonian description of the qubits. From operating the qubits in this way, you can derive what the fidelity of the full quantum computer is, at least for the control part. Next, the qubit simulator

can also produce the electrical signals generated by the qubits and this can be fed to the circuit simulator to emulate or simulate the read-out process and finally get the fidelity for the read-out. By doing this, the designer can really adapt the electrical circuits for the best performance of the full quantum computer. In order to be a little bit more pragmatic, let's look at an example. So, we know that in order to operate a single-qubit rotation you need to produce microwave pulses.

Example – Microwave generation





Electrical specifications

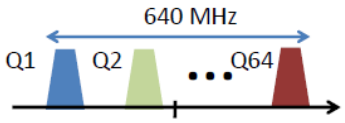
Error Source	Type	Value	Error [ppm]
Nuclear spin	noise	1.9 kHz_{rms}	4
Microwave frequency (nominally 6 GHz)	inaccuracy	11.2 kHz	125
	noise	11.2 kHz_{rms}	125
	wideband noise	12 μV_{rms}	125
Phase	Inaccuracy	0.64°	125
Microwave amplitude (nominally 17 mV)	inaccuracy	14 µV	125
	noise	14 μV_{rms}	125
Microwave duration (nominally 50 ns)	inaccuracy	3.6 ps	125
	noise	3.6 ns_{rms}	125 +
			1000 ppm

(Rabi freq. 1 MHz) ⇒ Fidelity = 99.9%

Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

These can be done as shown here, for example. You have an arbitrary waveform generator, for example this square pulse shown here in green that is multiplied by a sinusoidal carrier coming from the oscillator shown in blue. So the result of the multiplication is a short pulse at very high frequency. This is what you can feed to the qubits to do single-qubit rotations. Now the question is, how precise should this pulse be? With the approach I have shown you before using the simulators, we can analyze any error or non-ideality in this pulse and find out how this relates to the final error in the full quantum computer. In this table we show all the possible source of errors in such pulse such as amplitude errors, frequency errors, noise in the amplitude, noise in the phase and so on and so forth. And we can compute, here at the bottom, at 1000 ppm total error that a pulse, assuming all this error sources, will produce into the quantum computer. This error would result into a 99.9% fidelity for the qubit operation. In this way we can really relate qubits and electronics and find out how good our electronics should be. The follow-up question is: Is it possible to realize a circuit with such specifications, or not? To give a first response, let's look at what we can do using CMOS technology, but at room temperature - not cryogenic. Here I took as an example a practical implementation for the green block and the blue block.

Example – Microwave generation



- **CMOS @ room-temperature**
  - Digital-to-analog converter (10-bit 500 MS/s) P = 24 mW
  - Phase-locked loop (9.2 - 12.7 GHz) P = 13 mW

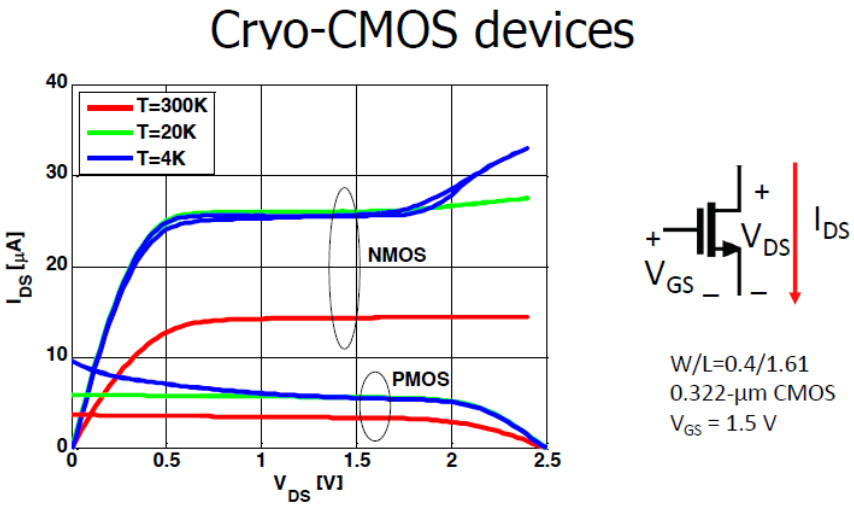
  P_{tot} = 37 mW/qubit
- **With frequency multiplexing?**
  - Digital-to-analog converter (12-bit 1.6 GS/s) P = 40 mW
  - Phase-locked loop (9.2 - 12.7 GHz) P = 13 mW

  P_{tot} < 1 mW/qubit

[Lin, JSSC 2012] [Raczkowski, JSSC 2015] [Lin, JSSC 2014]

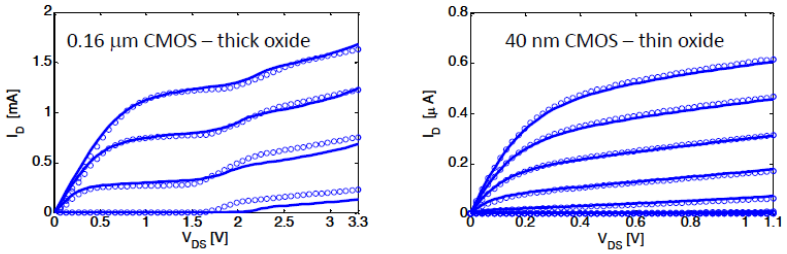
Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

So we can use a digital-to-analog converter to implement the green block and a so-called phase-locked-loop for the oscillator in blue. By choosing specific parts, we can see that such a system would consume a total power of 37 mW for a single qubit. So first of all we can conclude: yes, it is possible to make such electronics, but we consume quite some power. We can imagine that if we have to replicate this for thousands or millions of qubits it is going to dissipate a lot of power. If we have to operate such circuit at cryogenic temperatures, we have a problem because it is not easy to dissipate a lot of power at very low temperatures. But we can be a little bit smarter, and we can do this: We can make sure that this circuit is not only addressing a single qubit, but a number of them. For example, by multiplexing them in frequency. By doing that we can, for example, address up to 64 qubit with the same circuit. Of course, this would require some modifications in our circuit, so you see the number on the bottom changes. As a final result we have a higher total power, but we are addressing more qubits, 64 in this example. So we can get less than 1 mW per qubit. So, what is the main idea here? The challenge is that when we design the electronic interface, we have to look at the system level. We have to co-design the qubit processor and the electronics, in order to make smart choices that makes all the electronics possible, for example in terms of power consumption. But up to now we have seen examples only based on electronics that work at room temperature. Let's look to how those devices would work if you go to cryogenic temperature. Here, I want to show you how a single CMOS transistor works. Basically, a CMOS transistor is nothing else than a device that regulates the current flowing through 2 of its terminals, the so called IDS, as a function of the voltage that you apply on the other terminal of the device. Here on the left you see the current generated a by CMOS transistor at room temperature as a function of the voltage applied to the current terminals. Ideally you would like that this current is as flat as possible and independent from the voltage at the end of the transistor. In this plot, you see an NMOS and a PMOS, the 2 devices present in a CMOS technology. Infact, the C in CMOS stands for complementary technology, meaning that you have 2 different kinds of devices, but that is not very important for the sake of the following discussion. What is more interesting to see is that if you cool down the device, what you see is that the current increases. And that is good, because more current means that the full circuit can go faster. So you have an improvement in performance. But this is the cooling only down to 20 Kelvin. If we really go to 4 Kelvin, what you can start seeing is a very weird behavior, because you see that the current is not anymore flat there is a first strange effect. And you may also see that there is some hysteresis in the current, so the current is different if you sweep the voltage in one direction or the other. And that is very different behavior than what we see at room temperature. Are we able to handle that? In order to understand this phenomena a bit better, we can look at how different transistors in different CMOS technologies behave at 4 Kelvin. In our measurements at 4 Kelvin on the left, you can see exactly the behavior we have just discussed The current is not flat as it happens at room temperature, but it shows this funny kink. On the right, instead, there is a different CMOS technology and here you can see that the curves at 4 Kelvin really behave similar to what you would expect for a transistor at room temperature. So you have these very funny and different behaviors, but the nice thing is that we can predict such behaviors. I can take the standard model used for those devices at room temperature, and extend it to cryogenic temperature.



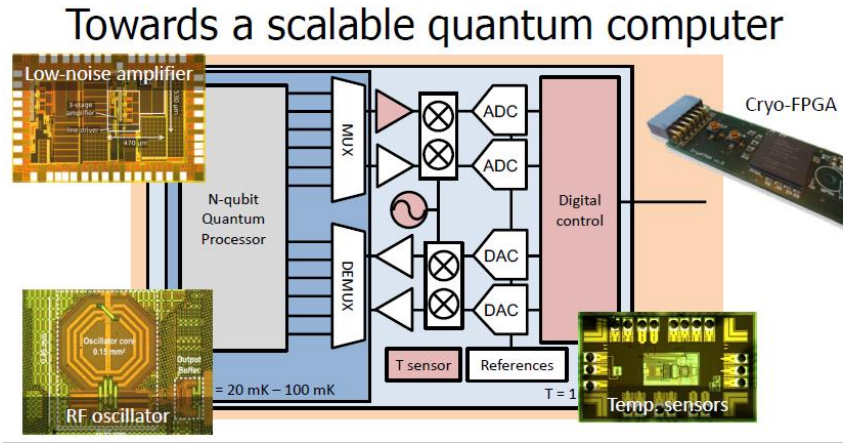
Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

Cryo-CMOS devices



Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

I can show here that this model, shown as the solid lines, fits very well with the measurement data, shown as dots. What is the bottom line here? Basically by using the standard models and the standard techniques, I can model these devices also at cryogenic temperature, so that I can really use them to make electronics that work at cryogenic temperatures. So by using these techniques, I will be able to build such complex systems. Here at QuTech, we have already started implementing a number of blocks that will be the basis for building such a large system. For example, we have investigated how to use standard digital circuit like FPGA's operating at 4K and below, and how to build temperature sensors integrated on silicon, RF oscillators, and low noise amplifiers, all operating at cryogenic temperature. All these blocks are required in the electronic controller for quantum processors. This cryogenic electronic interface will enable us to build the scalable quantum computers of the future



Fabio Sebastiano,Electronic interfaces for quantum processors –the challenges

Main takeaways

- Standard semiconductor technologies and superconducting devices are potential technologies that can be used to build the components for the electronic interface.
- Researchers use simulators to analyse the impact of electronic components on qubit performance.
- The quantum processor and the electronic interface need to be co-designed to get the best result.
- By extending the standard models and techniques, it is possible to predict the behaviour of different CMOS technologies at cryogenic temperature.

Practice Questions

Pulse Amplitude & Duration

One simple scheme to perform unitary operations on qubits is to subject them to a *control Hamiltonian* gH (g being the *amplitude*) for a time t. The unitary operation that results is exp(−iℏHgt).

Suppose that we've already found an amplitude and a time which produce a given unitary U, and we'd like to find a new amplitude/time pair that produce U−−√=U12.

Which of the following pairs will accomplish this?

- [ ] g2, t
- [ ] g, t
- [ ] g2, t2
- [ ] g, t2

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Uncertainty in Pulse Amplitude & Duration

The product of the pulse amplitude g and the time t is important for determining the operation carried out by a control pulse.

Suppose that our pulse amplitude is imprecise, equal to g+Δg for some unknown Δg.

The pulse time, however, is precisely known.

How should we set g and t to minimise uncertainty in the output unitary?

- ( ) The uncertainty in the output unitary is fixed by Δg, so every possible setting performs equivalently.
- ( ) Set g to be as large as possible.
- ( ) Set g to be as small as possible.

Learn more

Additional reading

- A time line of the steps forward in the development of **quantum computing**.
- In the scientific paper **Simulating Physics with Computers** Feynman mentions for the first time the potential of building a quantum computer.
- The scientific paper **Cryo-CMOS Circuits and Systems for Quantum Computing Applications** proposes a scalable solution for system integration at cryogenic temperatures.
- The guide **Principles of dilution refrigeration** gives some background on the refrigerators used for quantum computing. Dilution refrigerators have been used for cooling anything from millimetre-sized semiconductors to 3-ton gravitational wave detectors, down to milliKelvin (mK) temperatures.
- Fabio mentions the phenomenon of hysteresis in the second Section. This wikipedia page gives more information about this.

Module 2: Micro-architecture, compiler & programming language

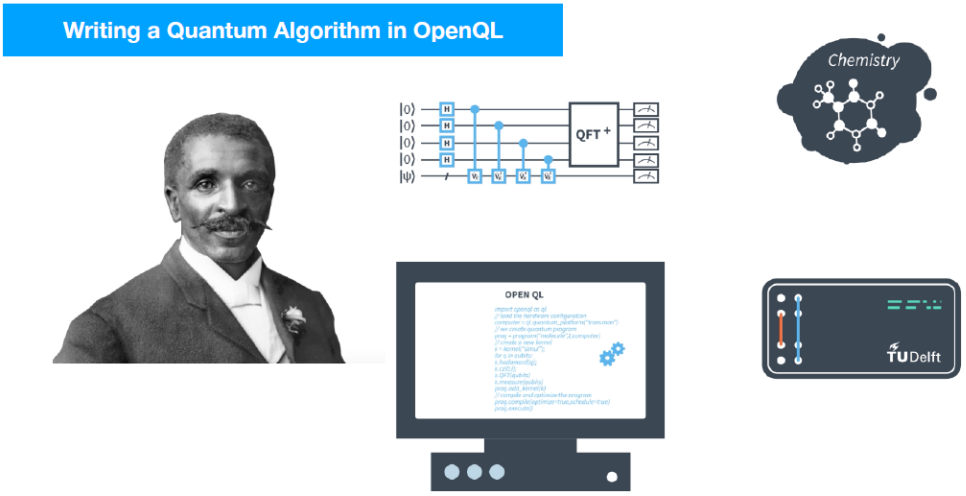
Introduction to Module 2

In the previous module, we refreshed our knowledge of the physical qubit layer, and introduced the Quantum-to-Classical layer which translates between the languages of classical digital instructions and qubit-specific analog waveforms that are required for both control and measurement.

Now we are ready to study this digital language in more detail. Over the course of four lectures and two quizzes, Nader Khammassi, Carmen G. Almudever and Koen Bertels introduce the micro-architecture, low-level programming languages and compilers used in quantum computing. These software layers decompose abstract quantum operations into concrete instructions that can be implemented on a physical device. We will also discuss the classical feedback loop that allows us to choose which quantum operations to execute depending on measurement results from the quantum processor itself. We hope you find this module informative and interesting.

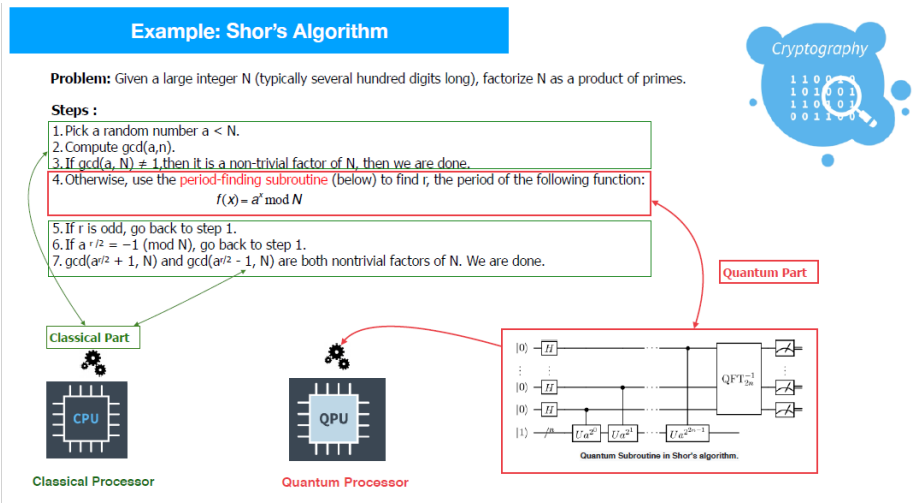
Micro-architecture - part 1

The first part of this module is concerned with the *microarchitecture* we'll use to specify a minimal set of instructions which can be executed in the "quantum accelerator" model. Nader Khammassi will introduce the *arbiter*, which determines whether to execute a given instruction on a classical or quantum processor. This spares the quantum computer from performing tasks which can be carried out more reliably and efficiently on classical hardware.



Nader Khammassi,Micro-architecture

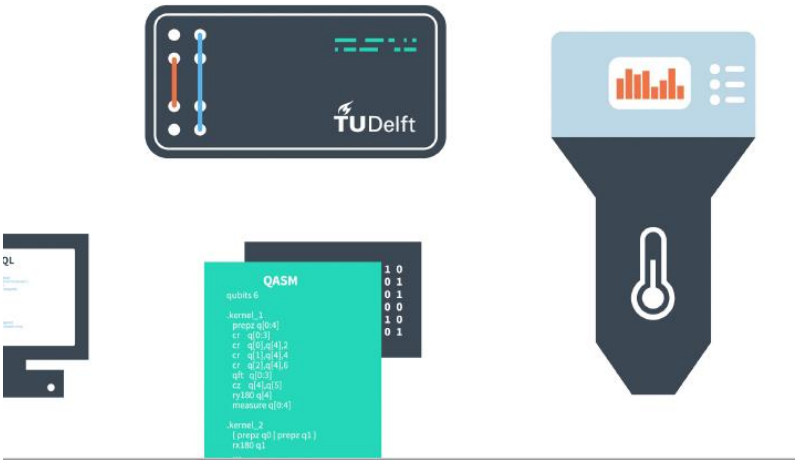
In the previous lectures, we saw that quantum computing can speed up significantly several algorithms from different application domains. Among those applications, for instance in cryptography, Shor's algorithm can accelerate prime number factoring; in chemistry, we can speedup molecule simulation; we can use the Grover algorithm for fast data search; and we can also accelerate pattern matching in genomics. We saw also how quantum algorithms can be expressed in a high-level programming language such as the OpenQL language designed at QuTech. In particular, we saw that those algorithms are composed of a mix of classical and quantum operations.



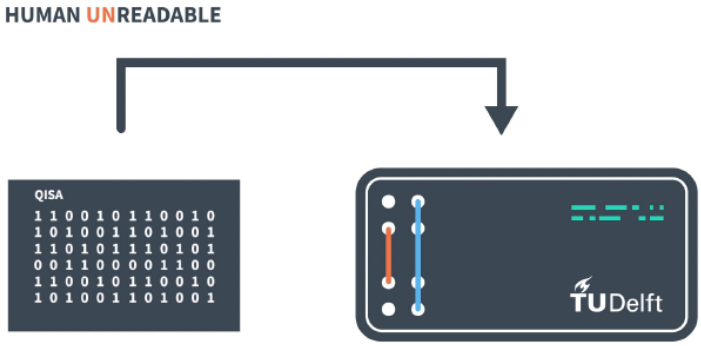
Nader Khammassi,Micro-architecture

The classical part can be executed on a traditional processor and the quantum part can be executed on the quantum processor. For instance, the Shor's algorithm is a famous quantum cryptography application for prime number factoring. This algorithm includes classical computations such as the Greatest Common Divisor computation which can be executed efficiently in a traditional processor, and the quantum part such as the Quantum Fourier Transform which should be executed on a quantum processor and operate on the qubits. As we saw in the compiler lecture, the initial human-readable program implementing the algorithm goes through different compilation steps, which produce a binary code. That binary

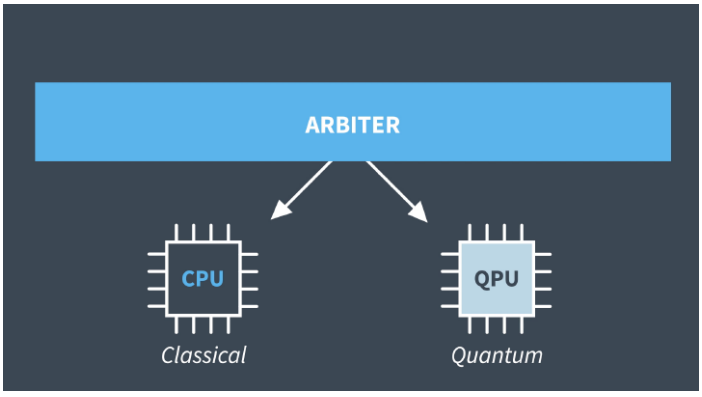
code can be decoded and executed by our quantum computer. The goal of the micro-architecture is to execute that code in a deterministic way and provide all the run-time support required to execute both the quantum operations and the classical operations and then return back the computation result, or the measurements. When loading the executable in our micro-architecture, it is important to distinguish the classical and quantum operations. For that, an “Arbiter” unit is responsible for sorting those instructions to distinguish classical and quantum instructions and route them to the appropriate pipeline: so, the classical instructions to the classical pipeline and the quantum instructions to the quantum pipeline. Those instructions are then decoded and executed either on the host processor or on the quantum processor.



Nader Khammassi,Micro-architecture

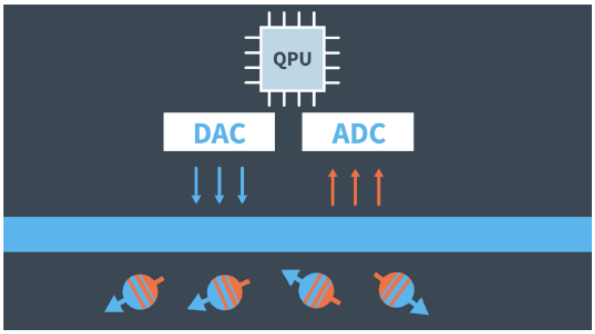


Nader Khammassi,Micro-architecture



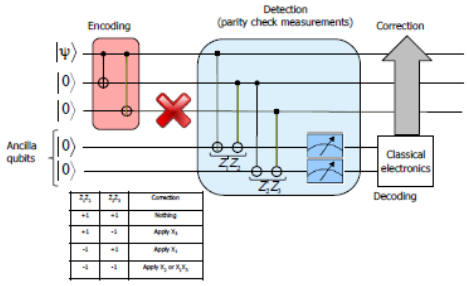
Nader Khammassi,Micro-architecture

Qubit Measurement



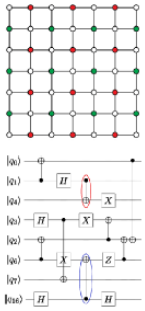
Nader Khammassi,Micro-architecture

Quantum Error Correction (QEC)



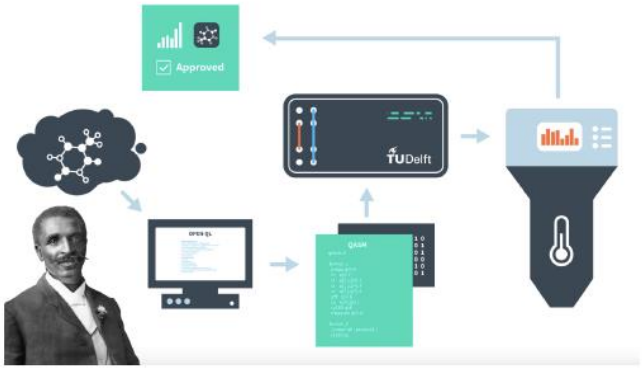
Nader Khammassi,Micro-architecture

Qubit Addressing and Routing



Qubit Symbol	Physical Address
q0	(0,1)
q1	(2,2)
q2	(1,2)
q3	(3,5)

Nader Khammassi,Micro-architecture



Nader Khammassi,Micro-architecture

One of the main tasks of the micro-architecture, or more specifically the quantum pipeline is to translate the quantum instructions into analog signals, which can be sent or applied to the qubits. These signals can be microwave pulses or constant voltage levels. So, each quantum instruction is fetched from the instruction cache, decoded, pushed in the execution queue, and translated to an analog signal using a Digital-to-Analog Converter. The digital quantum instruction needs to be translated into a micro-code instruction and ultimately translated in the right analogue microwave.

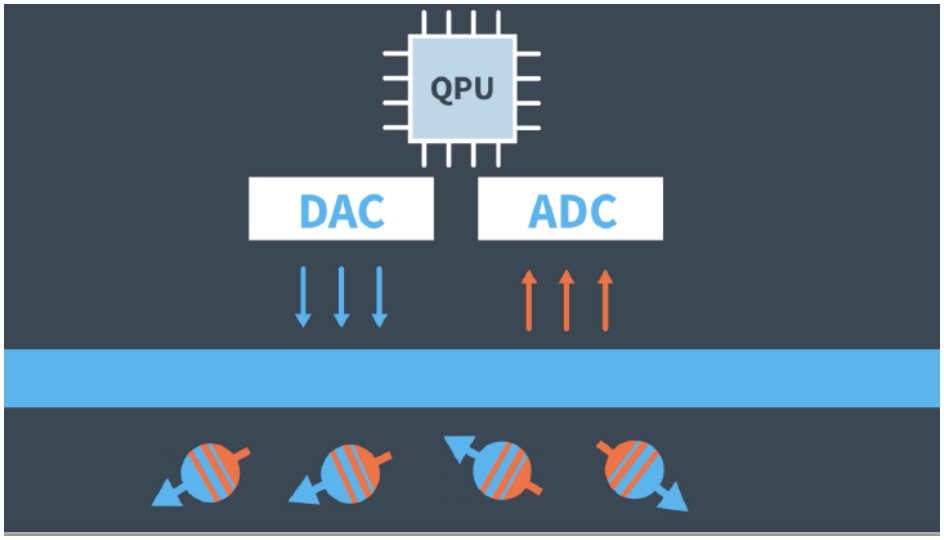
Main takeaway

- Many "quantum" algorithms are actually hybrids, having subroutines which can be carried out on a classical computer. This gives rise to the *arbiter*, a component of the software stack that determines which subroutines can be quickly carried out on a classical co-processor, keeping the quantum computer free for more important tasks.

Micro-architecture - part 2

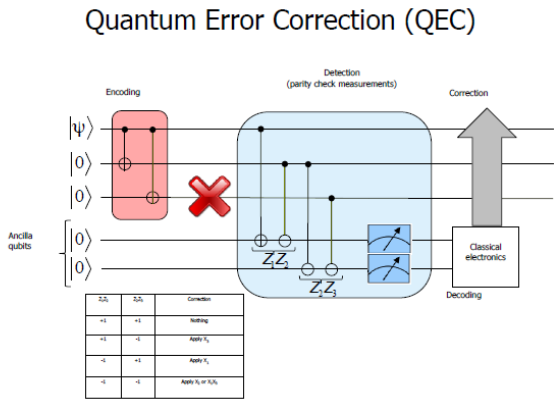
In the previous module, we introduced the "quantum-to-classical" interface as a means of storing quantum instructions on a classical computer, as well as classically interpreting the results of a quantum algorithm. In this lecture, Prof. Koen Bertels will introduce classical feedback, which is an important step in computations where the operation we wish to do next depends on a measurement result obtained *during the computation itself*. This is a necessity in quantum error correction, which we will introduce in the following module.

Prof. Bertels will also introduce the *qubit addressing table,* which is used to keep track of the positions of the various qubit states as they are routed in hardware. This routing is necessary, since a pair of qubits can only undergo an entangling interaction if the qubits are next to each other on the chip.



Koen Bertels,Micro-architecture

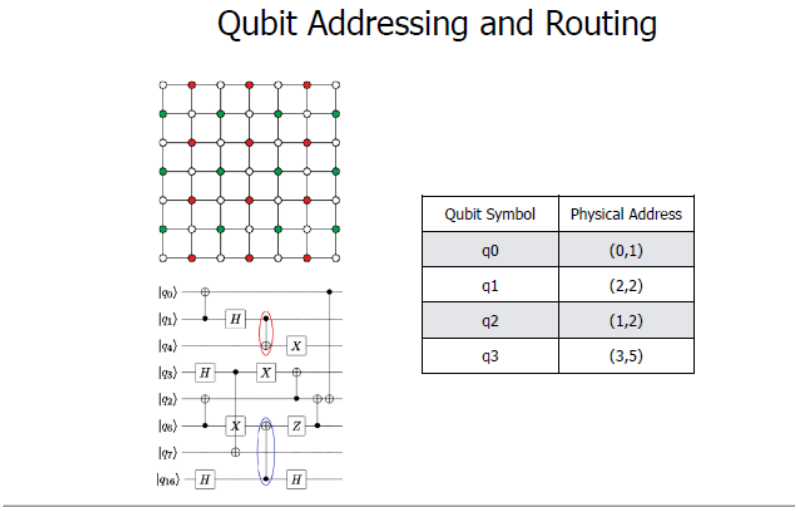
It is also very important to acquire the results of our computation, by means of the qubit measurements. We should not forget that measuring the qubit state will actually destroy the superposition. As we explained before, the amplitude of the states of the qubit represents the probability with which a particular state of the qubit will be measured. This again shows the non-deterministic way of quantum computing. Even though the largest amplitude implies the highest probability for being read, it will happen that several runs of the same algorithms are needed. And each time at the end of the algorithm, the measurement can be done and when done multiple times, the result, which has been read out, most of the times contains the result. For doing a measurement, we need to measure the qubit and convert the analog signal we receive back from the qubit using an Analog-to-Digital converter this time.



Koen Bertels,Micro-architecture

And then we apply a digital signal processing algorithm to discriminate the outcome of measurement – whether it is a 0 or a 1. The measurement outcome can be a final result or an intermediate result, which can be used to control further

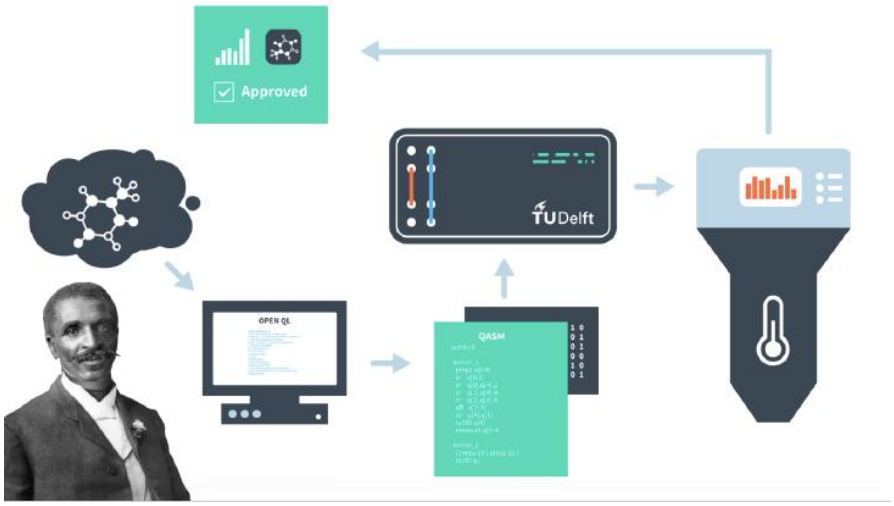
operations. Closely related to this measurement operation is quantum error correction and detection. We need to keep in mind that qubits are fragile and error-prone and due to their very limited coherence time, they are exposed to errors.



Qubit Symbol	Physical Address
q0	(0,1)
q1	(2,2)
q2	(1,2)
q3	(3,5)

Koen Bertels,Micro-architecture

Therefore, quantum error correction is a critical component of the micro-architecture; and a dedicated unit is in charge of triggering periodic error-correction cycles and correcting errors whenever they happen. Such that they offer the required fault-tolerance and provide correct results. Once the measurements of the ancilla qubits are received, a decoder will interpret the readouts and determine whether or not an error has manifested itself for which some correction is necessary. Quantum Error Correction is a very important functionality of the micro-architecture as certain estimates show that up to 90% of what a quantum processor is doing is related to that Quantum Error Correction activity. Another important component of the micro-architecture is the qubit addressing table, which allows us to keep track of qubits and their locations. The address and location of these qubits are basically used to update the physical address of each qubit used in our algorithm, whenever it is moved from one location to the other. This addressing table is crucial for accommodating the routing mechanisms, which move qubits around to enable two-qubit operations which require the qubits nearest neighbour to be as close as possible. Finally when operating all these units together, we can execute quantum algorithms and enjoy the opportunities that quantum computing has to offer in terms of computing speed-up.



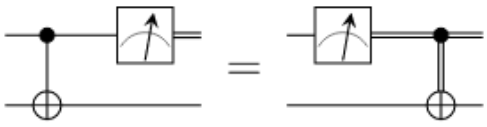
Koen Bertels,Micro-architecture

Main takeaways

- Certain quantum algorithms require different operations to be applied, depending on measurement results which are acquired during computation. This includes all known quantum error correction routines, which makes the feedback loop from quantum measurement data to new quantum instructions a crucial component of the stack.
- In order to execute algorithms on computers where not all pairs of qubits can be entangled with each other *natively* (using a single instruction), the microarchitecture is also able to re-assign qubit states to different addresses in hardware, thanks to the inclusion of a *qubit addressing table*.

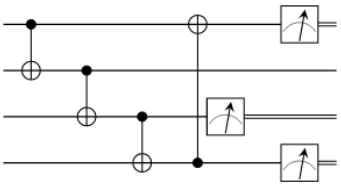
Practice Questions

One of the most important things that a classical arbiter does is to off-load computation from a quantum computer to a classical computer where possible, due to the relative ease of classical computing. One simple way in which quantum computations can be off-loaded onto a classical co-processor is based on the realisation that a CNOT followed by a measurement on the control qubit is equivalent to first measuring the qubit, then performing the *X*, or NOT operation conditioned on the measurement result, as seen below:



Note that the double line carries classical information.

Consider the following circuit:



Classically-Controlled Operations

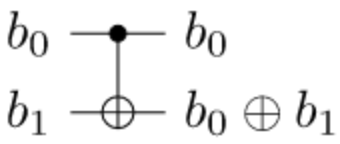
How many of the CNOT gates in the figure above can be eliminated using classical control?

- 0
- 1
- 2
- 3

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

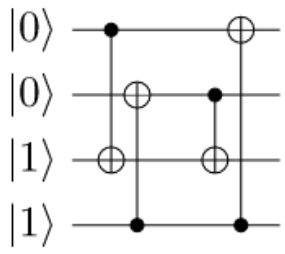
Another way to off-load parts of a quantum computation onto a classical processor is to 'absorb' CNOT gates that occur immediately after state preparation into the preparation itself.

This technique relies on the fact that the CNOT gate has the following action on classical bits:



Here, b0 and b1 are classical bits, with value 0 or 1, and the operation ⊕ performs summation modulo 2, so 1⊕1=0.

Consider the following circuit:



Classical State Preparation

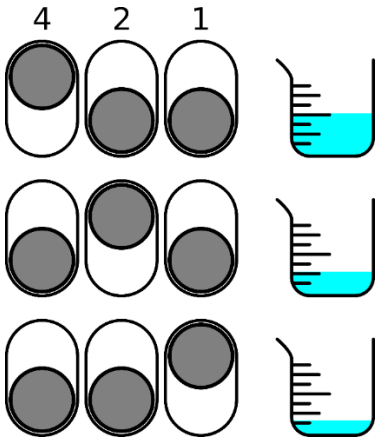
Which state should we prepare in order to eliminate the CNOT gates in the figure above?

- |1011⟩
- |0110⟩
- |1101⟩
- |0010⟩

Quiz 3: Micro-architecture

Digital-to-Analog conversion

In order to discuss the bottom layers of the stack in greater detail, it's necessary to understand *digital-to-analog converters* (DACs), which are used to produce continuous outputs in response to discrete inputs. To understand what DACs do, we'll imagine a simplified DAC that is controlled by three digital switches, and fills a beaker according to which switches are turned on:



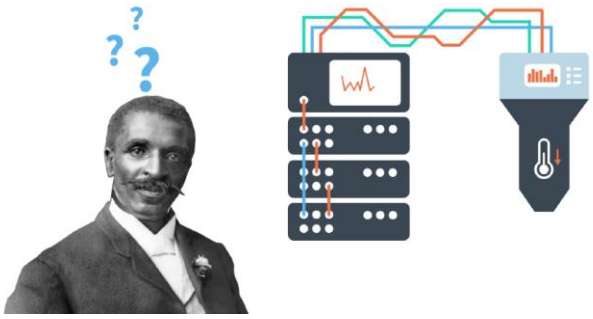
The left switch adds four units of water to the beaker, the middle switch adds two units, and the right switch adds one unit. The switches are independent, so if two of them are switched on, both amounts of water which are controlled by those switches will be added. For example, to fill the beaker with seven units, we turn all of the switches on, and the beaker is filled with 4 + 2 + 1 = 7 units of water.

Compiler & programming languages

People who design quantum algorithms cannot account for the specific details of a given quantum device. These details include the size of the device, as well as connectivity (which sets of qubits can be entangled with a single operation). In order to transform generic quantum algorithms into a form which can be executed on a device of our choosing, we'll need a *compiler*, a classical computer program which finds a near-optimal fit between the desired quantum operations and the limits of near-term devices. These compilers, in turn, require us to express quantum algorithms in specific *quantum programming languages*. In these two lectures, Nader Khammassi and Carmen G. Almudever will introduce compilers and programming languages.

Nader Khammassi,*Quantum Compilation Introduction*

As we saw in the previous lectures, since the early formulation of the principles of quantum computing, many quantum algorithms have been designed and programmed to provide a significant speed-up compared to their classical counterparts.

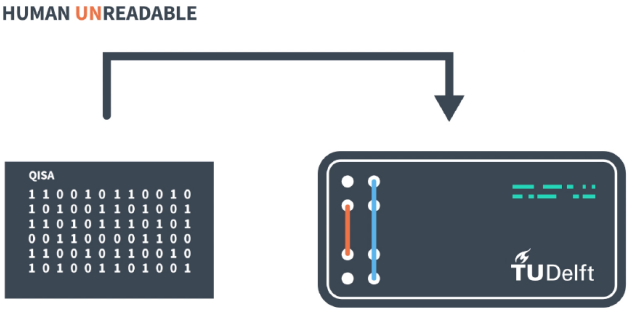


Nader Khammassi,*Quantum Compilation Introduction*

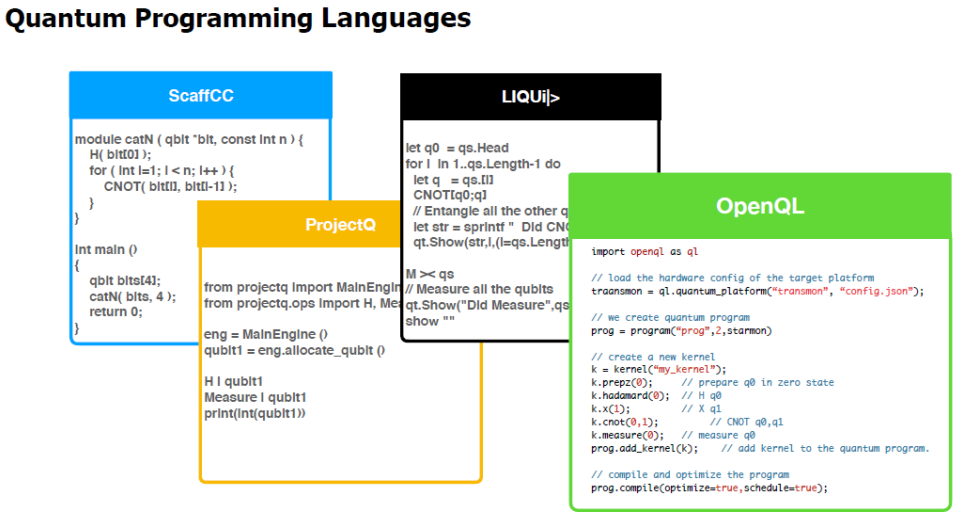
Many algorithms such as Shor's algorithm or Grover's algorithm have many applications in security and data search, machine learning, and designing new materials and developing new drugs. Current quantum computing experimental platforms are complex and hard to operate. Therefore, when designing new algorithms, one of the main challenges in the field of quantum computing is the implementation of those algorithms and their execution on a real quantum computer.

Nader Khammassi,*Quantum Compilation Introduction*

Or answering simply the question: “how to program a quantum computer?”. Programming a quantum computer requires the quantum computer to be programmable in the first place. And that means that it should have a micro-architecture capable of executing instructions. So, the user can write his algorithm using a high-level language such as the OpenQL language, compile it to transform the human readable code into executable code, and execute the produced binary code on the quantum computer hardware. To achieve that goal, a lot of work has been done at QuTech to design the micro-architecture, which accepts a well-defined set of instructions and execute them to perform the quantum operation on the qubits and acquiring back the results of that computation.

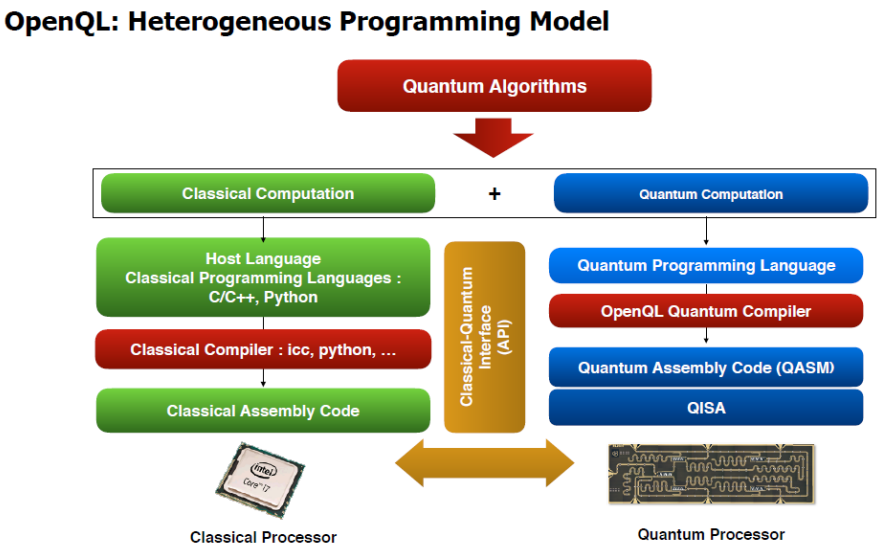


Nader Khammassi,*Quantum Compilation Introduction*



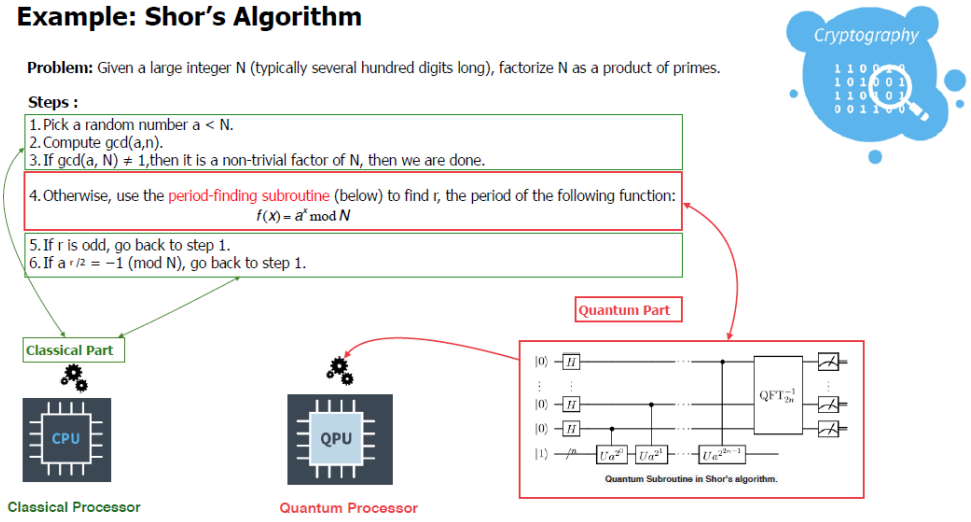
Nader Khammassi,*Quantum Compilation Introduction*

The microarchitecture is responsible of executing both the quantum operations and the classical ones which compose usually a quantum algorithm. The quantum instructions are ultimately translated into microwave pulses which are sent to the qubits through a Digital-to-Analog Converter, and measurements acquisition through an Analog-to-Digital Converter. We will present the micro-architecture details in the dedicated lecture, but for now let's assume that we have such a programmable architecture. To allow the programmer to express his algorithm easily, we need a high-level programming language which abstracts away the low-level details of the machine and offers a friendly programming interface for the user to write human-readable code and to express those algorithms. Many research labs propose different quantum programming languages and frameworks such as ScaffCC from the University of Chicago, or ProjectQ from ETH Zürich or LIQUI|> from Microsoft. Here at QuTech we have designed the OpenQL framework, which allows the programmer to write his algorithm including the quantum and the classical part. Like OpenCL, OpenQL follows a heterogeneous programming model, where the quantum computer is considered as an accelerator, and the general-purpose processor is used as the host processor. OpenQL supports two high-level languages for now, which are C++ and Python. Let's now consider the Shor's factoring algorithm, which is an example of a quantum algorithm which allows finding the prime factors of a large number and offers a significant speed-up compared to classical implementations. As we can see, the Shor's algorithm is a



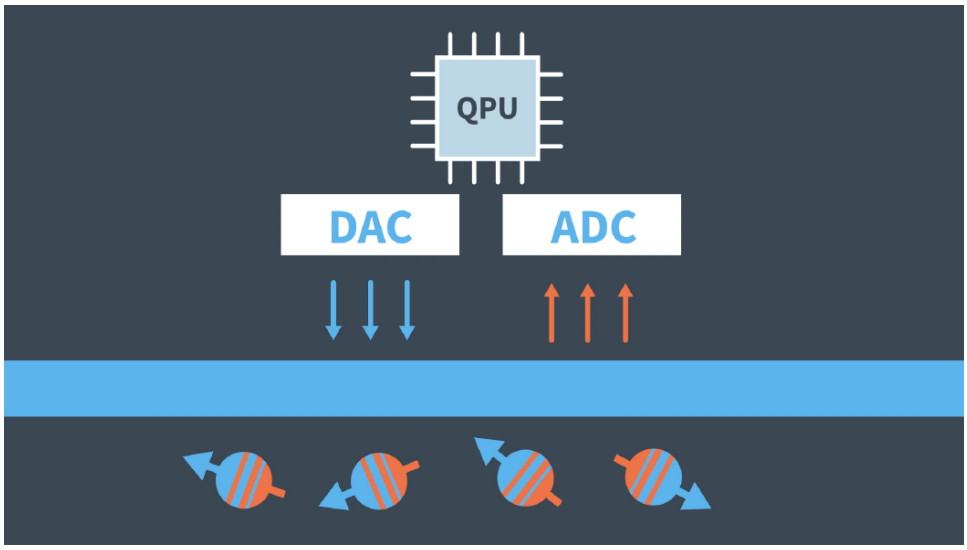
Nader Khammassi,*Quantum Compilation Introduction*

mix of classical instructions and quantum instructions. For instance, computing the greatest common divisor of two numbers can be done efficiently on a classical processor, while Quantum Fourier Transform (QFT) and other quantum operations can be done by the quantum processors.



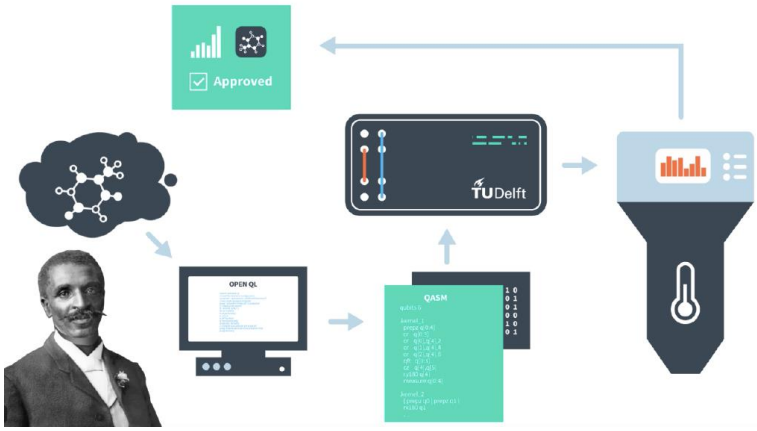
Nader Khammassi,*Quantum Compilation Introduction*

To transform these human-readable quantum algorithms into an executable machine code, we need a compiler. The job of the compiler is bridging the gap between the quantum algorithm, or the software, and the quantum hardware by transforming these algorithms into quantum operations, optimizing them and adapting them to the target quantum platform.



Nader Khammassi,*Quantum Compilation Introduction*

The compiler transforms a human-readable code into a quantum assembly code, and then into a binary executable code. In the next part, we'll see how this compiler is designed and what are the different stages of transformation, which are applied to the program to make it executable on real hardware.



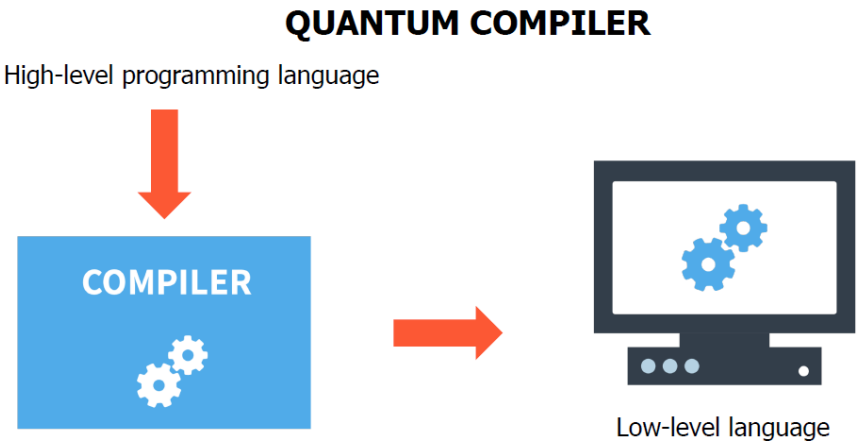
Nader Khammassi,*Quantum Compilation Introduction*

Main takeaways

- A quantum compiler translates expressions of quantum algorithms which are abstract and ignore details of the hardware into a more concrete form which accounts for the specific hardware to which the compiler is associated.
- An important limitation of near-term quantum hardware is connectivity: each qubit in a near-term device is only capable of undergoing an entangling operation with a few of its close neighbours. Every "long-range" operation must, therefore be decomposed into a sequence of nearest-neighbour operations.
- To schedule quantum algorithms, we express them using *dependency graphs*: networks whose sites represent operations, with two sites connected with an arrow from site A to site B whenever operation B requires operation A to be finished before operation B can begin.
- In order to execute quantum algorithms on hardware with limited connectivity, it is necessary to use SWAP operations to transfer qubit states from one location to another.

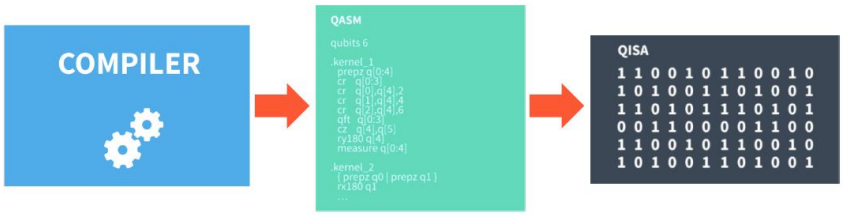
Circuit mapping, routing & scheduling

A quantum circuit must be translated into a form which can be executed on a quantum computer with limited size and connectivity. If there are multiple ways for a given circuit to be mapped onto a quantum chip, which one should we select? In this lecture, Carmen G. Almudever will detail the necessary steps in this selection; decomposition, optimisation, placement, routing, fault-tolerant synthesis and scheduling.



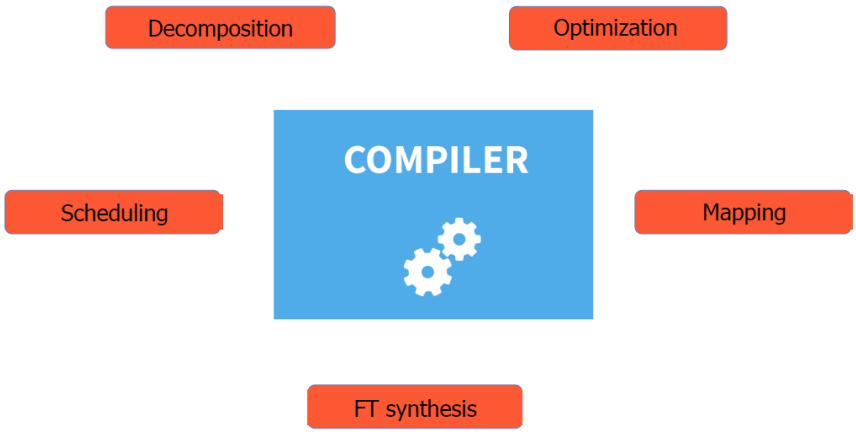
Carmen G. Almudever,Quantum Compiler

As we saw in the previous lectures, a quantum algorithm can be expressed by using a high-level programming language. However, this is still a human talking to a machine, and our quantum processor doesn't understand such a language.



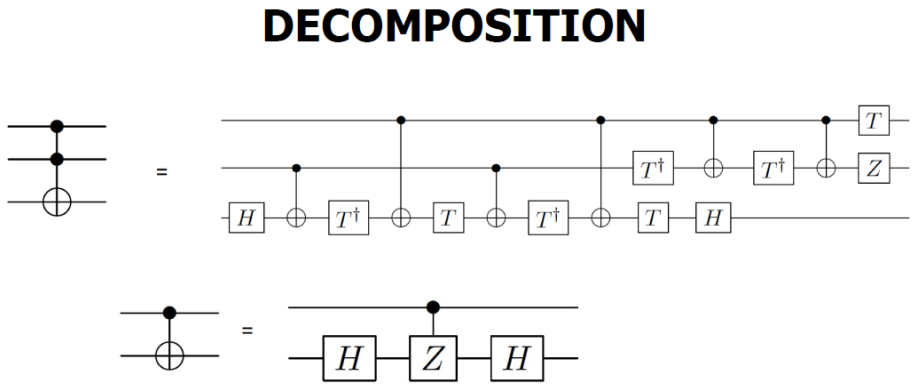
Carmen G. Almudever,Quantum Compiler

That is why we need a compiler that translates an algorithm written in a high-level language into a lower-level language or machine instructions that can be executed on a particular hardware, in this case: our quantum processor. Usually, a quantum compiler generates what is called Quantum Assembly Language, or QASM, that is optimally converted into a binary executable code. However, the mission of the compiler is not only to produce this low-level code, but also to manipulate the quantum circuit to get a version that is "runnable" on the quantum chip. By "runnable", I mean a simplified version that takes into account the details of the chip such as what operations it supports or what constraints must be respected. For this purpose, the compiler will perform several tasks. It will decompose the quantum gates. It will optimize the quantum circuit in order to reduce for instance the number of qubits or the number of operations. It will schedule the operations to maximize the parallelism of the algorithm. It will map the quantum circuit such that the constraints of the chip are satisfied. And it will create a fault tolerant version of the circuit.



Carmen G. Almudever,Quantum Compiler

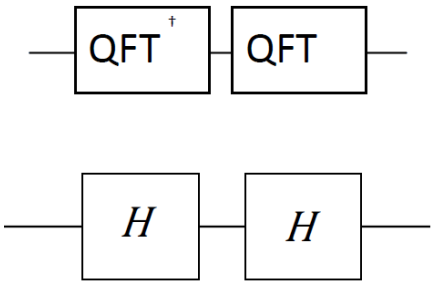
So, let's explain all of these steps by showing some examples. As mentioned, one of the tasks of the compiler is the composition. Any unitary gate single qubits, or multi-qubit gate needs to be decomposed into a reduced and discrete set of gates that are universal. One of the most popular fault tolerant universal set is called Clifford-T that includes, for example, the Hadamard, CNOT, S and T gates. Here, you can see an example in which a Toffoli gate is decomposed in a circuit that consist of Hadamard, T, T-dagger and CNOT. However, this circuit cannot be run yet on a quantum chip. We still need to go one step further and express such a circuit in a series of elementary or physical operations that our hardware can support. For instance, a CNOT is decomposed in into two Hadamard (gates) and a C-phase gate. The compiler can also perform some optimizations of the circuit.



Carmen G. Almudever,Quantum Compiler

As I just mentioned, circuits can be optimized in order to reduce the number of qubitsrequired, also known as the width of the circuit, or to reduce the depth of the circuit or number of gates. These optimizations can be performed at different levels.

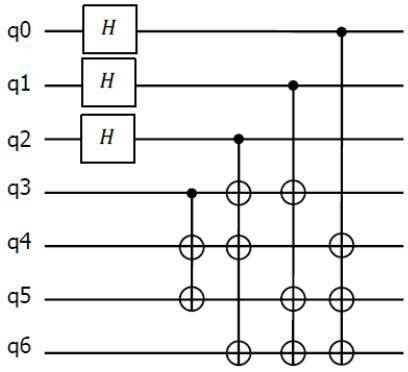
OPTIMIZATION



Carmen G. Almudever,Quantum Compiler

It can be done at the high-level description of your circuit, in which for instance it identifies the QFT and its inverse and it cancels them out, or it can be also be done at a more elementary level after the gate decomposition in which two Hadamards cancel each other. Another function of the compiler is the scheduling of the operations, and this is crucial when the target platform is your quantum chip. Minimizing the circuit latency or execution time of the circuit helps to reduce the potential of having errors during computation. In order to schedule the quantum operations, a quantum instructions dependency graph is built. In this graph, the circles represent instructions or operations and the edges represent the dependency between them. Based on this graph, different scheduling techniques can be used, such as 'As Soon As Possible' or 'As Late As Possible'.

SCHEDULING OF OPERATIONS



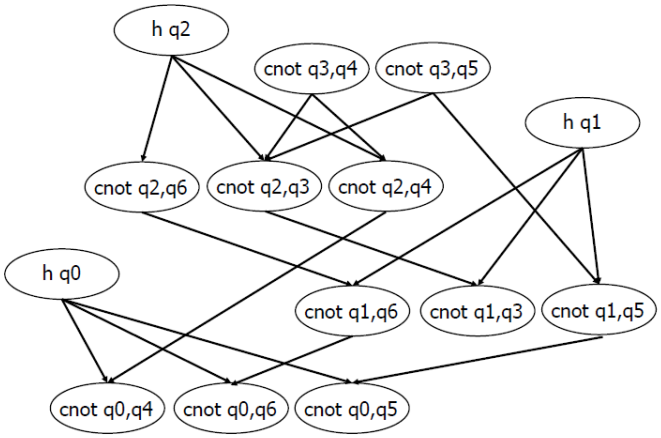

```
h q0
h q1
h q2
cnot q3,q4
cnot q3,q5
cnot q2,q3
cnot q2,q4
cnot q2,q6
cnot q1,q3
cnot q1,q5
cnot q1,q6
cnot q0,q4
cnot q0,q5
cnot q0,q6
```

Carmen G. Almudever,Quantum Compiler

In our example, the operations have been scheduled using an 'As Soon As Possible' technique, and this is the resulting circuit and the corresponding instructions. In our case, the circles are required for the execution of the circuit. Another part of our compiler is the mapping of quantum circuits. In order to run an algorithm on a quantum chip, the corresponding circuit needs to mapped onto it in a way such that the constraints are satisfied. One of the main constraints in experimental platforms is the limited connectivity between the qubits that reduces the interaction between them, for instance, to only the nearest neighbour. In other words, only the qubits that are neighbouring each other can perform a 2-qubit gate. Here, you can see an example of a simplified layout of a 7-qubit chip. Each circle represents a qubit and each arrow (represents) the possible interaction between them. Now the question is: where are the qubits in my circuit that we call virtual qubits placed in our quantum chip? Or in other words, what virtual qubits correspond to what physical qubits. This brings us to first step of the mapping process, that is the placement of qubits. For instance, we can map the virtual qubits to the

SCHEDULING OF OPERATIONS

```
h q0
h q1
h q2
cnot q3,q4
cnot q3,q5
cnot q2,q3
cnot q2,q4
cnot q2,q6
cnot q1,q3
cnot q1,q5
cnot q1,q6
cnot q0,q4
cnot q0,q5
cnot q0,q6
```

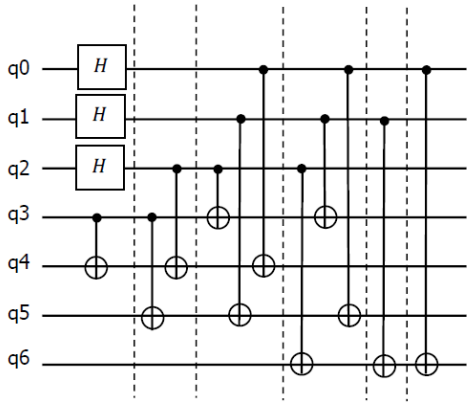


Carmen G. Almudever,Quantum Compiler

physical ones in this way. One can try to place a qubit in a way such that all connectivity constraints are satisfied and all the interactions between them are possible. However, it is not always possible to find such a placement in which, again, all the constraints are satisfied from the beginning. This means that non-neighbouring qubits, or qubits that are not connected to each other, need to be moved or routed to be placed in adjacent positions in order to perform two-qubit gates. This movement or routing of qubits requires extra operations that need to be inserted into the circuit as I will explain in the next example.

SCHEDULING OF OPERATIONS

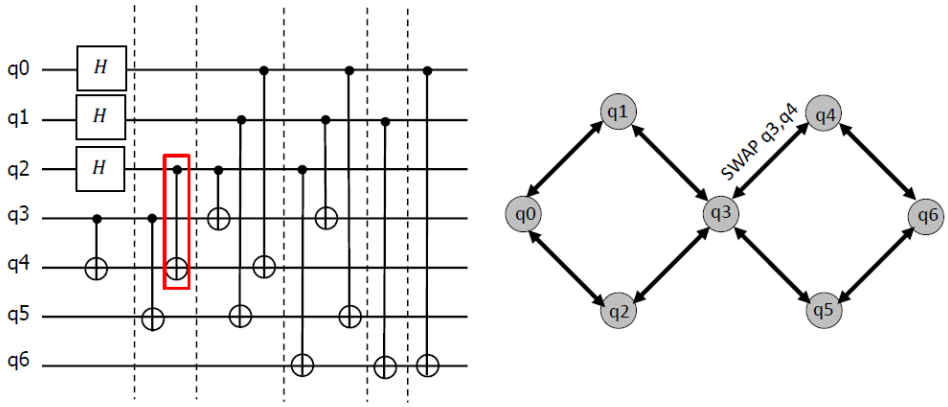
```
C1: h q0, h q1, h q2, cnot q3,q4
C2:cnot q3,q5, cnot q2,q4
C3:cnot q2,q3, cnot q1,q5, cnot q0,q4
C4:cnot q2,q6, cnot q1,q3, cnot q0,q5
C5:cnot q1,q6
C6: cnot q0,q6
```



Carmen G. Almudever,Quantum Compiler

Just to mention that different operations can be used for doing this kind of movement of qubits, or quantum states. For example, SWAP operations, in which the quantum states are exchanged, like the ones used in superconducting qubits, or the Shuttling used in spin qubits. If we look at an example if we are going to perform a CNOT gate between qubit 3 and qubit 4. That is possible, as there is a direct connection between them. If we move to the next CNOT, and nowe we are going to perform a CNOT between 3 and 5. Again, this is also possible as these qubits are neighbours. However, if now we want to perform a CNOT between qubit 2 and 4, they are not neighbours. They need to move to adjacent positions. This can be done by just swapping qubit 3 and qubit 4. And now, as you can see, qubit 4 and 2 and interact and perform such a CNOT. It is clear from this example that the routing or movement of qubits produces an overhead in terms of not only the number of operations, but also in terms of the execution time of the circuit. That communication overhead will affect the reliability of your quantum algorithm. Therefore, an efficient mapping in which highly interacting qubits are placed

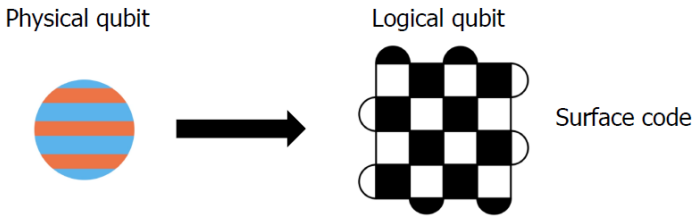
MAPPING OF QUANTUM CIRCUITS



Carmen G. Almudever,Quantum Compiler

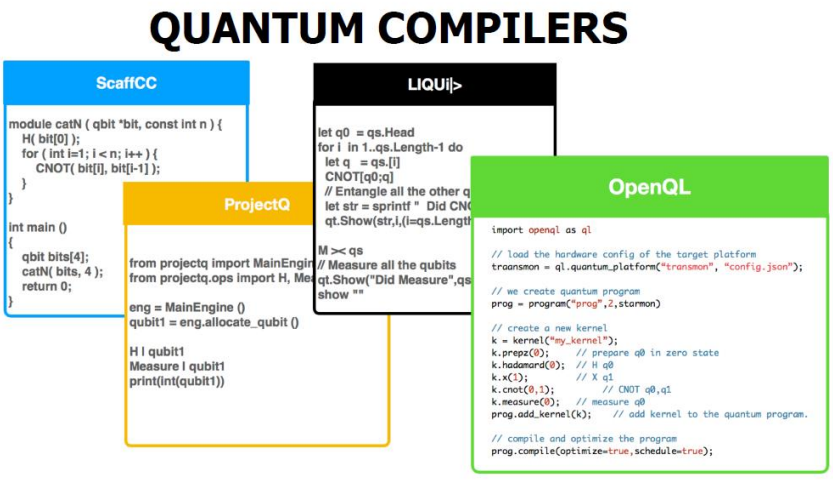
next to each other, combined with smart scheduling techniques and routing techniques will help to decrease the failures of your computation. Finally, another task of the compiler is to translate this high-level description of your circuit for the algorithm into its fault tolerant version. As you know, quantum bits are error prone, that means that they easily lose their state due to the interaction with the environment and therefore quantum error corrections and fault tolerant techniques are required.

FAULT TOLERANT QUANTUM CIRCUITS



Carmen G. Almudever,Quantum Compiler

A logical qubit is encoded into several physical qubits and the system needs to be continually checked in order to detect and correct possible errors. The use of quantum error correction will substantially increase not only the number of qubits that you will need in your algorithm, but also the number of operations. As, it was mentioned in the first part of this lecture, that there are several compilers for quantum computers such as ProjectQ from ETH Zürich, ScaffCC from Chicago University, LIQUi|> from Microsoft and our OpenQL compiler that has been developed here at QuTech. All of these compilers generate a kind of Quantum Assembly Language or QASM that can target different back-ends. Those back-ends can be a simulator, used to simulate quantum computation or it can be real quantum processors such as the ones that we are developing here at QuTech or the ones from IBM, Rigetti or Google. So, just to wrap up: in this lecture, we have learned that in order to express your quantum algorithm, a high-level programming language is needed.



Carmen G. Almudever,Quantum Compiler

A compiler which translates this high-level description into executable code that will make the required optimizations and transformations to obtain a circuit that is runnable on a quantum chip. These quantum instructions are part of what is called the quantum instruction set architecture and they will be sent to the micro-architecture for the execution on the quantum chip.

Main takeaways

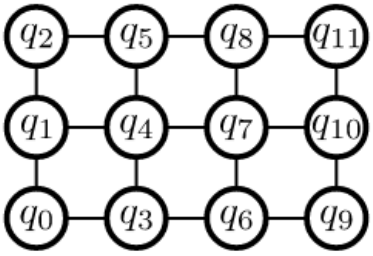
- Many quantum algorithms involve unitary gates which must be decomposed into single- and two-qubit gates in order to be executed.
- The most popular family of operations for decomposition is {*H*,*T*,CNOT}, because these operations can, in many quantum error-correcting codes, be executed easily.
- We must then decompose these operations further, into those that can be executed directly on the chip, also taking into account connectivity and time constraints.

Practice Questions

One of the convenient properties of entangled states is that, when a quantum system is in an entangled state, an operation on any component of that system can have consequences for the entire state. For example, consider the family of two-qubit states of the form $\alpha|00\rangle+\beta|11\rangle$. This state has two parameters, α and β, so it is mathematically identical to a one-qubit state, even though it is distributed over two qubits (this is an example of a *logical* state, which we will cover in more depth in the following module).

When a two-qubit register is prepared in such a state, a *Z* operation on either qubit will result in the state $\alpha|00\rangle-\beta|11\rangle$, which has the same effect as mapping the vector $[\alpha\beta]$ to $[\alpha-\beta]$. In effect, a logical *Z* operation has been applied. This, in turn, means that controlled-*Z* operations can be applied to the logical qubit, as long as one of the physical qubits holding the logical state is adjacent to the control qubit.

Consider the following 12-qubit layout:



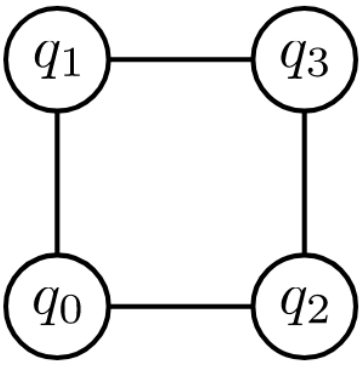
Delocalized Qubits

If a logical state is distributed across qubits q4 and q7 above, which of the following qubits are adjacent for the purpose of performing controlled-Z operations?

- ☐ q0
- ☐ q1
- ☐ q8
- ☐ q9

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

In the following question, we consider a device with four qubits:



Address Tables and SWAP-based Routing

Qubits are placed at the physical addresses (0,0), (0,1), (1,0), and (1,1), with labels as shown in the figure above. We can swap states between qubits using these physical addresses, as long as the qubits at those addresses are connected. For example, we can perform SWAP(0,0),(1,0), which will swap whichever vritual qubits are at those addresses (q0 and q1 above).

Which of the following sequences will swap q0 and q3, and leave the others stationary?

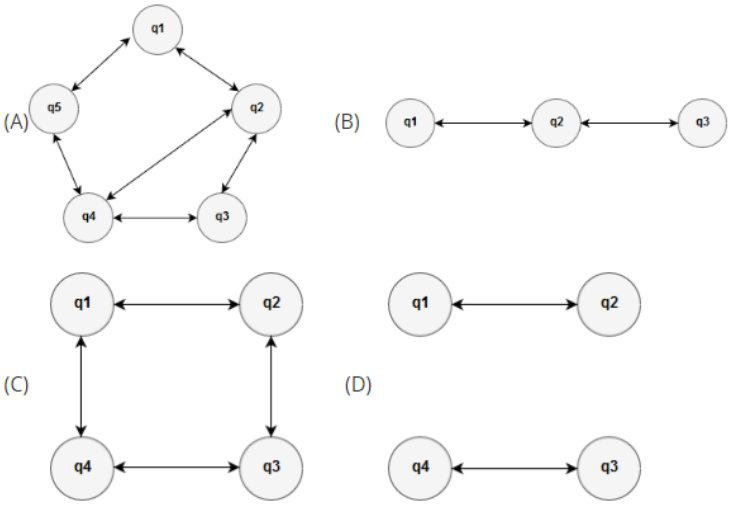
Select all correct answers.

- ☐ SWAP(0,0),(1,0), SWAP(1,0),(1,1), SWAP(0,0),(1,0)
- ☐ SWAP(0,0),(0,1), SWAP(0,1),(1,1), SWAP(1,1),(1,0)
- ☐ SWAP(0,0),(0,1), SWAP(0,1),(1,1), SWAP(0,1),(0,0)

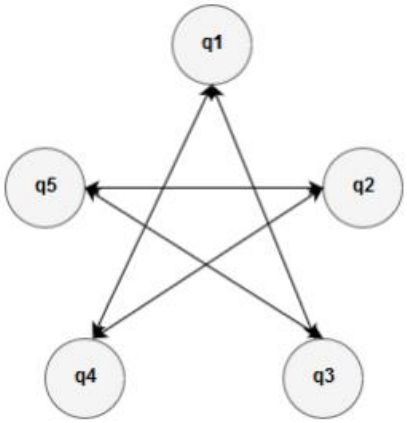
Quiz 4: Compiler & programming languages

Gate connectivity

For questions 4 and 5, we imagine that we are given quantum devices with the connectivity graphs seen below.



Shown below is a connectivity graph of a 5-qubit quantum processor. Using this as a reference, answer Question 5.



Module 3: Quantum algorithms & error correction

Introduction to Module 3

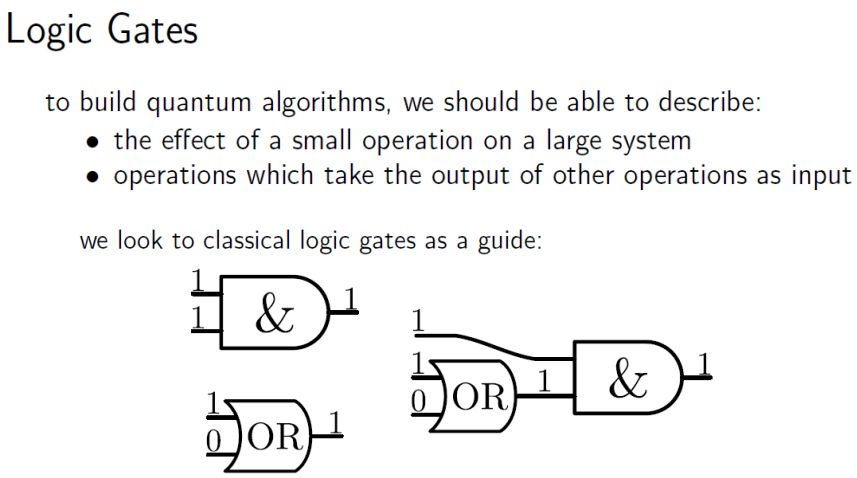
In the previous modules of this course, we have traversed the lower layers of the quantum stack, seeing how the instructions that make up a quantum algorithm can not only be stored on classical computers, but also manipulated by classical algorithms, in order to account for the individual characteristics of different quantum devices. Now we are prepared to ascend to the higher, more abstract layers of the stack, which are concerned with quantum circuits, quantum algorithms, and the quantum error-correction techniques that are required to make them resist the noise that affects quantum hardware.

This module begins with a discussion of quantum circuits and algorithms led by Ben Criger. Over the course of four densely-packed lectures and one quiz, Ben will touch on topics including the equivalences between small gates and circuits, phase kick-back, and the two 'textbook' quantum algorithms which use it: quantum phase estimation (which is also the prime area of application for the quantum Fourier transform) and Grover's algorithm. Over the next two lectures and one quiz, Professor Barbara Terhal will introduce quantum error correction, a vital technique for using noisy qubits and imprecise control to carry out quantum algorithms in an arbitrarily precise manner.

We hope that you enjoy this deep dive into a cutting-edge field of theoretical research.

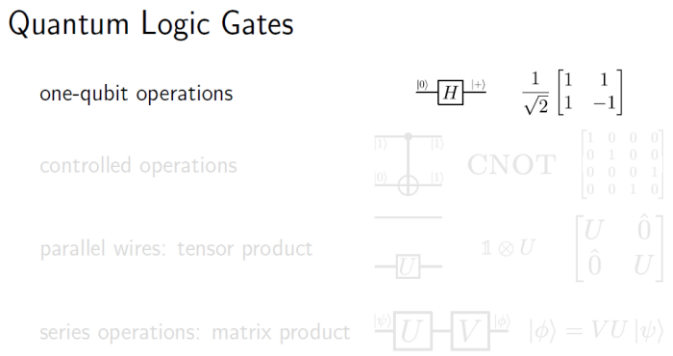
Quantum circuits

One of the important tasks of a quantum compiler, as discussed in the previous module, is to identify operations in an abstract quantum algorithm which cannot be executed directly, and replace them with sequences of operations which are logically equivalent (having the same effect on the qubits' states), and are composed of operations which can be executed 'on-chip'. But how do we know how to make such transformations? In the first lecture, Ben Criger will go over the common mathematical techniques which allow us to express and manipulate quantum circuits.



Ben Criger!,Algorithms:!,Quantum Circuits !

An algorithm is a sequence of steps, which solves a problem when carried out in order. Today, we're going to discuss how a quantum algorithm can be understood as a similar set of steps. In order to accomplish this, we're going to need a language that lets us assemble the quantum operations we've seen so far into larger procedures. Such a language should have two properties. First, it should be easy to describe the effect of a small operation on a large system, when we don't do anything to the rest of the system. Second, we should be able to easily write out a procedure that uses the output of one operation as the input for another. Of course, there's no reason to re-invent the wheel, we can look to classical logic gates for inspiration. Here, we see two of the logic gates of classical computing, AND and OR, each of which has its inputs and outputs carried by these wires. We can place an empty wire next to the OR gate to denote a bit which is passed directly through the circuit, and we can connect one of the output wires of the OR gate to the input of the AND gate, allowing us to build large, complicated logical circuits out of small, simple components. We can do something very similar with quantum operations.

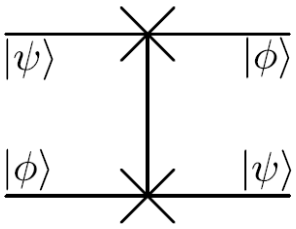


Ben Criger!,Algorithms:!,Quantum Circuits !

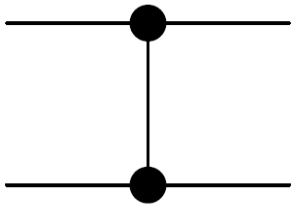
Here, we see the Hadamard operator, which has one input qubit and one output qubit. It's described by the 2-by-2 matrix on the right. There are also two-qubit operations, such as the CNOT, which has two inputs and two outputs. This operation is described by the 4-by-4 matrix also seen on the right. Now we come to the two techniques we use to compose small operations into large circuits. The first is that, whenever there's an empty wire, we place the identity operator on it, and

in order to calculate the total unitary that results from performing a small operation on a large set of wires, we take the tensor product of the operation with those identity operators. The second technique is for feeding the outputs of one stage of a circuit forward into a new stage, and it's perhaps a little simpler. We simply take the regular matrix product of the two matrices that describe the stages. It's important to be careful here, since the correct order for placing these matrices is the reverse of the order that they appear in the circuit. This is because, in the matrix product, the right-most matrix is the first one applied to the state psi. Before we use these concepts to analyze a quantum circuit, here are two more examples of two-qubit quantum operations which you may see later on, SWAP and CZ, which is also called CPHASE.

Quantum Logic Gate Examples



$$\text{SWAP} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



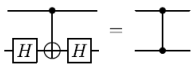
$$\begin{matrix} \text{CZ} \\ \text{CPHASE} \end{matrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Ben Criger!,Algorithms:!,Quantum Circuits !

SWAP can be used to exchange two states between two wires, which is useful if qubit states need to be moved from one place to another for some reason. The CZ, on the other hand, only places a minus sign on the one-one component of whatever state is input. Now that we've seen the CZ operation, let's take a look at how to analyze a quantum circuit which involves the CZ. Specifically, we're going to try to prove that we can generate a CZ using a CNOT and some Hadamard gates on the target qubit, that come before and after the CNOT.

Circuit Analysis

multiply matrices for small circuits
calculate effects on computational basis states
decompose into tensor products



$$I \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

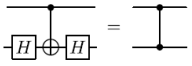
Ben Criger!,Algorithms:!,Quantum Circuits !

Circuit Analysis

multiply matrices for small circuits

calculate effects on computational basis states

decompose into tensor products



$$(I \otimes H)\text{CNOT}(I \otimes H) =$$

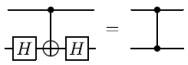
$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}$$

Circuit Analysis

multiply matrices for small circuits

calculate effects on computational basis states

decompose into tensor products



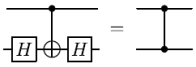
$$= \frac{1}{2}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & -1\\ 0 & 0 & 1 & 1\end{bmatrix}$$

Circuit Analysis

multiply matrices for small circuits

calculate effects on computational basis states

decompose into tensor products



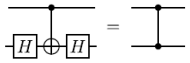
$$= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix} = \text{CZ}$$

Circuit Analysis

multiply matrices for small circuits

calculate effects on computational basis states

decompose into tensor products



$$
\begin{array}{llllll}
|00\rangle & \xrightarrow[I\otimes H]{} & |0+\rangle & \xrightarrow[\text{CNOT}]{} & |0+\rangle & \xrightarrow[I\otimes H]{} & |00\rangle \\
|01\rangle & \xrightarrow[I\otimes H]{} & |0-\rangle & \xrightarrow[\text{CNOT}]{} & |0-\rangle & \xrightarrow[I\otimes H]{} & |01\rangle \\
|10\rangle & \xrightarrow[I\otimes H]{} & |1+\rangle & \xrightarrow[\text{CNOT}]{} & |1+\rangle & \xrightarrow[I\otimes H]{} & |10\rangle \\
|11\rangle & \xrightarrow[I\otimes H]{} & |1-\rangle & \xrightarrow[\text{CNOT}]{} & -|1-\rangle & \xrightarrow[I\otimes H]{} & -|11\rangle
\end{array}
$$

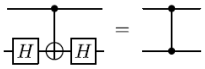
Ben Criger!,Algorithms:!,Quantum Circuits !

Circuit Analysis

multiply matrices for small circuits

calculate effects on computational basis states

decompose into tensor products



$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

$$(I \otimes H)\text{CNOT}(I \otimes H) = |0\rangle\langle 0| \otimes H^2 + |1\rangle\langle 1| \otimes HXH$$

$$H^2 = I \qquad HXH = Z$$

$$|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z = \text{CZ}$$

Ben Criger!,Algorithms:!,Quantum Circuits !

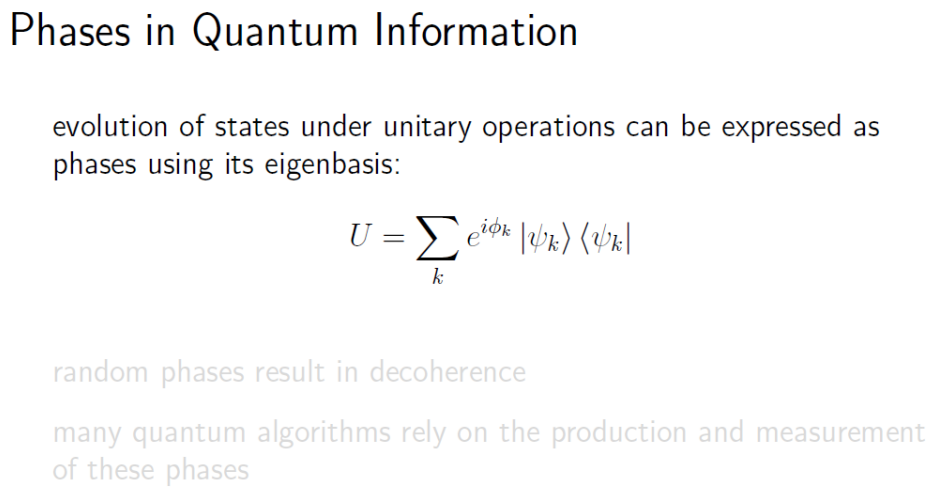
Now, there are three ways to accomplish this, and we'll go through each of them. The first is the simplest, but also the most time-consuming. We're going to take tensor products of one-qubit operations with identities on the empty wires, as seen here. Then, we can write out the entire circuit in matrix form. Multiplying these matrices one at a time gradually reduces the size of the problem. And once we've multiplied them all, we can see that the matrix describing the entire circuit is equal to the matrix described in the CZ that we saw earlier. A simpler technique, which works well on this circuit, is to calculate the effects on individual basis kets without writing them out in vector form. This saves us a little space. Here we can see that the effect of the Hadamard gate on the second qubit is to transform its state into the plus-minus basis. The CNOT puts a minus sign on the one-minus component of the state, since plus and minus are the plus- and minus-one eigenstates of the X operator, and the final Hadamard returns the states to the standard basis. This is identical to the action of the CZ gate on these basis states. The third and final method, which we'll use to analyze quantum circuits is to express the relevant tensor products without using matrices, then try to simplify the resulting expression for the circuit. As we can see, the tensor product expression for CNOT sandwiched with Hadamards simplifies to zero-zero tensor H squared plus one-one tensor H X H. It's easy to show using two-by-two matrix multiplication or ket notation that H-squared is the identity, and H X H is Z, leaving us with a controlled-Z operation, whose action we can verify using basis states. These three techniques, and the quantum circuits that they're used to analyze, each have a role to play in the design of new quantum protocols, and are important tools for us to use to further our understanding of quantum algorithms.

Main takeaways

- Quantum gates act in a manner similar to classical logic gates. They have inputs and outputs, and can be composed together into more sophisticated circuits.
- Many methods exist for proving the equivalence of two circuits. If the circuits under consideration are small, all methods work. As the circuits become larger, more technical methods are used to reduce the amount of mathematics to something we can handle.

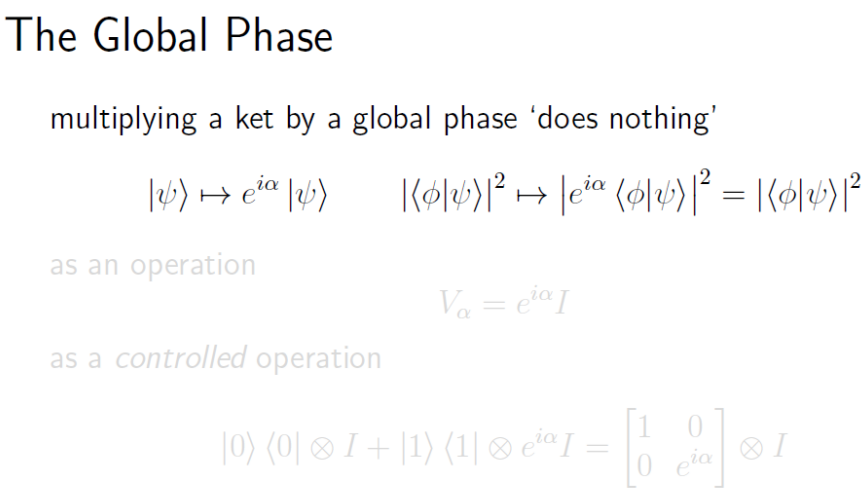
Phase kickback

Many quantum algorithms involve applying a unitary operator to one of its eigenstates, resulting in a phase being applied to the input state. For example, Shor's factoring algorithm and Grover's search algorithm both involve this step. There is a surprisingly simple trick for reading these phases out, which Ben will introduce to you here.



Ben Criger!,Algorithms:!,Phase kickback !

The execution of a quantum algorithm requires us to prepare states, perform quantum circuits on them, and then to read out the result. Today we're going to introduce a commonly-used technique for this final step which uses some relative phases. These phases, which are numbers of the form e-to-the-i-phi, play a very important role in quantum mechanics, quantum information, and quantum computation. Every unitary operator can be written out in its eigenbasis, and its action is reduced to a set of these phases. If we put one of the eigenstates in, the output is the same state, multiplied by that phase.

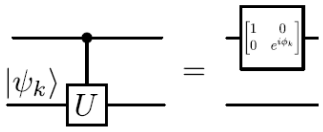


Ben Criger!,Algorithms:!,Phase kickback !

If a phase is randomly applied to some of the components of a quantum state, this decoheres the quantum state, so keeping track of phases is important even at the physical level, before we discuss any quantum algorithm. Nevertheless, many quantum algorithms do involve the accumulation of non-random phases, so it's convenient that there's a way to read these phases out, which we'll derive in this lecture. Let's first look at the simplest of all phases, the global phase.

Phase Kickback

if we apply a controlled-U to $|\psi_k\rangle$ (an eigenstate of U), $|\psi_k\rangle$ is unchanged, and $|0\rangle\langle 0| + e^{i\phi_k}|1\rangle\langle 1|$ is applied to the *control* qubit



this is still true for many-qubit U

Ben Criger!,Algorithms:!,Phase kickback !

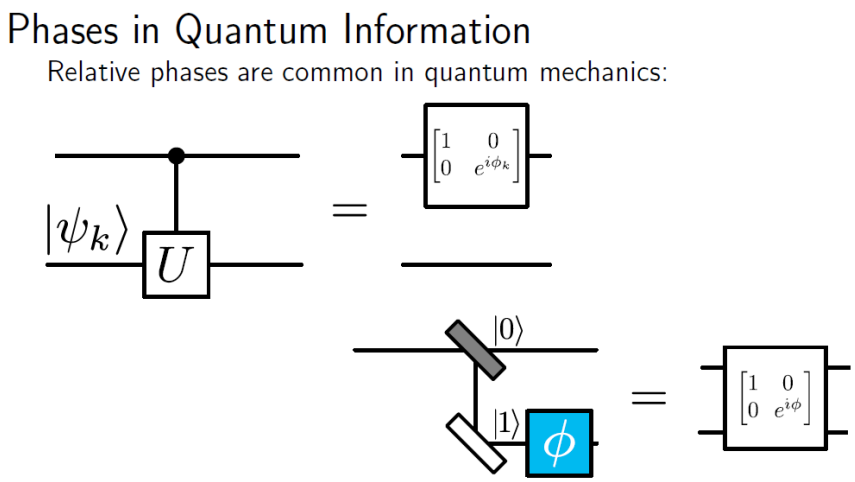
If we had an operation that multiplied a state by a global phase, like a unitary does when one of its eigenstates is input, the output probabilities that are defined by Born's rule don't change. This means that the global phase has no physically measurable effect. Nevertheless, we can define an operator V-alpha which multiplies any input state by a global phase. It's pretty simple, just e-to-the-i-alpha times the identity. Nothing special. However, something interesting does happen when we define a controlled-V-alpha operation. A phase is accumulated on the one component of the state of the control qubit. This is a stark contrast to the normal way of thinking about controlled operations, in which something happens to the target qubit, depending on the state of the control. This implies something called phase kickback. If we apply a controlled-U operation to a register where the target subsystem is prepared in an eigenstate of U, then U acts like the V_alpha operator, and the corresponding phase is transferred to the control qubit, where it can then be read out, in principle. Even in the case where a single read-out doesn't specify the phase exactly, this procedure can be repeated, since it doesn't alter the input state psi_k. This is still the case if the target register contains many qubits, or other types of quantum systems. The versatility and simplicity of using a single qubit to read out the action of an arbitrarily large unitary is what makes phase kick-back an important building block for quantum algorithms.

Main takeaways

- An operator *U*, applied to one of its eigenstates $|\psi\rangle$, results in a state $e^{i\phi}|\psi\rangle$. Normally, such a *global* phase produces no measurable effect.
- Using a single ancilla qubit, we can turn this global phase into a *relative* phase, whose influence on an input state can be measured.

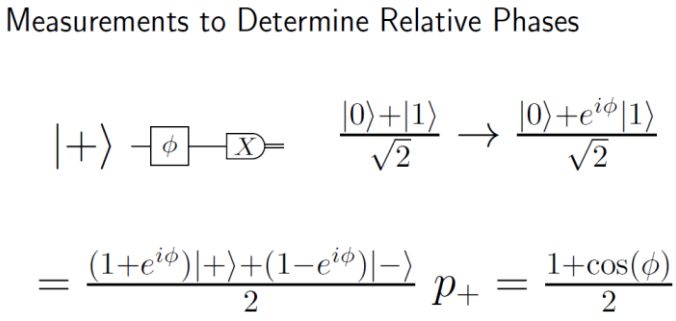
Quantum phase estimation and the quantum Fourier transform

We've learned, in the previous Section, how to record the phase that an eigenstate of a unitary picks up when that unitary is applied. But how do we determine what this phase is, given the probabilistic nature of quantum measurements? In this Section, Ben will discuss the two different scenarios in which a relative phase has to be measured, and the two different methods which are specialised to deal with each scenario.



Ben Criger!,Algorithms:!,Phase estimation!

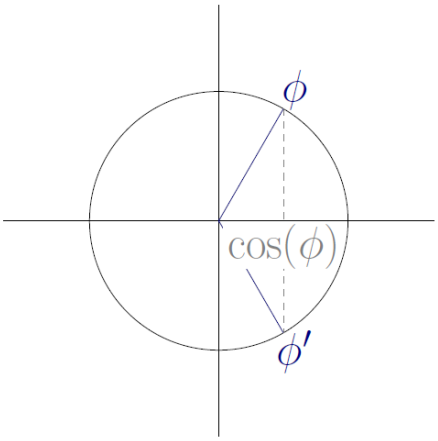
We've seen previously how quantum phases can be used to output the results of a quantum algorithm. Now we're going to see a technique for estimating these phases that incorporates another quantum algorithm, in order to ensure that we get a precise estimate. Relative phases are important and ubiquitous in quantum information. We can produce one of these relative phases using a unitary U by inserting an eigenstate of U, which we call psi_k, into a controlled U operation, resulting in the corresponding phase phi_k being kicked back onto the control qubit. Alternately,we can produce a relative phase in a physics experiment.For example, assume that we have a beam-splitter here that splits an incoming light beam into two paths, one of which passes through a piece of material that shifts the phase of the light's wavefunction. This can produce a physical state with that relative phase on it. Now we come to the question, how do we learnthe values of these phases? If we input a plus state to the phase gate phi, and then measure in the X basis, the plus state transforms from 0 + 1 over root 2 to 0 + e to the i phi 1 over root two, which has coefficients 1 + e to the i phi and 1 - e to the i phi in the +- basis.



Ben Criger!,Algorithms:!,Phase estimation!

This results in a probability of measuring the output to be in the + state that is 1 + cos(theta) over 2. Let's take a look at the performance of the cosine of phi measurement. Now, phi is a point on a circle and the cosine of phi is the x coordinate of phi, just on the x-axis there. Already we can see one of the problems with doing a cosine of phi measurement on its own, there's another angle phi' with the same cosine as phi, making it impossible to tell the difference between them with only these measurement results. Now, if that weren't a big enough problem, there's also the fact that, if we have some

Measurements to Determine Relative Phases



ϕ is a point on a circle

$\cos(\phi)$ is its x coordinate

but $\cos(\phi) = \cos(\phi')$

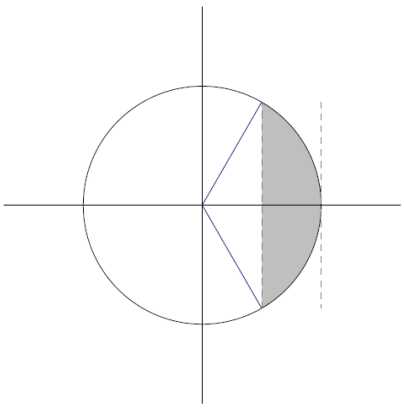
$\phi \sim 0 \rightarrow$ poor precision

$\phi \sim \pi/2 \rightarrow$ good precision

Ben Criger!,Algorithms:!,Phase estimation!

imprecision on our measurement of cos(phi), suppose for instance that we know the cosine of phi is in this gray band, but not where exactly, then our estimates of phi get very imprecise when phi is near 0. This is in contrast with the situation when phi is close to pi/2, or 90 degrees, where the same amount of imprecision on cos phi results in much less imprecision on our estimate of phi, a much smaller wedge seen here. So how do we fix this? Well, we can run a second type of experiment which involves preparation of the +i state, which is 0 + i 1, then we feed that through the phase gate, and measure out in the x basis as before.

Measurements to Determine Relative Phases



ϕ is a point on a circle

$\cos(\phi)$ is its x coordinate

but $\cos(\phi) = \cos(\phi')$

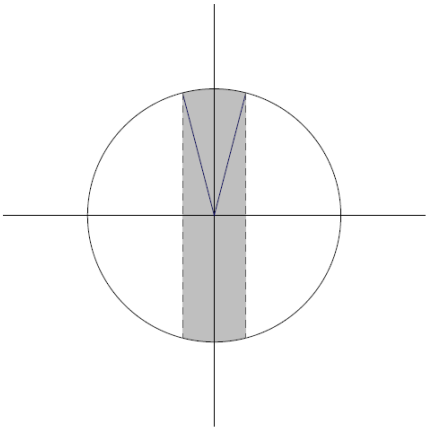
$\phi \sim 0 \rightarrow$ poor precision

$\phi \sim \pi/2 \rightarrow$ good precision

Ben Criger!,Algorithms:!,Phase estimation!

If we do the same algebra to analyse this experiment, we end up with a probability of measuring the + state which now depends on the sine of phi, which is the y coordinate of phi in the diagram we drew previously. Now, we have only one range of angles that's consistent both with our imprecise estimates of sine phi and the cosine of phi. Also, note that the precision of the estimate is independent of phi itself, and remains the same as we travel around the circle. This precision, the width of the interval for phi, scales as one over the square root of s, where s is the number of measurements.

Measurements to Determine Relative Phases



ϕ is a point on a circle

$\cos(\phi)$ is its x coordinate

but $\cos(\phi) = \cos(\phi')$

$\phi \sim 0 \rightarrow$ poor precision

$\phi \sim \pi/2 \rightarrow$ good precision

Ben Criger!,Algorithms:!,Phase estimation!

Now that's all well and good, but you may wonder whether we can do better. We can, in fact, but first we'll have to consider a simplified case of phase estimation. Let's focus on the phases 0 and pi. In the + state experiment from earlier, these angles produce a probability of measuring + that's either 0 or 1 exactly. As long as we're guaranteed that the phase we're trying to estimate is one of these two values, then one measurement suffices to give us exact knowledge of the phase, so we don't need to worry about precision.

Measurements to Determine Relative Phases

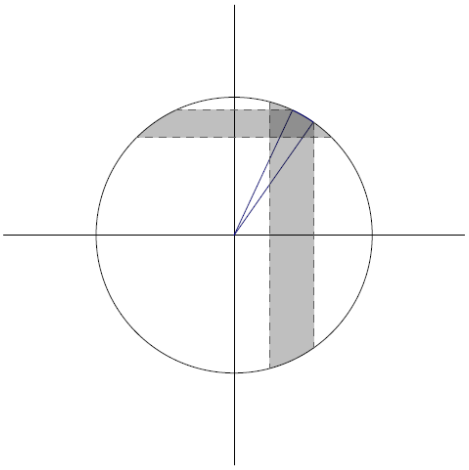
$$|+i\rangle \; -\boxed{\phi}-\boxed{X}= \qquad \frac{|0\rangle+i|1\rangle}{\sqrt{2}} \rightarrow \frac{|0\rangle+ie^{i\phi}|1\rangle}{\sqrt{2}}$$

$$= \frac{(1+ie^{i\phi})|+\rangle+(1-ie^{i\phi})|-\rangle}{2} \; p_+ = \frac{1-\sin(\phi)}{2}$$

Ben Criger!,Algorithms:!,Phase estimation!

To generalise to other, smaller angles, we'll need some notation, namely that of binary fractions. Here we express phi as 2 pi times a number omega, which is between 0 and 1. Omega is a sum over these values b_k, eachof which can be 0 or 1, times 2 to the minus k, for k ranging from one to infinity. We can write this out as 0.b1b2b3 et cetera. For example, one over 2 will now be denoted 0.1, where normally you would use 0.1 to denote 1/10, in base ten.Also, one eighth, 1/2^3, will be 0.001, and we can express other fractions like 3/8 as binary strings like 0.011. In order to make full use of this notation, we'll also need to note that we can control the phase that gets kicked back by applying multiples of phi, rather than phi itself. This is true both in the case where we use phase kickback on a quantum circuit, and in the case where we're doing a physics experiment.

Measurements to Determine Relative Phases



only one ϕ matches $\sin(\phi)$ and $\cos(\phi)$

precision is good, regardless of ϕ

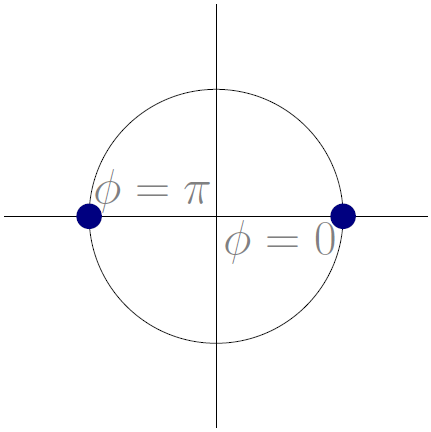
$\Delta\phi \sim \frac{1}{\sqrt{s}}$

Ben Criger!,Algorithms:!,Phase estimation!

The main difference between these two cases is that, in a quantum computer, U to the Mth power may be implemented more cleverly and in less time than simply doing U M times in a row, as you have to do if you're running one of these physics experiments. For this reason, let's assume from now on that we're doing simulated U in a quantum computer rather than a direct physical U in a physical experiment. Now we're ready to estimate phases that have one of four values; 0, pi/2, pi or 3 pi/2. There are a few convenient facts to discuss. First, we recall that e to the 2 pi i is 1. Next, we use this fact to derive that applying the four-valued phase twice gives us a two valued phase which is either 0 or pi, depending only on the value of b_2. This gives us an intuitive algorithm to follow to learn these phases with these four values. First, we apply the phase 2 phi to the + state, and perform a single measurement to learn the value of b2. Then, we prepare a new special input state that cancels out the phase 0.b2 and perform another measurement to learn b1. As long as we can adjust the phase on the input state such that the output state always has a phase of 0 or pi, one measurement is all we need to determine that bit of the phase exactly.

Special Phases: 0, π

To increase precision, consider phases we can measure *exactly*:



in the $|+\rangle$-state experiment, $p_+ = 0$ or 1 exactly.

as long as it's one of those two phases, we get full precision with one measurement.

Ben Criger!,Algorithms:!,Phase estimation!

Aside: Binary Fractions

To measure other phases exactly, we need *binary fractions*:

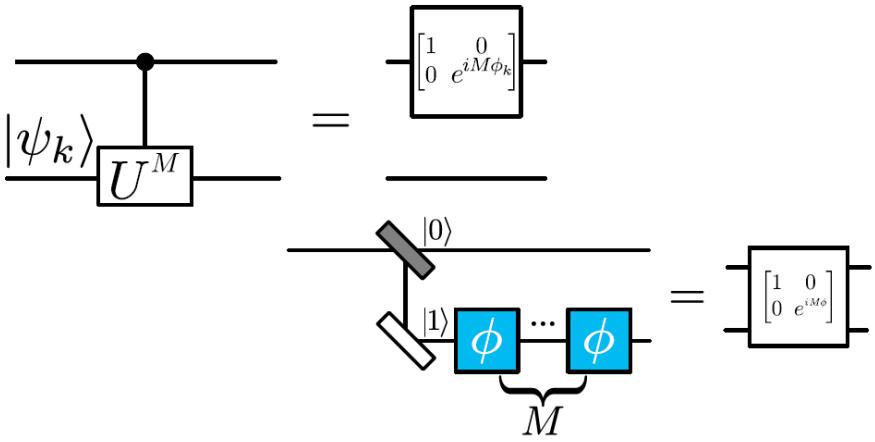
$$\begin{aligned}\phi &= 2\pi \times \omega \\ &= 2\pi \times \textstyle\sum_{k=1}^{\infty} b_k 2^{-k} \\ &= 2\pi \times (0.b_1 b_2 b_3 \ldots)\end{aligned}$$

For example:

$$\frac{1}{2} = 0.1 \qquad \frac{1}{8} = 0.001 \qquad \frac{3}{8} = 0.011$$

Aside: Powers of Unitaries

We can *control* the kicked-back phase, applying ϕ multiple times:



Special Phases: 0, $\pi/2$, π, $3\pi/2$

Convenient Facts

$\exp(2\pi i) = 1$

$\exp(2i\phi) = \exp(2\pi i 2\omega) = \exp(2\pi i b_1.b_2) = \exp(2\pi i 0.b_2)$

Algorithm

Apply 2ϕ to $|+\rangle$ and measure to learn b_2

Apply ϕ to $\frac{|0\rangle + \exp(-2\pi i 0.0 b_2)|1\rangle}{\sqrt{2}}$ and measure to learn b_1

Now, in order to estimate arbitrary phases, we begin by choosing a number of measurements that we'd like to do, and we'll call it s again for simplicity. We apply the phase 2 to the s - 1 phi to + to learn the last bit of b that we're interested in, called b_s. Then, we adjust the phase on the input state to get rid of the last bit on the output, and we apply 2^{s - 2} phi, so that the output phase is 0.b_{s-1}. After this, we can continue cancelling known bits of the phase by shifting the

Estimating Arbitrary Phases

Choose s, the number of measurements

Apply $2^{s-1}\phi$ to $|+\rangle$ to learn b_s

Apply $2^{s-2}\phi$ to $\frac{|0\rangle+\exp(-2\pi i 2^{s-1} b_s)|1\rangle}{\sqrt{2}}$ to learn b_{s-1}

Keep applying smaller phases to prepared input states to learn b_k.

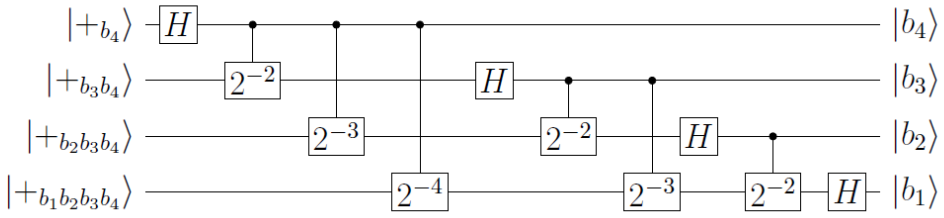
Precision $\sim \frac{1}{2^s}$

input state, and cancelling unknown bits of the phase that we're not yet ready to measure by applying a multiple of phi, until we've learned all the bits of omega that we wanted. Since this returns one of 2^s uniformly spaced values around the circle, we get a precision that's one over 2 to the s, much more precise than the 1 over root s scaling we saw earlier. And one of the final very special things we can learn about this procedure is that it can be implemented coherently, using all of the phased plus states at once. We take advantage of the fact that, if we perform a Hadamard on a state with a 0 or pi phase, we get a state that's 0 if the phase is 0, and one if the phase is pi. Then, we can use these states as the coherent controls to phase shifting gates that perform the shifts we discussed earlier. If we write out the entire circuit, we see that its inverse would take a bit string as input, and return a set of phased states where the phases encode those bit values. This is precisely the Quantum Fourier Transform, which underpins many quantum algorithms.

Estimating Aribtrary Phases

Conditional input state preparation can be done *coherently:*

$$H\left(\tfrac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i 0.b\right)|1\rangle\right)\right) = \frac{1+(-1)^b}{2}|0\rangle + \frac{1-(-1)^b}{2}|1\rangle$$



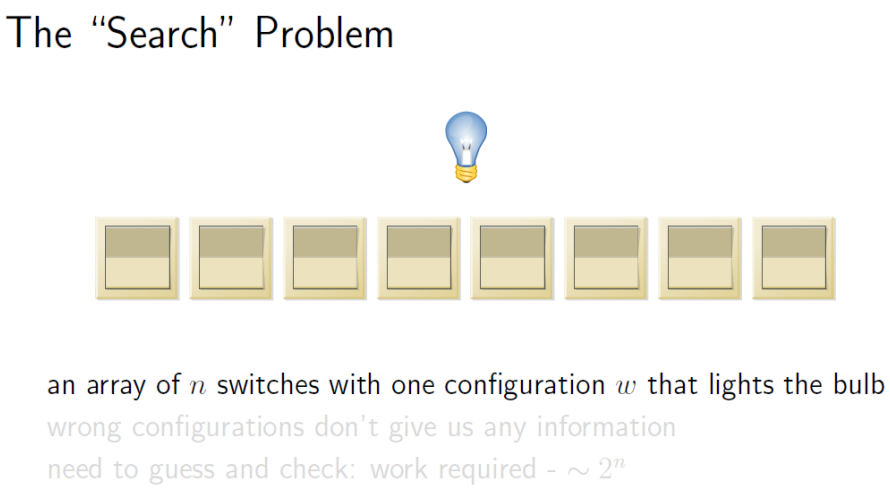
(Inverse) Quantum Fourier Transform

Main takeaways

- Due to the probabilistic nature of quantum measurement, it is necessary to prepare an input state multiple times in order to be able to learn the value of a phase with precision.
- Depending on whether the phase is produced by a quantum algorithm, it may be more efficient to apply phases of the form $2k\varphi$ to an array of ancilla qubits which are prepared in special input states which are specifically tailored to get the most out of each quantum measurement.
- The procedure we use to measure phases with high precision is the inverse of the quantum Fourier transform.

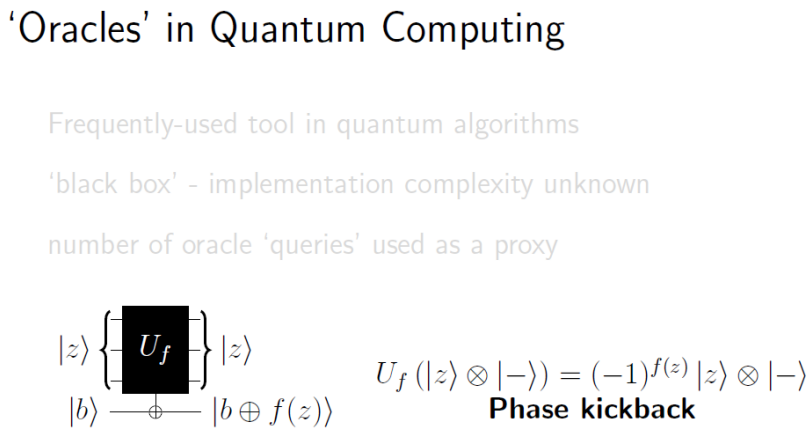
Grover's algorithm

In this Section, Ben introduces Grover's algorithm, which searches through bit strings to find one which is different from the others. This different element (often called 'the marked state') is labelled by an *oracle* or black-box function. This prevents us from evaluating the number of quantum gates required, as the circuit decomposition for such an oracle is, in general, unknown. Instead, we use the number of oracle *queries*, or uses, as a stand-in. This number of queries scales as N−−√, in contrast to the N queries which are required classically. This makes Grover's algorithm interesting, though less of a 'killer app' than quantum phase estimation or factoring.



Ben Criger!,Grover's algorithm!

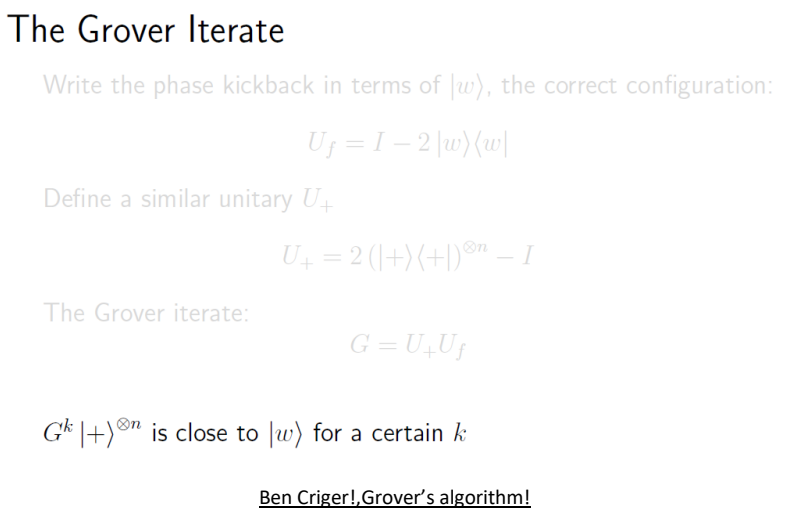
Now we're going to learn about another quantum algorithm that, while it doesn't provide as large a speedup over its classical counterpart, does still have an interesting structure that makes it worth discussing. The problem to be solved is as follows. Imagine that there is an array of n switches, which all have to be set in a correct configuration w, in order to light a bulb. Now, to make the problem difficult, some of these switches have been installed upside-down, and there's no manual lying around, so the all-up configuration isn't necessarily correct, as it should be. If we tried a certain configuration of switches, some up some down, it wouldn't tell us anything about which switch was set incorrectly, all we would see is that the bulb is either lit or not. For this problem, if we limit ourselves to classical algorithms, there's no essentially better strategy than to guess and check repeatedly until we find w, the correct configuration. This requires a number of switch flips that scales as 2 to the n, which can get quite large as n grows.



Ben Criger!,Grover's algorithm!

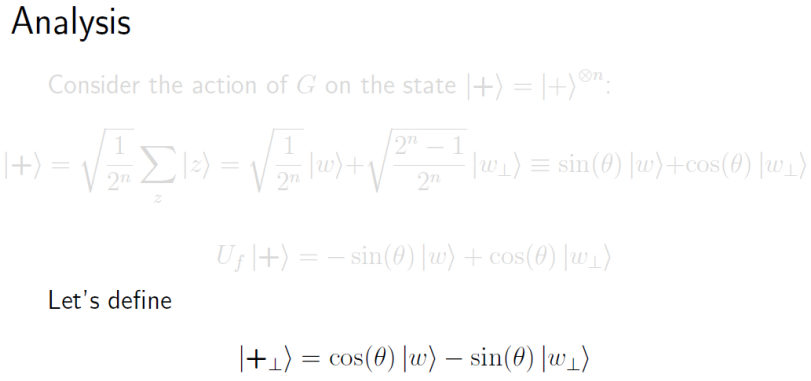
To lay out the corresponding quantum algorithm, we're going to need to make use of another building block, the oracle. Oracles are used in many different quantum algorithms, when the complete implementation of a given function is

unknown. An oracle is a black box that we can prove is unitary, though we don't know the circuit complexity of its implementation. In place of a proper complexity which is worked out in terms of the space and time cost of running a quantum circuit, we use the number of queries to the oracle, which is the number of times that it's used, as a proxy. The oracle for the lightbulb problem will look like this. We input a configuration z, some eigenstate in the z basis on each qubit, and if the bulb would light up, an ancilla qubit gets flipped. Already we can see that, if we input a minus state on the ancilla, we can obtain what I'll call U_f, which maps w to -w and acts as the identity on all other states in the z basis. We can see that the operation U_f can be written as identity minus 2 times w-w. If this is unclear, pause the Section, and take a moment to evaluate the action of this operator on the state w, and on other states which are orthogonal to w. We can also define another unitary u+ that does almost the same thing, except that it acts as the identity on the all plus state, and applies a minus one phase to any state orthogonal to the all plus state in the x basis. Now, we're going to take the product of these two operators and call it the Grover iterate G. We're going to show in the following section that if we apply G k times to the all plus state as an input, we end up with a state that's very close to w, the state described in the winning configuration.



Ben Criger!,Grover's algorithm!

First, let's write the all plus state out neatly. We can see that the all plus state is a uniform superposition over all states in the z basis, and that this necessarily includes w, regardless of what w actually is. There's also another term which is proportional to w perp, which is a uniform superposition of everything in the z basis that's orthogonal to w. We're going to write the coefficients of this wavefunction as angles to simplify our calculation later on.



Ben Criger!,Grover's algorithm!

Now we can see that the first part of G, U_f, just places a minus sign on the w term, easy enough. To see the consequence of then performing U_+, though, we'll now have to define another state + perp, which is just a state that's orthogonal to the all plus state, written out using w and w perp.

Analysis

We change basis:

$$\begin{aligned} U_f |+\rangle &= -\sin(\theta) |w\rangle + \cos(\theta) |w_\perp\rangle \\ &= \left(\cos^2(\theta) - \sin^2(\theta)\right) |+\rangle - (2\sin(\theta)\cos(\theta)) |+_\perp\rangle \\ &= \cos(2\theta) |+\rangle - \sin(2\theta) |+_\perp\rangle \end{aligned}$$

We apply U_+ and change back:

$$\begin{aligned} G |+\rangle = U_+ U_f |+\rangle &= \cos(2\theta) |+\rangle + \sin(2\theta) |+_\perp\rangle \\ &= \sin(3\theta) |w\rangle + \cos(3\theta) |w_\perp\rangle \end{aligned}$$

Now we can express U_f + in the +, +perp basis, using a little bit of trigonometry. Eventually, we get to the expression cos 2 theta + - sin 2 theta + perp. Applying U_+ and changing the basis back, we can see that the Grover iterate takes our initial angle theta and replaces it with the angle 3 theta. If we did this k times in a row, we'd end up with not 3 theta, but 2 k + 1 theta.

Analysis

Applying G^k to $|+\rangle$ results in

$$\sin((2k+1)\theta) |w\rangle + \cos((2k+1)\theta) |w_\perp\rangle$$

In order to obtain $\sin((2k+1)\theta) \sim 1$, we need

$$\begin{aligned} (2k+1)\theta &= \frac{\pi}{2} \\ k &\sim \frac{\pi}{4\theta} \\ &\sim \frac{\pi}{4}\sqrt{2^n} \end{aligned}$$

This is the square root of the number of function evaluations required classically.

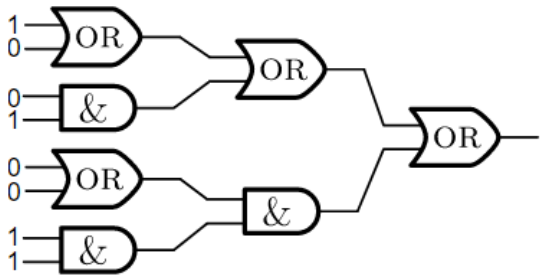
In order to obtain a state close to w, then, we have to choose k so that the sine of 2 k + 1 theta is close to one. Now, the sine is close to 1 when the angle is close to pi by 2, so this gives us a k roughly equal to pi divided by 4 theta, which is roughly pi/4 times root 2 to the n. This scales as the square root of the number of configurations we'd have to try in the classical case, providing us with an algorithm that has a polynomial speedup over the classical one.

Main takeaways

- Not all quantum algorithms are necessarily exponentially faster than classical algorithms. Grover's algorithm, for example, is only *quadratically* faster than the classical brute-force method.
- Some quantum algorithms use *oracles*, which are black-box unitary operators whose circuit decompositions are not known. The number of times the oracle is used (the number of *queries*) is used as a surrogate for complexity.

Practice Questions

In the question below, we will consider the following classical circuit:



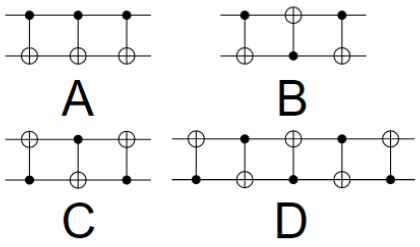
A Brief Classical Computation

Recall that the & gate outputs a 1 only if both of its inputs are 1, and that the OR gate will output a 1 if either or both of its inputs are 1.

What would be output by the circuit shown above?

- 0
- 1

Consider the following circuits:



Decomposing SWAP Gates

Which of the circuits above performs a SWAP operation?

Select all correct answers.

- [ ] A
- [ ] B
- [ ] C
- [ ] D

Phase Kickback Example

In phase kickback, a controlled version of the unitary U is applied to one of its eigenstates |ψ⟩, resulting in a relative phase of exp(iφ) being applied to the control register.

Consider a phase kickback protocol where $U = \frac{1}{3}\begin{bmatrix} 2+\frac{1+i}{\sqrt{2}} & 1-\sqrt{2}+i \\ 1-\sqrt{2}+i & \sqrt{2}+1+\sqrt{2}i \end{bmatrix}$, and $|\psi\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ \sqrt{2} \end{bmatrix}$.

What is the resulting phase φ?

- 0
- π8
- π4
- |ψ⟩ is not an eigenstate of U, φ is undefined.

Tensor Products

In the lecture, Ben discussed the tensor product as a way to calculate the effect of a quantum gate when applied to a small component of the total system. To calculate the tensor product of two matrices A and B, A⊗B, we replace every element Ajk of A with a block, or submatrix, equal to Ajk×B.

We can get a better understanding for tensor products by looking at an example.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We can write out the tensor product $A \otimes B$ using lines as a visual aid to divide the matrix into blocks:

$$A \otimes B = \left[\begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ \hline 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{array}\right]$$

We see that the upper left block is equal to $A_{00}B$, the upper right block is equal to $A_{01}B$, and so on.

Now consider the following two-qubit unitary operator:

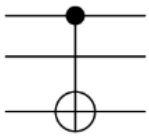
$$U = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & -i & 0 & -i \\ i & 0 & i & 0 \\ 0 & -i & 0 & i \\ i & 0 & -i & 0 \end{bmatrix}$$

Which tensor product expression corresponds to U?

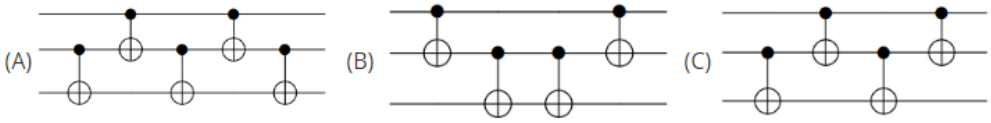
- X⊗I
- X⊗Y
- H⊗Y
- Z⊗Z

Quiz 5: Quantum algorithms

In this question, we consider a three-qubit device. We wish to perform a CNOT between two qubits which are not adjacent, but are both adjacent to a 'middle' qubit, shown below:

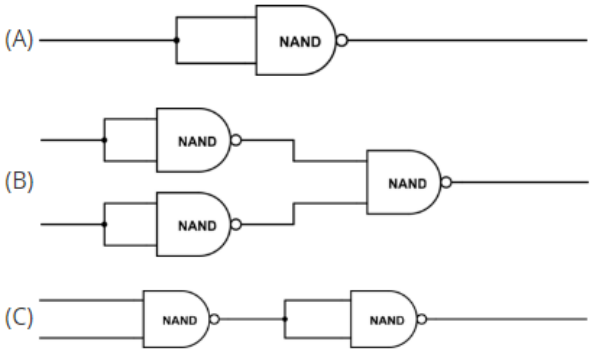


Given below are three quantum circuits. Using these as options, answer the following question.



Classical circuit decomposition

Shown below are three classical circuits constructed using boolean NAND gates (which map classical bit values {x,y} to 1, unless both inputs are 1, in which case, the output is 0). These circuits also involve 'fan-out' operations, which map a single classical bit to two classical bits with the same value.

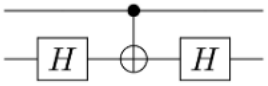


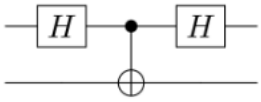
Quantum composition

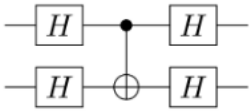
Below we see a CNOT gate with the top qubit as the target:

Suppose that we are only permitted to use CNOTs which have the top qubit as the control, and that we have access to single-qubit unitaries.

Consider the following circuits:

(A) 

(B) 

(C) 

Global and relative phases

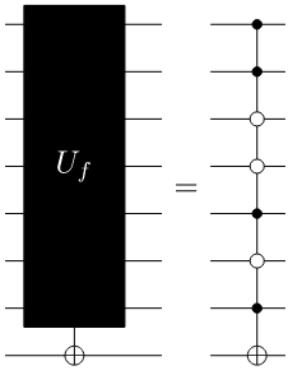
Here, we see a set of unitary matrices. Using these as a reference, answer the following question.

(A) $\sqrt{\frac{1}{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ (B) $\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}$ (C) $\sqrt{\frac{1}{2}}\begin{pmatrix}-1 & -1\\ -1 & 1\end{pmatrix}$ (D) $\sqrt{\frac{1}{2}}\begin{pmatrix}i & i\\ -i & i\end{pmatrix}$

(E) $\sqrt{\frac{1}{2}}\begin{pmatrix}i & i\\ i & -i\end{pmatrix}$ (F) $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ (G) $\sqrt{\frac{1}{2}}\begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}$ (H) $\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$

A circuit for the Grover's oracle

Below, we see a circuit which, just like the oracle in Grover's algorithm, flips the bottom qubit if and only if the other qubits in the register are set to a specific state:

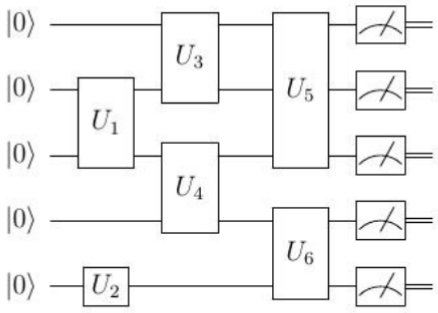


Here, the black control dots indicate that the control qubit must be in the $|1\rangle$ state in order for the NOT gate to be applied, and white control dots imply that the control qubit must be in the $|0\rangle$ state.

Consider this oracle while answering the question below.

Quantum Error Correction - Codes

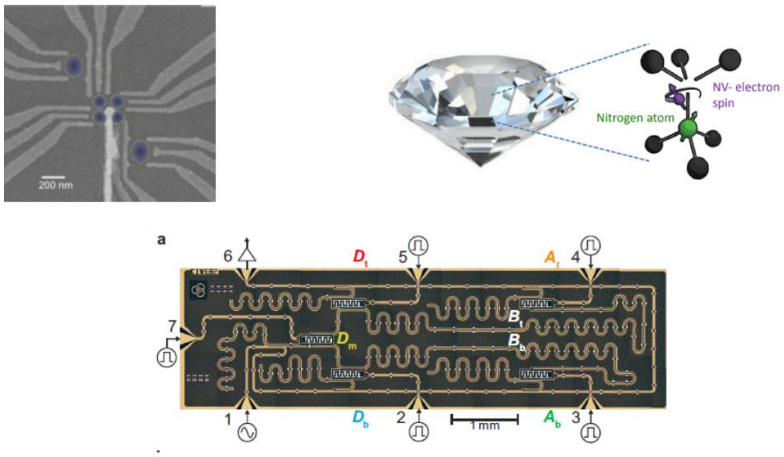
Large computations, either classical or quantum, require each of their operations to have a low logical error rate in order to succeed. Otherwise, the incorrect results of a few operations can be fed forward, and the whole computation will output random results. This is an example of the well-known "Garbage In, Garbage Out" principle from computer science.



Some 5-qubit quantum circuit with 6 gates

Barbara Terhal,Quantum Error Correction

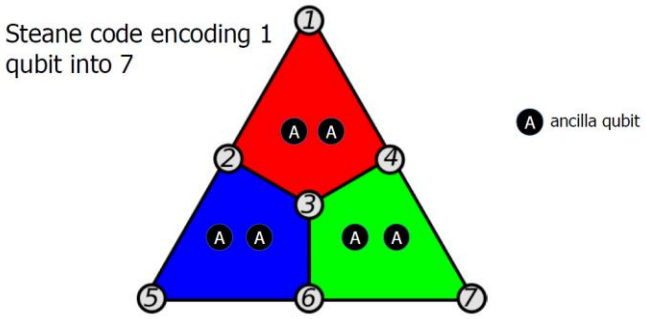
To prevent this, it is necessary to encode the states of the qubits we want to use to compute into *quantum error-correcting codes.* These codes allow errors to be detected and their effects to be reversed, preventing error propagation. In this lecture, Professor Barbara Terhal will introduce a few of the most popular quantum codes currently used. Quantum Error Correction is a theory of how to reverse or undo noise and errors on quantum systems. This theory was developed in the '90, right when the field of quantum computing took off. It was believed that this theory was an absolute necessity for quantum computers to be ever realized. The reason that quantum error correction is important, is that we want quantum computers to do large computations. Such a computation or quantum circuit breaks down into many steps, namely single and two-qubit gates and a measurement of all the qubits at the end.



Barbara Terhal,Quantum Error Correction

When each gate fails to operate properly with some probability, then these failures will accumulate leading to a final answer which cannot be trusted. And the more gates in the circuit, the higher the chance that the final outcome will be essentially garbage. In addition, a qubit which does not undergo any gate, tends to decohere and relax to a fixed, typically thermal, state. So how to make the failure probability per gate low, for example as low as 10^-15? For none of the qubits

under current investigation like the ones that you hear about in this “MOOC”, do we have gates with failure rates of order 10^-15. At best, the failure rate of a single or two-qubit gate is less than, say, 1%. The past 20 years have seen a lot of progress in making better qubits with longer lifetimes and better gates, but to reach 10^-15 by hardware advances just seems implausible. Now, the idea of quantum error correction is to represent quantum information redundantly. For example, instead of a single qubit, we use 7 qubits. And these together constitute one new so-called logical qubit. On this logical qubit we operate as follows. First, we constantly monitor for errors on any of the 7 qubits. We can’t do this by measuring these qubits, why not, otherwise we lose their state. Instead, we will include actually ancilla or helper qubits, which we couple to the data qubits and we will only read out the ancilla qubits. And with this error information we infer what errors have happened. In this way we can undo or reverse these errors. But wait, we can’t just correct any old error. First of all, a qubit can undergo two types of elementary errors, there is bit flip and there are phase flip errors. All other errors on a qubit are linear combinations and/or products of these errors. Now quantum error correction codes are designed such that only errors on small subsets of qubits can be corrected. For example, for the 7 qubits shown previously, an error on any single qubit can be corrected, but not an error on any subset of 2 or more qubits.



Barbara Terhal,Quantum Error Correction

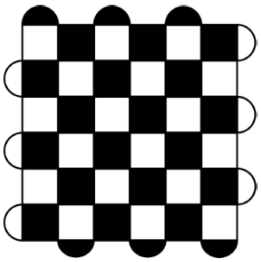
Error correction is then effective when errors on single qubits are much more likely than errors on pairs of qubits. In other words, likely errors will get corrected and the less likely errors will remain. So we improve matters. By adding more redundancy, for example representing a single logical qubit by, say, 49 qubits, one can correct larger subsets of errors. And this means that the failure rate of the logical qubit, which is determined by the errors which do not get corrected, can get really small. With 10.000 qubits it may be possible to have a failure rate of 10^{-15}, particularly with a code called the surface code. An attractive feature of the surface code is that its qubits can be physically placed on a 2D planar chip and only local connections are needed for error correction and logical gates. But what quantum error correcting codes are there and which one is in fact the best? Well, there are many classes of codes and there is no one single answer to which one is best. A code can in general encode k into n qubits. Furthermore, the code has some distance d which means that it can correct almost up to d/2 errors. The Steane code that we have seen has distance 3 and it can correct a single error.

Bit-flip error X: $|0\rangle \rightarrow |1\rangle, |1\rangle \rightarrow |0\rangle$

Phase-flip error Z: $|0\rangle \rightarrow |0\rangle, |1\rangle \rightarrow -|1\rangle$

Barbara Terhal,Quantum Error Correction

Surface Code Family



A bigger code encoding 1 qubit into 49 qubits placed on the lattice sites.

Scalable approach:

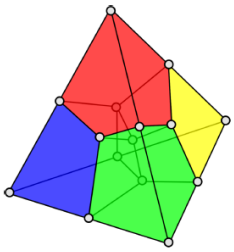
d by d lattice with d^2 qubits, one per lattice site, together representing one logical qubit.

Barbara Terhal,Quantum Error Correction

A Quantum Error Correcting Code

encodes k logical qubits into n physical qubits and has distance d.

A code which encodes 1 qubit into 15 with distance 3



Barbara Terhal,Quantum Error Correction

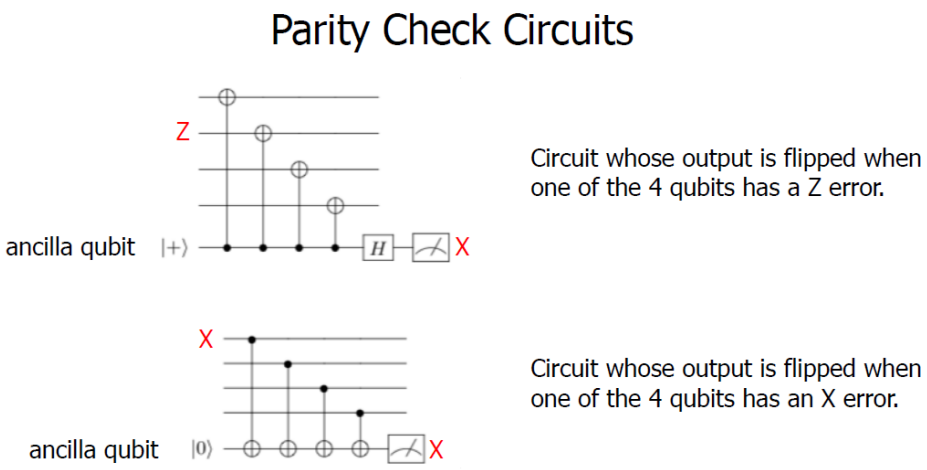
So typically one would like to have a high distance and high k versus n. But more important in practice is in fact, how one gathers error information and how one does that fault-tolerantly. And this is what we will discuss in the next Section.

Main takeaways

- Large-scale computations require much lower logical error rates than those we can produce directly in hardware.
- A variety of encoding schemes exist to protect qubits against external noise, and the effects of imprecise control. More redundancy in the encoding scheme tends to lead to better protection against errors.
- Measuring individual qubits to determine whether an error has occurred will destroy the superposition necessary to maintain a logical state, so measurements which detect errors must be done indirectly, using ancilla qubits.
- There are two fundamental types of errors on qubits: X and Z.

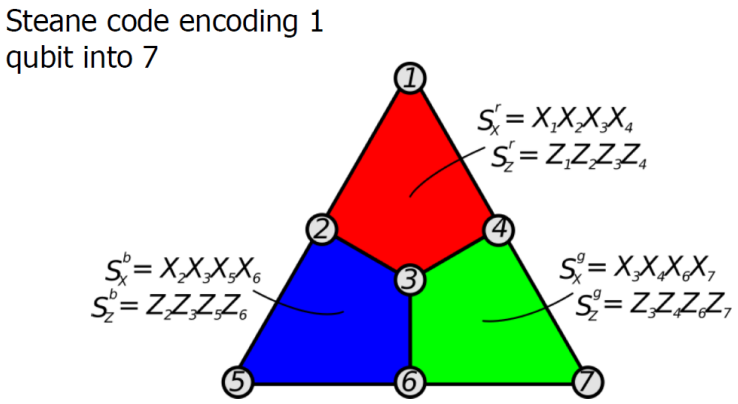
Quantum Error Correction - Fault Tolerance

In order to ensure that our quantum bits remain protected, we have to perform error-detecting measurements very frequently. However, the circuits we use to perform these measurements, if poorly designed, may introduce more errors than the code can correct. Designing circuits which perform *fault-tolerant* quantum error correction is a cutting-edge field of research, and Professor Terhal will focus in on the surface code, the favoured candidate for near-term quantum error correction. This discussion concludes with the introduction of the *fault-tolerant threshold*, the physical error rate which must be reached in order for quantum error correction to be effective.



Barbara Terhal,Quantum Error Correction

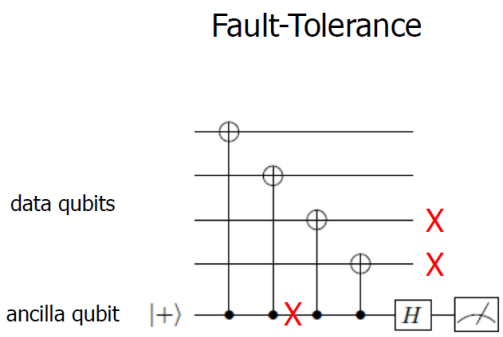
The most common class of quantum error correcting codes are stabilizer codes. For these codes, error information is gathered by executing so-called parity check circuits, as you see on the slide. For some of these stabilizer codes one gathers information about X and Z errors separately. In such a case one uses two types of circuits to gather error information. One circuit is for Z errors, shown at the top, one other circuit for X errors, shown at the bottom. For example, we can have circuits, which look at the parity of a subset of 4 qubits, as you see on the slide. Now, if any of the qubits has a Z error on input, the ancilla qubit will be flipped at the end, heralding the error. You can verify this yourself. Similarly, to detect and correct an X error one can use the circuit at the bottom whose ancilla qubit is flipped on output when any of the incoming qubits has an X error. For the Steane code that we have seen, there are parity checks on three subsets of qubits.



Barbara Terhal,Quantum Error Correction

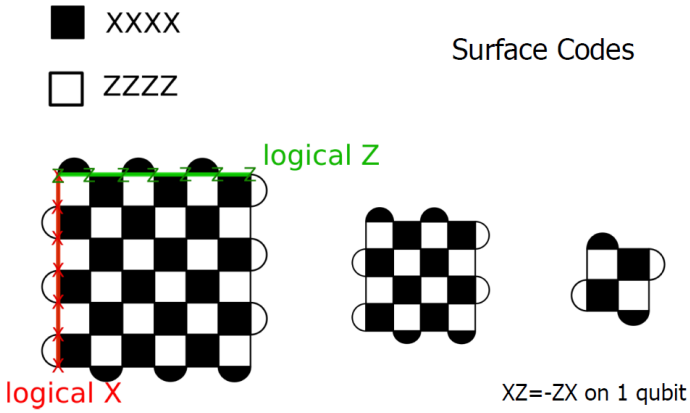
These subsets are colored red, green and blue. For each subset there is a parity check for detecting and correcting X errors and there is a parity check for detecting and correcting Z errors, hence there are 6 checks to be measured in total. A parity check circuit is made out of gates which can fail themselves. For example, an X error can happen on the ancilla qubit after the second CNOT gate. It could be that the CNOT itself produces this error. This X error causes the control of the next 2

CNOT gates to flip. The effect is two extra X-flips on two of the data qubits. If our code was not made to correct these two X errors, then this is really bad. Because it took only 1 fault of that CNOT, to cause 2 errors.



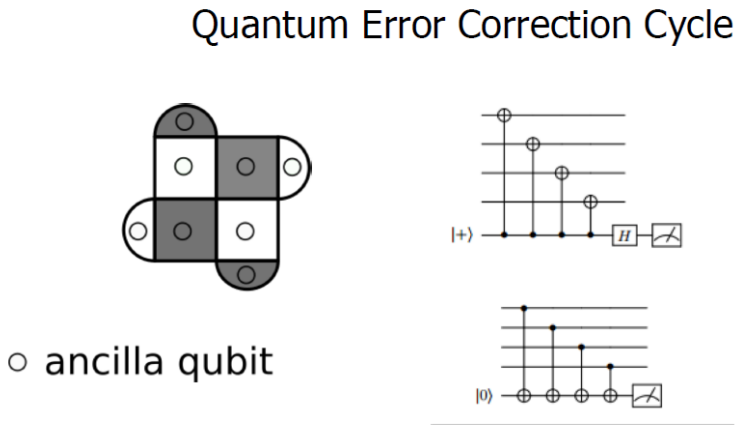
Barbara Terhal,Quantum Error Correction

It didn't look like we were making anything better. We then say that the circuit is in fact not fault-tolerant, it cannot tolerate any faults in its execution. To do quantum error correction and get an improved error rate, we only want to use fault-tolerant circuits. Only with fault-tolerant circuits can we make the logical error rate smaller than the error rates on the constituent qubits. Let us now take a deeper look at the surface codes. Shown on the left is a distance-7 surface code with 49 qubits, there is a qubit on each lattice site – it's not drawn. Each black square represents a X-parity check between 4 qubits detecting a Z error.



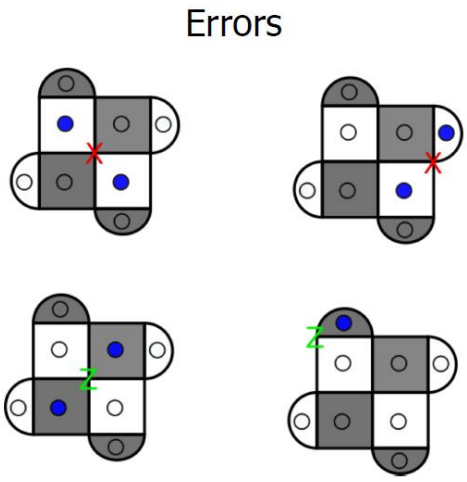
Barbara Terhal,Quantum Error Correction

Each white square represents a Z-parity check which similarly detects X errors on the 4 qubits on which it acts. At the boundary of the lattice the parity checks involve only two qubits. The distance of the code is 7 since the logical operators of the logical qubit act on at least 7 qubits. You see the logical operators on the slide. The defining feature of the logical X and Z operators is that they mutually anti-commute, but at the same time they commute with all the parity checks. You can verify this yourself by using that X and Z anti-commute when they act on the same qubit, otherwise they commute.



Barbara Terhal,Quantum Error Correction

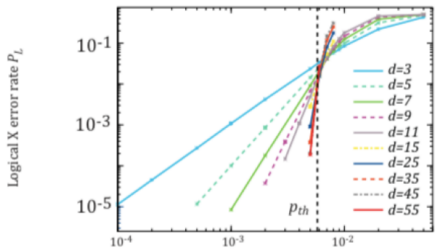
Shown on the very right is the smallest interesting surface code which has 9 qubits and it has distance 3. To execute the parity check circuits for the surface code, we add ancilla qubits to the data qubits, as you see in the slide. The ancilla qubits can be placed in the middle of the plaquettes so that they interact via CNOTs, via controlled NOTs, with the neighboring data qubits. One quantum error correction cycle is the execution of these parity checks. Here you see X and Z errors on the data qubits are detected by the parity check circuits. The ancilla qubits are colored blue when they detect an X or a Z error. Since there can be faults in the parity check circuits themselves, as we discussed, these are not the only errors.



Barbara Terhal,Quantum Error Correction

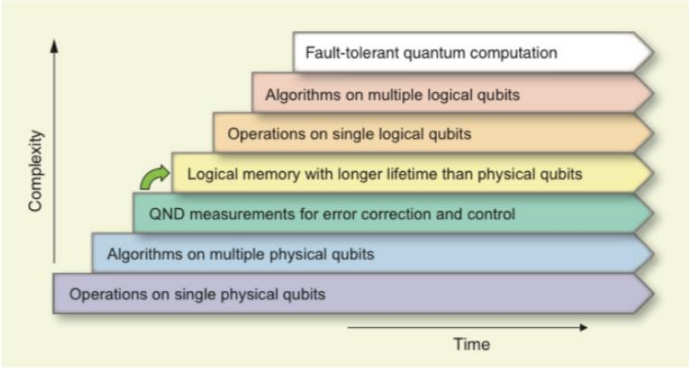
For a small surface code such as the one shown, fault-tolerance of the parity checks can be guaranteed, partially by repeating the parity check measurements. Such repetition is certainly beneficial when faults lead to incorrect parity check outcomes. For example the parity check outcome heralds an error when there was no error, but just a measurement error at the end, or vice versa. It doesn't detect the error. The goal of preserving a qubit in time with the surface code is now to apply many quantum error correction cycles and deduce from the parity check outcomes what errors have taken place. The logical failure probability of the logical qubit is determined by how well we do this. Now we can imagine using larger and larger surface codes with distance 3 and up. If the error rate on each gate is below some critical value, the error correction process or these error correction cycles, improve with larger d. Namely what we see is that the logical failure rate will decrease exponentially in d. However, if the error rate on each gate is above some critical value, then instead error correction becomes worse with larger d. The whole error correction code makes errors more likely rather than less likely. The critical error rate per gate is called the noise threshold. This threshold depends on the method of processing error information and it depends on what types of errors we have. It lies somewhere between 0.5 and 1% for the surface code. Thus in order for hardware to benefit from the use of surface code error correction, it means that idling steps, single-

Threshold



P_{th}=0.57%

Numerical data from: A. Fowler, M. Mariantoni, J.M. Martinis and A. Cleland, *Surface codes: towards practical large scale quantum computation,* Phys. Rev. A 86, 032324 (2012)



From: M.H. Devoret and R. Schoelkopf,
Superconducting Circuits for Quantum Information: an Outlook
Science **339**, 1169 (2013).

qubit and two-qubit gates all have to have an error rate at least below 0.5%. Reaching error rates this low or lower is thus an important target for qubit hardware. We are currently on a long and steep road towards fault-tolerant quantum computing. And on this road there are many steps and milestones. In this “MOOC” you will see some operations and small algorithms implemented on physical qubits. Then a next step is to try and store a logical qubit using quantum error correction. The idea is to store it for a time, which is at least longer than any of its constituent qubits. So far, experiments at several labs have shown how to detect a single error using a four-qubit code and how to store a bit, just a regular classical bit, using three or five qubits. Experiments to implement the smallest 9-qubit surface code are currently being planned. In other words, it has not yet been shown experimentally that with more and more redundancy a logical qubit can have a longer and longer lifetime. Once we store a logical qubit, let’s say that is successful, we can apply schemes that let us do single and two-qubit gates on these logical qubits as well. We have many theoretical schemes available for this. This means that once we have logical qubits we can envision running quantum algorithms on these qubits. And if these qubits, these logical qubits, have low error rates, the quantum algorithms can be long and hence realize a powerful quantum computation.

Further Reading & Viewing:

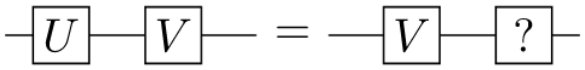
- E. Campbell, B.M. Terhal, C. Vuillot, Roads towards fault-tolerant universal quantum computation, Nature **549**, 172-179 (2017)
- C. Vuillot, Dissecting a tetrahedral color code https://youtu.be/erkeCxQ0-g4
- B.M. Terhal, Quantum Error Correction for Quantum Memories, Reviews of Modern Physics **87**, 307 (2015)
- J. Kelly et al., State preservation by repetitive error detection in a superconducting
- quantum circuit, Nature **519**, 66-69 (2015)
- C. Vuillot, Is error detection helpful on IBM 5Q chips? https://arxiv.org/abs/1705.08957
- M. Takita et al. Experimental demonstration of fault-tolerant state preparation with superconducting qubits , Phys. Rev. Lett. **119**, 180501 (2017)
- B.M. Terhal, Three Lectures on Quantum Error Correction at the IDEA League School onQuantum Information Processing 2015, see https://youtu.be/7FnLMVuVTow

Main takeaways

- To detect errors in most quantum error-correcting codes, it is necessary to measure *stabilisers* using *parity check circuits*, which couple the qubits affected by the stabiliser to one or more nearby ancilla qubits.
- The order in which gates are executed in these circuits, and the arrangement of ancilla qubits can have drastic effects on the logical error rate.
- Every operation which can be performed on a physical qubit can, in principle, be performed on a logical qubit, though there are typically only a few such operations which can be implemented *fault-tolerantly*, introducing only correctable errors with high probability.
- Fault tolerance systems have *threshold error rates*. If the physical operations used in a fault-tolerant quantum error correction protocol have error rates which are smaller than this value, then increasing the redundancy of the underlying code will decrease the logical error rate.

Practice Questions

When analysing quantum computations, it is often necessary to determine the effect of an operation in the middle of a circuit on the final output. To accomplish this, we must complete the following circuit diagram:



Conjugating Operations

Which expression completes the identity above?

Recall that order of operations in circuit diagrams is left-to-right, and right-to-left in matrix products.

- $U^\dagger VU$
- $VUV^\dagger$
- $UVU^\dagger$
- $V^\dagger UV$

Desired Error Rate

To make a conservative estimate of the maximum allowable error probability, let's assume that, if a single logical gate is in error during a large computation, that we'll be able to deduce the correct output by repeating the computation a small number of times. Let's also assume that every logical operation (state preparation, gate, and measurement) has the same logical error rate pL, and that every qubit undergoes an operation at every time step.

If our quantum computation involves n=1000 logical qubits and T=1000 time steps, what pL ensures that we have a single logical error per run, on average?

Required Code Distance

Suppose that a certain quantum computation requires a logical error rate of 10−8 to be successful on average.

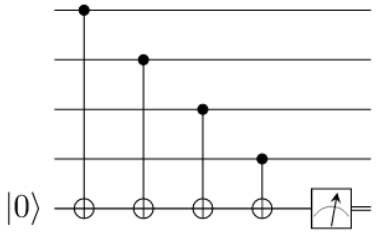
This logical error rate can be approximated using a formula from Fowler, Mariantoni, Martinis, and Cleland (equation 11):

$p_L \sim 0.03(p/p_{th})^{d+1}{}^{2}$,

where p is the physical probability of error, pth is the threshold error rate, and d is the code distance (here assumed to be an odd number).

If (ppth) is 0.1, what code distance is necessary in order to achieve a logical error probability less than 10−8?

Consider the following circuit:



Pauli Error Detection

A Pauli error E on a data qubit will propagate through each of the CNOT gates in the figure above. As discussed earlier, the result of an error E propagating through a unitary gate U is $UEU^{\dagger}$.

Which type, or types, of errors, can be detected by the circuit above?

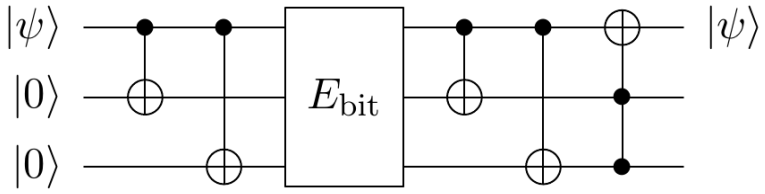
An error is detected if it propagates onto the ancilla qubit, producing a measurement result of $|1\rangle$.

- ☐ X
- ☐ Y
- ☐ Z

Quiz 6: Quantum Error Correction

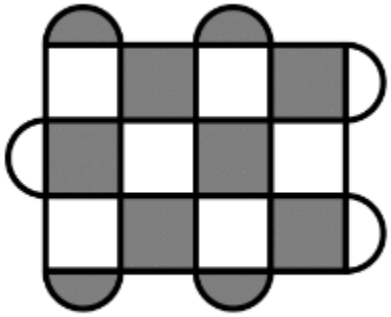
Bit-flip circuit

Unfortunately, we can not directly use the repetition code for quantum bits, because we can not copy (or 'clone') qubits. We can, however, make use of entanglement. Below is a circuit to correct bit flips in a quantum state, based on the circuit that encodes a classical bit into the d=3 repetition code. Here, bit-flip errors occur in the region labelled by Ebit.



4-by-5 surface code

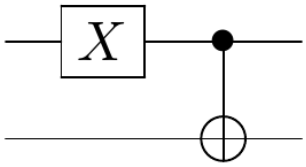
Here, we see the 'checkerboard' diagram for a 4-by-5 surface code:

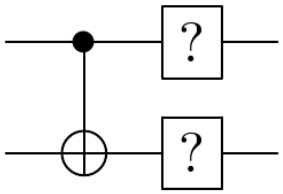


Equivalent circuits

Gates in quantum circuits are noisy, and they also propagate errors which have already occurred from qubit to qubit.

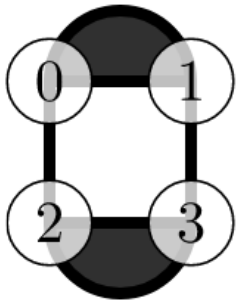
See the simple circuit below:





Distance-2 surface code

Below, we see a surface code with 4 data qubits, 1 logical qubit, and a distance of 2:



Each of the qubits has been labelled with a number. Consider this code when answering the following question.

Learn more

Additional reading & watching

- Dive deeper into the topic of QEC, and try this scientific paper **Quantum Error Correction for Beginners**
- If you prefer watching over reading, you could try this Section: **Mini Crash Course on Quantum Error Correction**

Module 4: Quantum Internet (part 1)

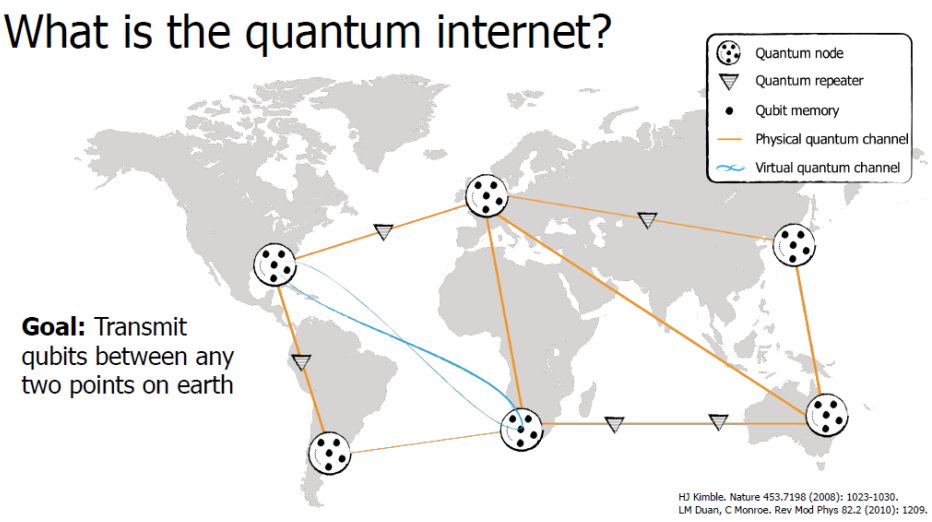
Introduction to Module 4

We have now examined every layer of the stack for building and programming a quantum computer. However, we know from the development of classical computers that computers which can communicate with each other through networks are far more useful than computers which are isolated. This naturally raises the question of how to connect quantum computers to one another. How do we perform a CNOT between two qubits, when they're not even in the same country?

One of the convenient properties of quantum mechanics is that such 'non-local' gates can be performed, as long as we consume a shared entangled state for each non-local gate we want to do. This distribution of entanglement is the fundamental task of a quantum internet. However, this distribution of entanglement does more than enable non-local computation. It can be used to replace many of the communication protocols we use today with faster, or inherently private alternatives. Over the course of the next three lectures and two quizzes, David Elkouss will introduce the tasks that a quantum internet can perform better than its classical counterpart, as well as the components that make up these networks. We hope you enjoy learning about this fascinating new technology.

Tasks for a Quantum Internet

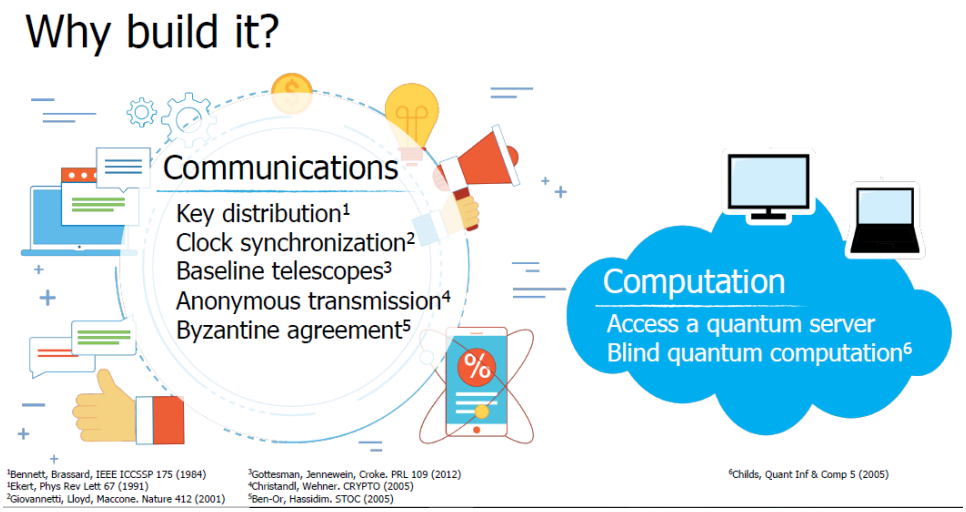
A quantum internet sounds great, but why should we want it? What will be the differences between a classical internet and a quantum internet? In this Section, David explains some of the tasks that a quantum internet will perform. The most commonly known application of a quantum internet is quantum key distribution, though there are many more potential applications, which David will discuss.



David Elkouss,Communication tasks with quantum resources

The internet, has had a tremendous impact on our world. Our long-term vision is to build a matching quantum internet that will operate in parallel to the internet we have today. So what is the quantum internet? Well, it s a network. The nodes will be quantum devices of different computational capabilities. These nodes will be connected via quantum channels, channels that allow the transmission of quantum information. Intermediate nodes, called quantum repeaters, will be placed between adjacent nodes whenever losses are too large. Finally, via entanglement swapping and teleportation it will be possible to create virtual channels between nodes that have never interacted. This network will enable the transmission of quantum information between any two points on earth.

Building a quantum internet will be a tremendous challenge. Why would we want to do it, given the success of the internet we already know? We want to build it, because it will be useful. There are many communication tasks for which quantum resources offer a qualitative advantage. Perhaps the most famous of these tasks is key distribution. Suppose that to distant parties want to exchange a secure key, perhaps to perform encryption with it at a later point or for other purposes. Unfortunately, it is not possible to distribute secure keys between distant parties with classical resources. At least, unless one is willing to believe that some computational problems are untractable or that the computational devices of a potential eavesdropper are limited in certain ways. On the other hand, quantum resources allow to distribute secure keys between distant parties. Key distribution is the best known task, but it is not the only one. We know of better protocols for synchronizing clocks or for extending the baselines of telescopes. Quantum does allow to transmit both classical and quantum information in a completely anonymous way. Quantum also promises advantages for problems in distributed systems. One example is so-called Byzantine agreement. In this task, the objective is to agree in the value of a bit in a coordinated fashion in the present of a subset of faulty and potentially adversarial nodes. Protocols require a number of communication rounds in order to reach agreement. The number of rounds of the best classical protocols roughly scales linearity with the number of users. However, with quantum resources, it is possible to reach agreement with a constant number of rounds independent of the number of users. There are many other communication tasks, notably in the two-party crypto domain. But perhaps surprisingly, quantum communication also finds application in computation. Suppose that we reach the time when the first powerful quantum computers are built. No matter how the technology evolves, at first these quantum computers will not be widespread and will be owned by a few actors: universities or perhaps big corporations. A quantum network would allow to transmit a quantum state from a client, perhaps the partial output of a computation, transmit it to the distant server, let the server finish the computation and get back the resulting state.



David Elkouss,Communication tasks with quantum resources

Quantum also allows, for the following more sophisticated task. It is possible to perform a computation in a remote server in such a way that the server does not learn anything about the state or about the computation performed. This is called blind quantum computation and requires the client to be connected via a quantum channel to either provide some input state or, in other protocols, to perform some simple measurements. These are just some examples of the tasks that we know today. Analogous to the classical internet, we also expect that most applications, and probably the most useful ones are yet to be discovered. In summary, building a quantum internet will be a tremendous scientific and engineering challenge. We want to undertake it, because we believe it will also be tremendously useful.

Main takeaways

- Entanglement swapping and teleportation allow the creation of virtual channels between nodes that have never interacted.
- Quantum repeaters can be placed between nodes to compensate for transmission losses in quantum channels.
- Quantum key distribution, clock synchronization and problem-solving in distributed systems are some of the ideal tasks for a quantum network.

Practice Questions

Many of the tasks performed by a quantum internet require us to process entangled states which are received by a network component. For this purpose, we often use *Bell basis measurement*, which translates the maximally entangled states of the Bell basis into computational states which can then be read out more simply. In ket notation, this basis transformation is as follows:

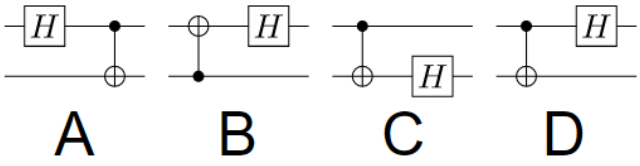
$$\left|\phi^+\right\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \mapsto |00\rangle$$

$$\left|\psi^+\right\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \mapsto |01\rangle$$

$$\left|\phi^-\right\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \mapsto |10\rangle$$

$$\left|\psi^-\right\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \mapsto |11\rangle$$

Consider the following four circuits:



Bell Basis Measurement

Which of the four circuits above performs the Bell basis measurement?

- A
- B
- C
- D

Quantum networks are essentially machines which produce shared entanglement. One of the most fundamental applications for shared entanglement is teleportation, in which entanglement is consumed in order to transmit a quantum state. Consider circuit D above:

Teleportation

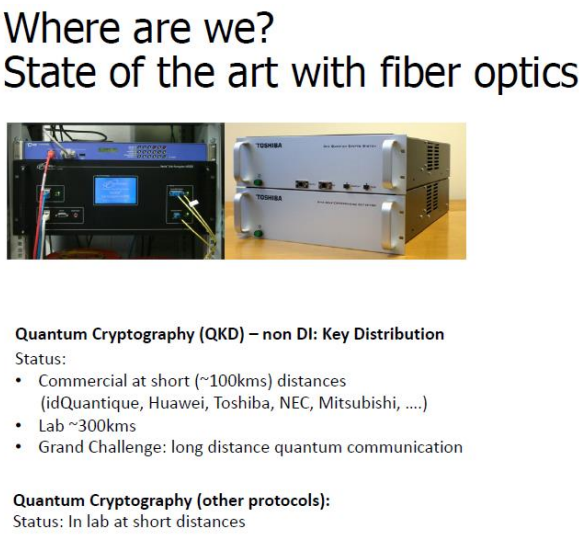
Before running the circuit above, Alice possesses a qubit in a state $|\psi\rangle$, and half of a Bell pair, of which Bob has the other half. We decompose $|\psi\rangle$ in the computational basis as $\alpha|0\rangle+\beta|1\rangle$.

If this circuit is carried out, and the measurement outcome is 10, what is Bob's state at the end?

- $\alpha|0\rangle+\beta|1\rangle$
- $\alpha|1\rangle+\beta|0\rangle$
- $\alpha|0\rangle-\beta|1\rangle$
- $\alpha|1\rangle-\beta|0\rangle$

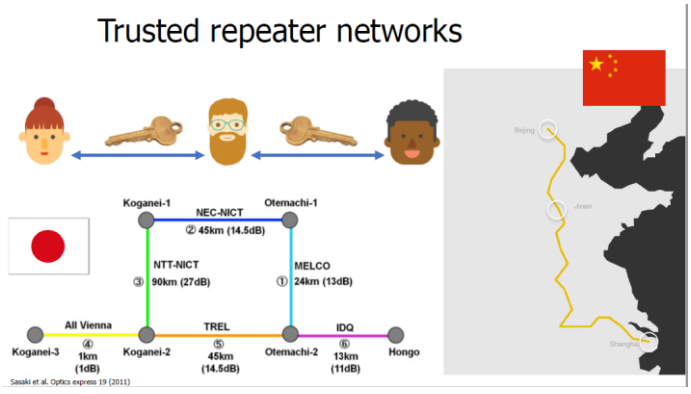
Quantum communication

Quantum computers, networked together using shared entanglement, permit us to carry out a wide variety of new communication protocols. But how well can these networks be implemented, using today's technology? In this Section, David will introduce *the quantum repeater*, a crucial ingredient in the realization of quantum networks over long distance. He will then outline some of the differences between ground-based and space-based quantum networks, the two types which are the subject of current research.



David Elkouss,Towards long distance quantum,communications

There is a host of communication tasks were quantum resources offer an advantage with respect to classical ones. The future quantum internet, were any two nodes in the globe can exchange quantum information and, at each node sits a full-fledged powerful quantum computer, will allow to implement all these tasks. But where do we stand now? What can we do today? Most experimental efforts have been directed towards the implementation of quantum key distribution. However, the situation is a little bit different depending on the physical medium chosen to transport quantum information. If we consider fiber optical communications, it is already possible to acquire devices for quantum key distribution from a number of commercial companies. With these devices, it is possible to distribute secure keys between parties that are separated by a distance of about 100 kilometers. With lab equipment, we can go a little further, perhaps to up to 300 kilometers. However, for fundamental reasons that we will discuss later it is not possible to go beyond this distance with a direct connection. It is possible, in principle, to go beyond this distance by collocating intermediate devices that act as quantum repeaters.

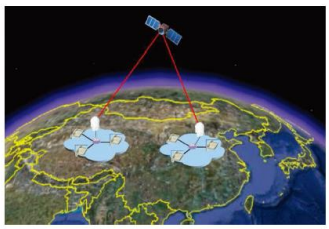


David Elkouss,Towards long distance quantum,communications

While this is possible in principle, this type of functionality has not yet demonstrated in experiment. There are a handful of other protocols, mostly cryptographic, that have already been implemented in the lab at short distances. Some of these have very similar hardware requirements as quantum key distribution. In consequence, we could expect that they can

also be implemented over distances of a few hundred kilometers in the near future. Until we have quantum repeaters, there is a temporary solution called the trusted repeater. This is a solution that only works for the task of key distribution. The idea is very simple, suppose that Alice and Bob want to distribute keys over 400 kilometers. Since they cannot bridge this distance with quantum key distribution devices, they call their friend Charlie for help. Charlie's lab is collocated just between Alice and Bob, at 200 kilometers of distance from each.

Where are we?
State of the art with free-space communications



Entanglement over a distance of > 1200km via satellite

Pan Group, Science, July 2017

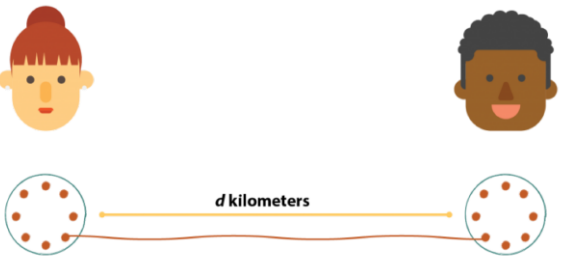
David Elkouss,Towards long distance quantum,communications

Then Alice distributes a key with Charlie using standard quantum key distribution and Bob also distributes a key with Charlie. Finally Charlie makes public the sum of the two keys. It turns out, that provided that Charlie is trusted and does not leak any information, Alice and Bob obtain fully secret keys. This paradigm has been used since the early 2000s to distribute keys over long distances. Two of the most famous examples are in the slide. On your left, a metropolitan area network with trusted devices in Tokyo. On your right, an extremely long chain of trusted repeaters connecting the cities of Beijing and Shanghai. In the case of free space communications, the characteristics of fiber optical channels that do not allow to reach long distances do not apply to free space. In principle, it is possible to perform quantum communication tasks over long distances. The state of the art is shaped by a series of amazing experiments that the Pan group performed around 2017. They managed to distribute post-selected entanglement over a distance of 1200 kilometers. In another experiment, the group also managed to distribute secure keys between ground and satellite and finally, with the help of the satellite as a trusted repeater, they distributed secure keys between cities in Austria and China. The need for a trusted repeater in this setup stems from the fact that the satellite does not see Austria and China at the same time. Hence, in the absence of a powerful quantum repeater that stores the quantum information sent from one of the parties and transmits it to the other party at a later time, it is necessary to rely on the paradigm of trusted repeaters. Let us go back to fiber optical channels and think from scratch why it is so difficult to reach long distances. Now, I want to discuss what are some of the challenges for establishing long-distance entanglement and a very idealized solution. Let us consider that two distant parties, Alice and Bob, are connected via a quantum channel. Since I am a theorist, we also imagine that Alice and Bob have noise-free quantum memories available to them and, even more, they can transfer qubits from their memories to the input of the channel and store incoming qubits into the memory without any error or decoherence.

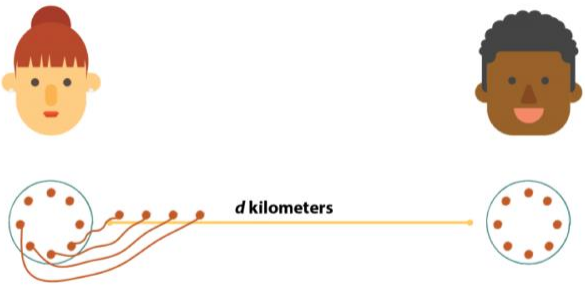
Towards long distance

Transmission probability: $p(d) = 10^{-d/\tau}$

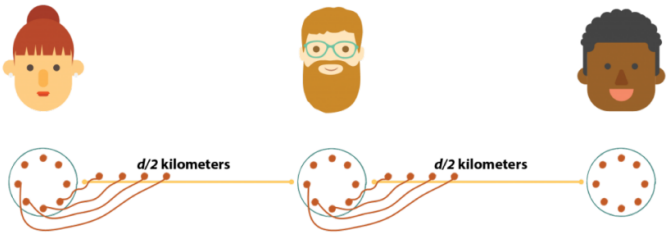
David Elkouss,Towards long distance quantum,communications



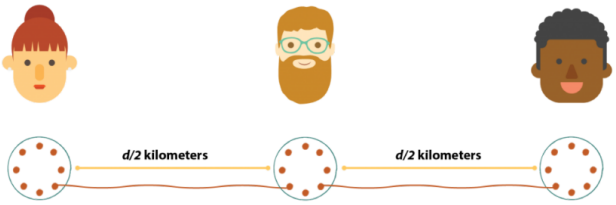
David Elkouss,Towards long distance quantum,communications



David Elkouss,Towards long distance quantum,communications

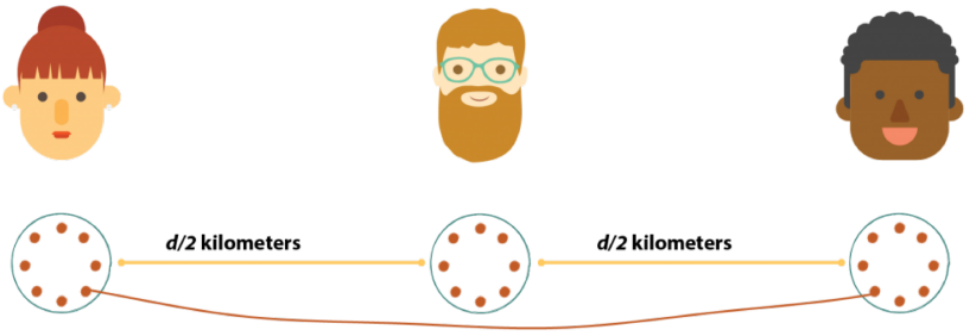


David Elkouss,Towards long distance quantum,communications



David Elkouss,Towards long distance quantum,communications

An initial strategy for establishing entanglement between them could be as follows. Alice, prepares an entangled state locally between two qubits in her memory. Then she takes one of the two qubits and encodes it into a degree of freedom of a photon and sends the photon to Bob via the fiber optical cable. When Bob receives the photon he stores it in his quantum memory. Now Alice and Bob share entanglement. So, you might be wondering, what is the catch? The catch is that Bob might never receive anything. The probability that Bob receives the photon scales exponentially as a function of the cable or channel length. More precisely, let d be the length of the cable, the probability is exactly: P(d)=10^{-\frac{d}{\tau}} where \tau is a parameter, called the attenuation length, that depends on the type of fiber optical cable. Let us go over a couple of examples to understand the implications of this exponential decay. Let us assume that the attenuation length is 50 kilometers. Then, first, we can observe that every 50 kilometers, the probability of getting a photon from Alice to Bob gets divided by 10. It is 0.1 after 50 kilometers, 0.01 after 100 kilometers, etcetera. For instance, if we try to connect Delft and Madrid which are at an approximate distance of 1500 kilometers, the probability of getting one photon to the other side would be 10^{-30}. Of course, we can partially compensate these photon losses by repeating the process many, many, many times. However, a quick calculation reveals that on average Alice and Bob need a number of attempts equal to the inverse of the probability to transmit one photon. Hence, in the case of the Delft-Madrid link, Alice and Bob would need 10^{30} attempts on average. Let us transform this idea of repeating many times into a protocol for entanglement distribution that we call protocol A. In this protocol, Bob sends Alice a message indicating whether he received the photon or not. In the case it did not arrive, Alice resets her memory and tries again. Since these messages cannot travel faster than light, in the Delft-Madrid example Alice and Bob would need to wait at least 3 x 10^{21} years on average to get their first entangled pair. Protocol A can be improved by multiplexing. That is, Alice does not need to wait for Bob's message to try again, she can generate a new pair as soon as the first photon is sent to Bob. But even if Alice can prepare entangled pairs, and store the corresponding qubits at high rates, long distance communications are still out of reach. Let us call the rate at which Alice prepares entangled pairs the repetition rate. For instance, if the repetition rate is 1GHz, Alice and Bob would still only generate entangled pairs at a rate of roughly one pair every 3 x 10^{14} years. Still not very impressive. Let us call this protocol, protocol B. The solution to this problem is to place intermediate devices between Alice and Bob. We call them quantum repeaters. These devices basically exploit the teleportation trick to induce channels with larger attenuation lengths. Let us take a glance at how they work. Let us assume that we place a third party at half the distance between Alice and Bob. This party, we can call him Charlie, also lives in the idealized world I described above. That is Charlie has a perfect quantum memory and can transfer qubits from his memory to the channel and vice



David Elkouss,Towards long distance quantum,communications

versa without noise or losses. So how do we benefit from the presence of Charlie? Consider the following protocol. Simultaneously Alice and Charlie, and Charlie and Bob implement protocol B over an optical fiber cable of half the total distance. The rate at which these protocols produce entangled pairs over half the distance is 10^{d/(2\tau)} times the repetition rate. The protocols in each link are asynchronous. This implies, for instance, than when the first entangled pair is ready at one of the links, say Alice-Charlie, the other will have nothing. Since we have assumed perfect memories, this is not a problem. Since the link Alice-Charlie keeps the pair stored and continues producing additional pairs. Once the other link, Charlie-Bob, produces the first pair, Charlie uses the entangled pair with Bob to teleport his half of the entangled pair with Alice. Alice and Bob end with an entangled pair as desired. The rate at which this repeater protocol produces entangled pairs is equal to the rate at which the short links produce entangled pairs, that is 10^{d/(2\lambda)} times the repetition rate. This rate is equivalent to the one that we would obtain if the attenuation distance had doubled or if the length of the link had halved. This idea can obviously be generalized to a larger number of intermediate stations. We can place repeaters between Alice and Charlie and between Charlie and Bob. In this idealized world, combining multiplexing, that is protocol B, with arbitrarily many quantum repeaters between Alice and Bob it is possible to completely eliminate the problem of losses in optical fiber. In summary quantum repeaters can bridge long distances. In a sense they allow to implement channels with larger attenuation distance. We have not discussed the effects of noise in this procedure. The preparation of entanglement corresponds with a sequence of quantum gates that are or might be noisy. Then qubits are sent through a quantum channel that is also subject to noise. And finally since the protocols we described are asynchronous, the parties need to wait for particular events to occur. In the meantime the states in their memories decohere. Similar to error correction, if one is willing to sacrifice the rate at which the task is performed, it is possible to take action against these noisy processes. In the case of entanglement, it can be distilled. The idea is that one takes several noisy entangled pairs and combines them into a smaller number of higher quality entangled pairs. The problem gets more complicated when there is more than one intermediate repeater. The base step is to distribute entanglement between adjacent nodes. However, there are several options to proceed from that base step. One can distill highly entangled states before proceeding, or one could proceed directly and distill after entanglement has been swapped. It is possible also to take different decisions for different nodes. At present, it is unclear what combinations work best. This is a very interesting open problem that needs to be tackled in order to distributed entanglement over long distances.

Summary

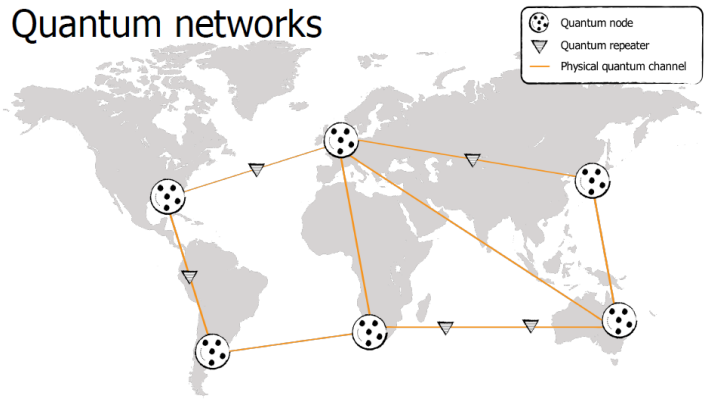
- Repeaters can bridge long distances
- Entanglement distillation
- Difficult optimization problem

Main takeaways

- The *quantum repeater* is a small quantum device which can convert multiple entangled states distributed over short distances into a single entangled state distributed over a large distance.
- This is essential, because otherwise, the probability of Bob receiving a quantum state sent by Alice would decay exponentially as a function of the distance between them.
- In the absence of quantum repeaters, a classical device can be used in repeating signals, as long as the device is *trusted*.
- We can use *entanglement distillation* to combat the effects of noise in quantum channels, by consuming multiple noisy entangled states in order to produce a single less noisy state.

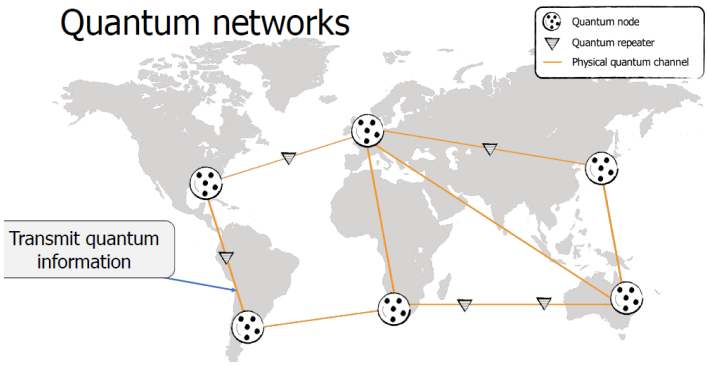
Elements of a Quantum Internet

Now that you've seen the possibilities the quantum internet has to offer, you can ask the question: So what do we actually *need* to build a quantum internet? In this Section, David will explain the elements needed to construct an internet, and how we can realize those elements.



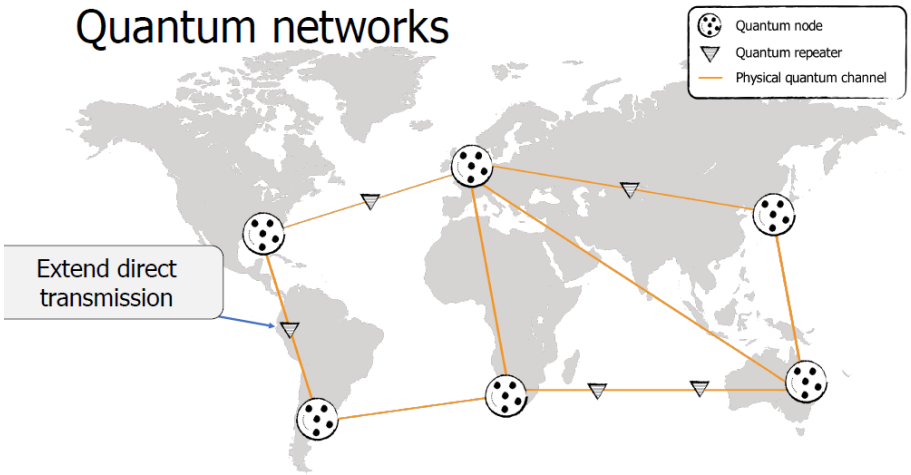
David Elkouss,Elements of a quantum network

Let us now describe the basic elements of a quantum network. We have the physical medium that enables the transmission of quantum information, that is, the quantum channel. We have the device that allows to extend the reach of direct connection links, that is, the quantum repeater. And finally we the device in charge of performing the quantum information processing for the user, we call it the end node or quantum node. The role of the channel is to transmit quantum information between adjacent nodes, be it end-nodes or quantum repeaters. Although there are some intriguing alternatives like cargo ships have been proposed, in practice two types of channels are used: free-space channels and fiber-optical channels.



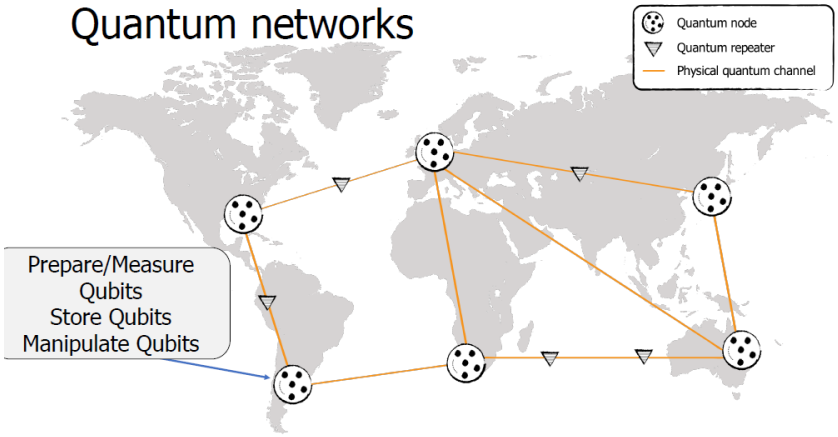
David Elkouss,Elements of a quantum network

Free-space channels have smaller losses but are normally restricted to line of sight scenarios and can be severely affected by daylight. Fiber optical channels have high losses, transmissivity, as we have already said, scales exponentially with distance. But they have the huge advantage that they are already deployed. Quantum information can be transmitted over the very same fiber-optical channels used by the telecom companies. The quantum internet, might very well use both types of channels for different situations. In any case, physical channels always introduce a certain amount of loss and noise. The former, mostly affects the rate at which protocols can be performed. The latter needs to be overcome either via error correction or via entanglement distillation. Both impose additional requirements on the quantum information processing capabilities of the nodes. That is, the necessary quality of the quantum memory and gates at the end nodes depend on the noise in the channel.



David Elkouss,Elements of a quantum network

Quantum repeaters are necessary both for ground as for satellite communications. Notably, in ground communications, the transmissivity of fiber optical channels scales exponentially with distance. Without quantum repeaters, end points in the network cannot exchange quantum information. In consequence, there is no truly quantum protocol that can be performed. The partial exception is quantum key distribution, if one is willing to trust all intermediate nodes. One might naively expect repeaters to be a simple device that boosts power or amplifies the signal. However, quantum information cannot be copied or naively amplified. Hence, quantum repeaters rely on different paradigm than their classical analogues. The quantum internet of the future will have a full-fledged quantum computer seating at each node. And, indeed, we already know of applications that benefit from these extreme resources.



David Elkouss,Elements of a quantum network

However, most applications of the quantum internet do not require a quantum computer at each node. In fact, many applications require resources as simple as the capability to prepare a small number of single qubit states and the ability to measure in two different basis. Errors in preparation, noise in the channel, are dealt without quantum error correction but by purely classical post-processing. This means that, at least for these applications, the threshold fidelities and coherences required are much lower than those necessary for performing interesting computations with a quantum computer. The reason why Quantum Internet protocols can outperform classical communication with such relatively modest resources is due to the fact that their advantages rely solely on inherently quantum properties such as quantum entanglement. Which can be exploited already with very few qubits. It is fundamentally impossible to replicate the properties of entanglement using classical communication. In contrast, a quantum computer must feature more qubits than can be simulated on a classical computer in order to offer an advantage.

Main takeaways

- A quantum network consists of three main building blocks.
- The quantum channel is the physical medium that enables to transmit quantum bits. Two types of channels are commonly used: free-space channels and fiber-optical channels.
- Quantum repeaters are fundamental to faithfully connect nodes distant from each other.
- We call the end node (or quantum node) the device in charge of performing the necessary tasks of quantum information processing.

Practice Quiz Copy - The Quantum Internet

Transmission Losses in Atmosphere

In this question, we consider a simplified modele of transmission loss in the atmosphere. We assume that, when a photon is travelling though the atmosphere, whether at sea level, or in "free space", it can be absorbed if it interacts with an atom or molecule in the air. This implies that the fraction of photons which successfully reach a target will decay exponentially with length:

$f_T = 10^{-d\tau}$,

where τ is a characteristic length scale directly proportional to atmospheric density. At sea level, this density is approximately $1 kgm^3$. At an altitude of $\sim 20km$, however, this density decreases to $\sim 0.1 kgm^3$.

If a given QKD system could function acceptably at a range of 100km at sea level, at what distance (in kilometres) would it achieve the same functionality at 20km altitude?

Quantum Repeater Success Probability

Suppose that Alice and Bob are attempting to generate shared entanglement using $n-1$ quantum repeaters, which are evenly spaced between them, so that n Bell pairs are generated at short range, instead of Alice and Bob. Transmission losses between each pair result in a success probability of $10^{-d n \tau}$ for each short-range entanglement generation attempt.

If all n attempts must succeed simultaneously in order to generate entanglement, what is the overall success probability?

- 0
- $10^{-d\tau}$
- $10^{-dn\tau}$
- $n \times 10^{-d\tau}$

Telecommunication Fibre Optic Cables

As mentioned in the Section, researchers are attempting to use pre-existing fibre optic cables to distribute entangled photon pairs. However, these cables are typically used to transmit classical information, contained in pulses of laser light which contain large numbers of photons. This results in a large increase in the probability of successful transmission.

Why do QKD schemes use single photons, rather than these large pulses?

- Single photons are easier for physicists to understand.
- Single photon detectors are extremely sensitive, and would be destroyed by such a pulse.
- Eve can measure the travelling pulse to determine the state being sent.

Learn more

- Check out this article to know more about the launch of the first quantum satellite.
- Take a look at the original report on BB84, the first quantum cryptographic protocol.

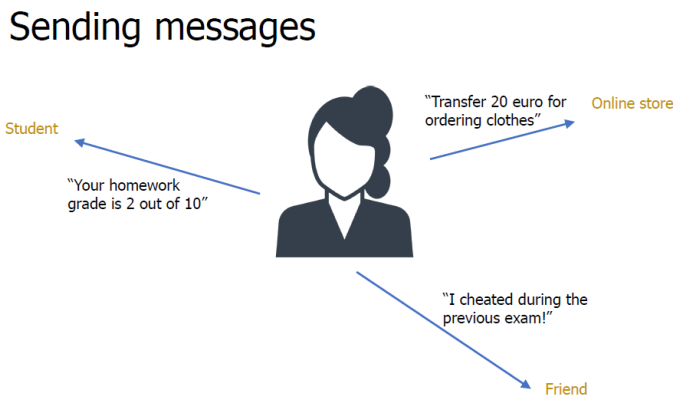
Module 5: Quantum Internet (part 2)

Introduction to Module 5

We have begun to study the quantum internet, discussing its importance, as well as its implementation in the near term. This discussion has included a high-level overview of the protocols we can implement on a quantum network, as well as the central role that quantum repeaters play in distributing entanglement, which is a fundamental task for the quantum internet. In this module, we begin by taking a step back. Tim Coopmans will introduce the "one-time pad", a simple yet powerful cryptographic protocol which uses shared randomness to achieve total privacy. He follows this with a discussion of the BB84 protocol, which generates shared randomness using the unique features of quantum mechanics. Afterwards, Filip Rozpedek will then discuss the difficulties in creating ideal entangled states between nodes. He introduces *entanglement distillation*, which can be used to filter out impure states in entanglement distribution processes. This module concludes with an opportunity for you to work out a simple example of entanglement distillation for yourself. We hope you enjoy this detailed treatment of quantum networks.

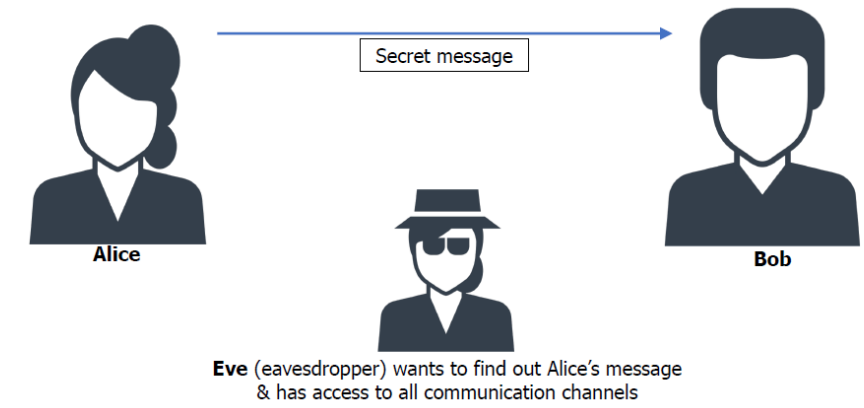
Secure communication using the one-time pad

In this course, we have focused mostly on the truly quantum aspects of quantum computing and networking. In this lecture, we make an exception. Here, Tim Coopmans will introduce the *one-time pad*, an extremely simple, and very secure *classical* cryptographic protocol, which is not used in practice, due to the inconvenience of creating the shared randomness required to encrypt messages. We will see in the later lectures of this module exactly how this necessary shared randomness can be generated using quantum mechanics.



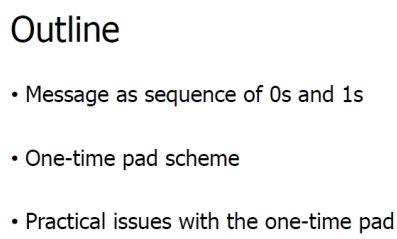
Tim Coopmans,Secure communication using the,one-time pad scheme

Image that in an online store, you buy a new t-shirt. The t-shirt costs 20 euro, so through your bank account; you transfer the money to the online store. You specifically want the 20-euro to go to the online store, since if they don't receive the money, they will not send you the t-shirt. Also, you want to wire just the 20-euro, not less and definitely not more either. Maybe even you don't want anyone else to find out you bought a new t-shirt, because you told your friends you wouldn't spend more money on clothes this month. So in this transaction between you and the online store, you want two things: (1) it should be secret: only you and the online store should know about it, and possibly your banks. (2) no-one should be able to alter the message into, for example, transferring 200 euro instead of 20. There are many more situations in real life where you want the message to remain either private or intact, or both. For example, when your teacher e-mails you your grade, which you are actually embarrassed about. Or when your friend tells you a secret that no one else is supposed to know. The bottom line here is that when sending a message, in most cases in real life we want to do so securely. In this Section, we will consider the following scenario. Alice, here on the left, wants to send a secret message to Bob, on the right. Unfortunately, Alice and Bob are far away from each other and only have means to communicate that can be intercepted by an eavesdropper, usually called Eve. For example, if Alice and Bob use computers with a cable in between, then Eve could be tapping the cable. Or, if Alice and Bob use regular paper mail to communicate, Eve could be the postman



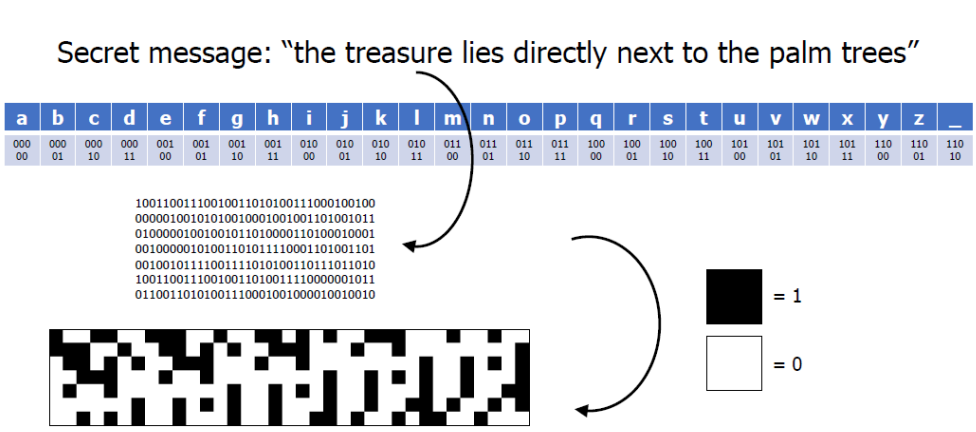
Tim Coopmans,Secure communication using the,one-time pad scheme

who can secretly open and close their letters. The main question is: how can Alice and Bob communicate such that Eve cannot read their messages? One way to solve this problem is by using a scheme that allows us to send messages securely, which is the so-called one-time pad scheme. Disclaimer: in this entire Section, we will not deal with any concepts from quantum computing or quantum communication, but we will use the concepts explained here in the next Section, which is about secure communication using quantum devices. In this Section, we will do three things: First, you will see how to encode a textual message in zeroes and ones. Then, we will see how the one-time pad scheme for secure communication works. And third, we will note some issues with the one-time pad for practical use. First, let us translate a piece of text to a sequence of zeroes and ones. Suppose we have the following secret message: "the treasure lies directly next to the palm trees". For every letter, we have now made a sequence of five zeroes or ones. For example, every letter a, we encode using the sequence zero-zero-zero-zero-zero, and the letter t as one-zero-zero-one-one.



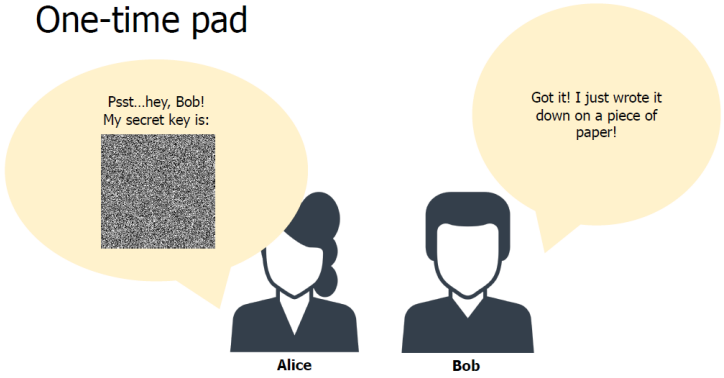
Tim Coopmans,Secure communication using the,one-time pad scheme

By using this 'dictionary', we can write every letter in the secret message as a block of five digits. By gluing those blocks together, we have now translated the message into just zeroes and ones. We can reverse this procedure by first cutting up the sequence into blocks of five digits, and then decoding each of the blocks using the dictionary in the reversed direction. We can take this a step further and translate the message into a black-and-white image. We do so by writing every one as a black pixel and every zero as a white pixel, and this results in the image shown below. Now that we know how to encode a piece of text in bits or in an image, we are ready to learn how the one-time pad scheme works. The first step in the one-time pad scheme is that Alice creates a secret key. Here depicted on the left as a black-and-white pixel image. The key should be chosen completely random in order to guarantee security of the scheme. Alice then goes to Bob and Bob copies the secret key, for example on a piece of paper or stores it digitally on his computer. Now Alice and Bob go their own way each, and later on Alice wishes to send a secret message to Bob. The secret message is 'Hello there', here depicted on the left of the screen as an image. In the one-time pad scheme, Alice takes her key and takes the so-called bitwise exclusive OR or bitwise XOR of her two images. This means that for every pair of pixels, one from the message and one from the key, she creates a new pixel which is: - white if the two original pixels had the same color; -



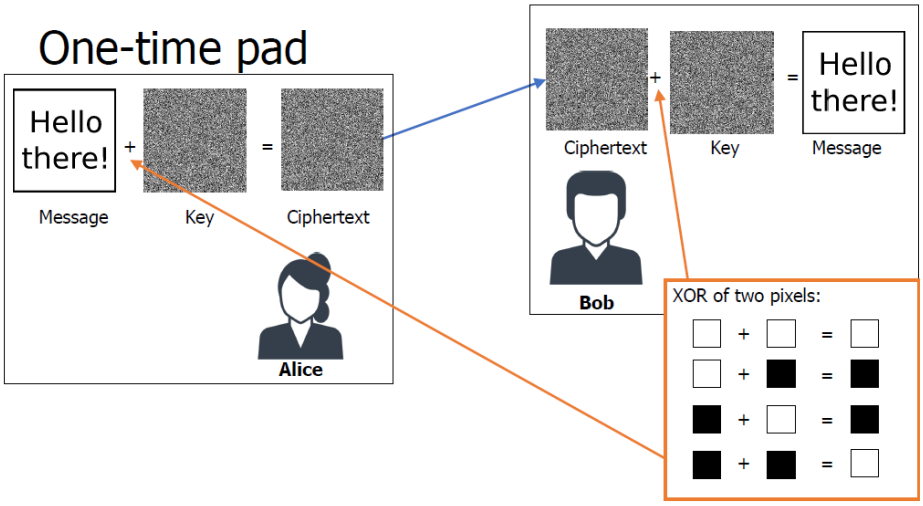
Tim Coopmans,Secure communication using the,one-time pad scheme

and black if they were different. The new image she creates this way is called the ciphertext. This is the image she will send to Bob. Bob then receives the ciphertext and does exactly the same as Alice did. He takes the pixelwise XOR of the ciphertext and the key, the key which he had copied from Alice beforehand. And if you take the XOR of these two message out comes the original message. It might take a bit of thinking to understand why you get the original image back if you take the pixelwise XOR with the key twice.



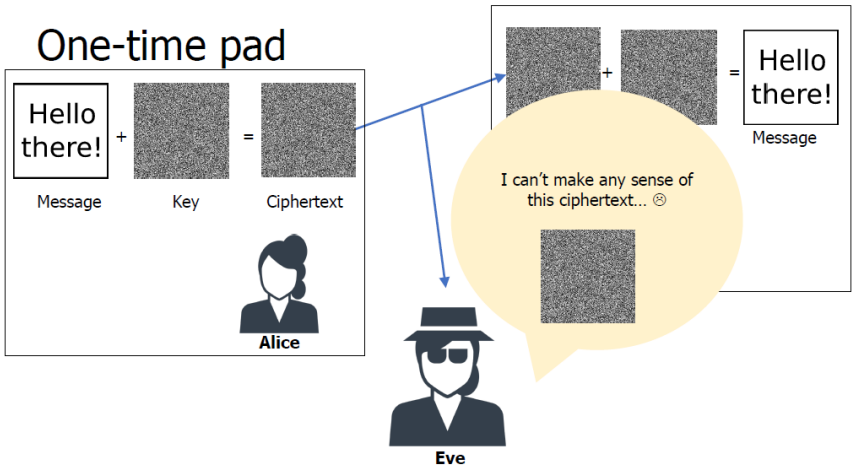
Tim Coopmans,Secure communication using the,one-time pad scheme

I encourage you to try to understand why this is the case indeed. The only thing we now need to think about is Eve, who has access to all communication channels between Alice and Bob. In particular, Eve can intercept the ciphertext, but since the key that Alice used was completely random, the ciphertext also looks completely random to Eve, who does not possess the key. Using this one-time key, Alice has thus sent her message securely to Bob. You probably already guessed from the name that a one-time key may only be used once. Let us see with an example why this is the case. Suppose that Alice wants to send a second message to Bob and uses exactly the same key as she had before. She goes through the same procedure with the pixelwise XOR of the key and the message and thus produces a second ciphertext, which she also sends to Bob. Since Eve has access to all communication channels between Alice and Bob, Eve also intercepts this message. But now the situation is very different: now Eve has two ciphertexts. And by taking the pixelwise XOR of the two ciphertexts, she will get the pixelwise XOR of the two messages, which Eve can now suddenly both read. Again, it might take some thinking why the XOR of the two ciphertexts results in the XOR of the two messages. The underlying idea here is that adding the two ciphertexts is an addition of the first message, the key, the second message, and the key again. Taking the XOR of an image with itself yields a completely white image, so we could also say the 'key drops out'. The result is thus the XOR of the two messages. I encourage you to try take a piece of paper, and try to work out that the XOR of two ciphertexts indeed equals the XOR of the messages. So we have seen that the one-time pad scheme can be used to send

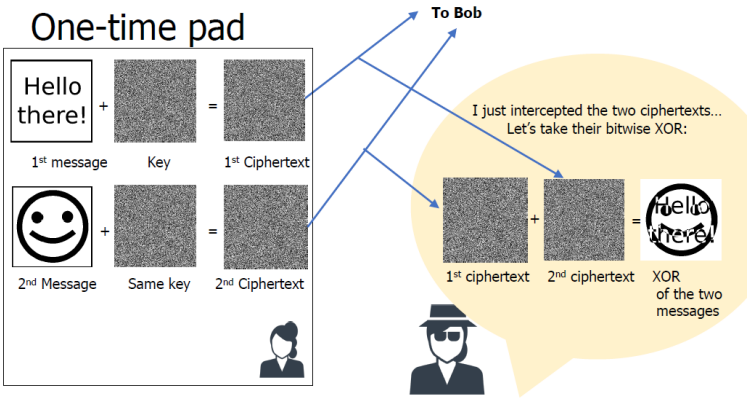


Tim Coopmans,Secure communication using the,one-time pad scheme

a single message in a completely secure way. You might wonder if the one-time pad is used a lot in today's world, where we use e-mail, bank transactions, and other digital means of communication almost every day. The answer is no, because the one-time pad has a few practical issues: First, we have just seen that it may only be used once if you want security to be guaranteed. So every time you would want to send a private message to a friend, you need to establish a fresh secret key. Second, the key must be as long as the message itself.

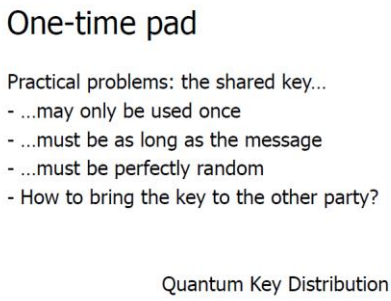


Tim Coopmans,Secure communication using the,one-time pad scheme



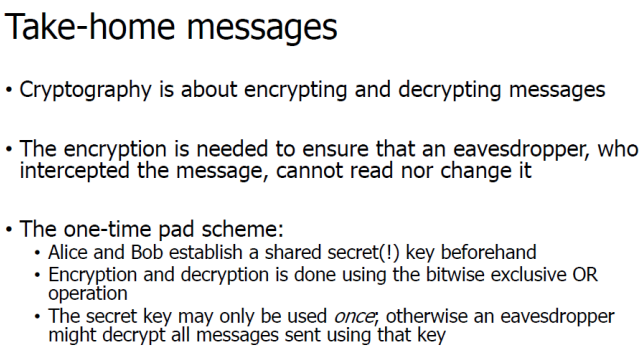
Tim Coopmans,Secure communication using the,one-time pad scheme

This can become impractical if you watch a movie online, which quickly reaches to a gigabyte of data. And then, in practice, perfect randomness is hard to produce. And maybe most importantly: how do you bring the key to the other party? You don't want to go to the physical location of your bank every time you want to transfer money online. There are several ways to overcome these problems, mostly by simply using different schemes, which have disadvantages that the one-time pad scheme does not have.



Tim Coopmans,Secure communication using the,one-time pad scheme

But there is one way where we can still use the one-time pad, and that is by the use of Quantum Key Distribution, which will be the topic of the next Section. Let us briefly summarize the contents of this Section. In cryptography, we deal with the encryption and decryption of secret messages. We cannot just send the message itself, since an eavesdropper might read it, or even worse: tamper with it. We have seen the one-time pad scheme, in which Alice and Bob use the XOR operation to encrypt and decrypt. Lastly, the name comes from the fact that the key may only be used once if security against the eavesdropper needs to be guaranteed.



Tim Coopmans,Secure communication using the,one-time pad scheme

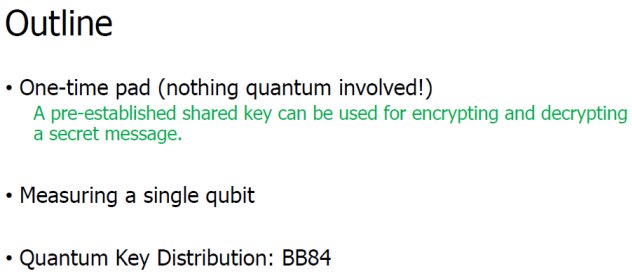
Main takeaways

- Cryptography is the study of methods to transmit messages through channels which an eavesdropper can access, in such a way that the eavesdropper cannot read them.
- There is a simple, classical scheme, called the one-time pad, which provides a sender and receiver (Alice and Bob) with a means of transmitting a secret message, given a string of random zeroes and ones which is the same size as the (binary-encoded) message.
- To use the one-time pad, Alice simply adds the random bits to the message, and transmits the sum to Bob, who subtracts his copy of the random bits (both of these operations are equivalent to XOR when the messages are binary).
- This cryptographic scheme can only fail if Eve gains access to the pad, or the same pad is used more than once.

Quantum Key Distribution

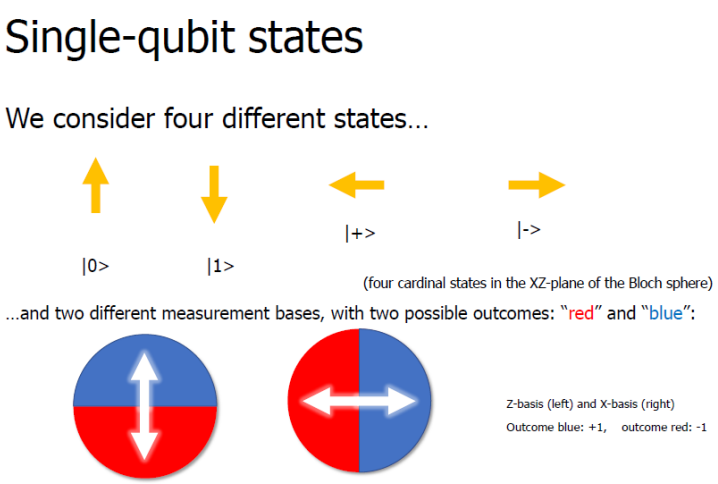
Now that we have seen the power and simplicity of the one-time pad, we are left with a single question: how do we generate these pads in a convenient manner, such that Eve cannot gain access to them, and such that we never have to use a pad repeatedly? In this lecture, Tim Coopmans will explain a simple quantum protocol, called BB84 (hinted at in the quizzes for Module 4), which allows Alice and Bob to obtain correlated random bits, without ever having to disclose the value of those bits.

Please note: at 2:13 a small error appears in the Section. Tim says here: "In the left one we measure a horizontal state in a horizontal basis". However it should be: "In the left one we measure a vertical state in a vertical basis."



Tim Coopmans,Quantum Key Distribution

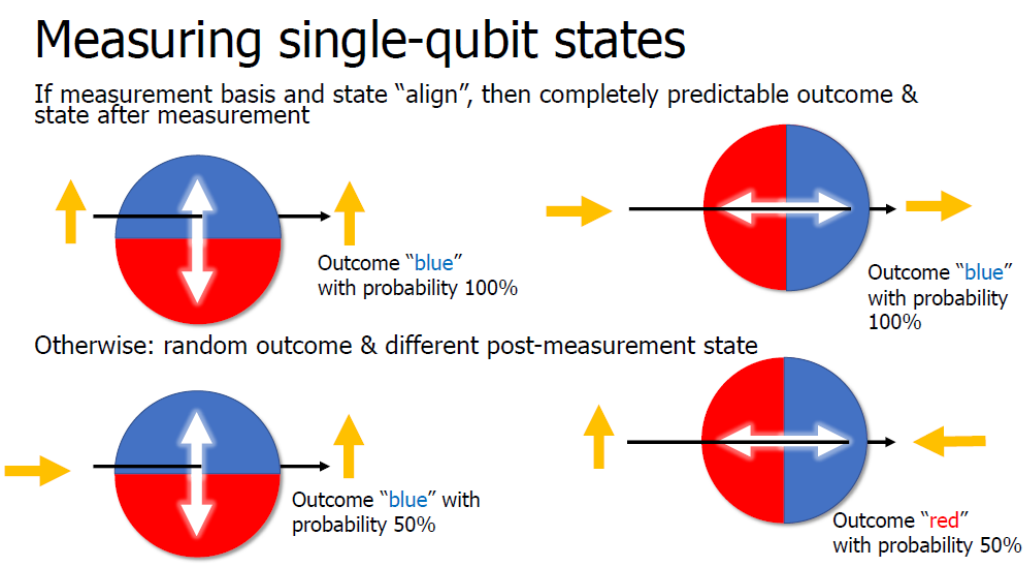
In the previous Section, we saw how the one-time pad scheme can be used to send a message securely. There is nothing quantum about the one-time pad scheme: it only involves classical zeroes and ones. In this Section we will use a Quantum Key Distribution scheme to generate a secret key, which can then be used in the one-time pad scheme for sending a message securely.



Tim Coopmans,Quantum Key Distribution

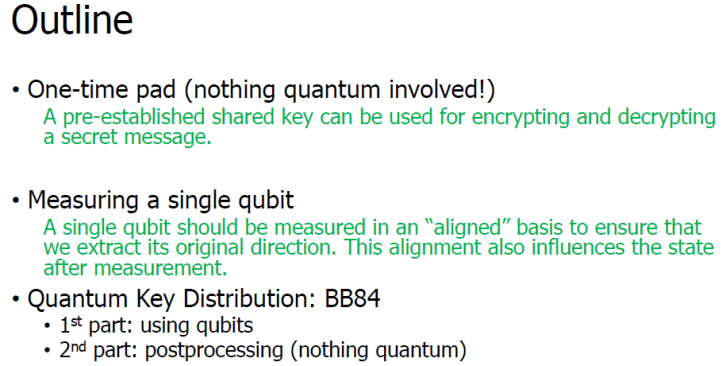
We will first give an overview of the possible single-qubit states and measurements we will use. The second part of the Section is devoted to explaining a famous Quantum Key Distribution scheme, called BB84, named after its inventors Charles Bennett and Giles Brassard. In the BB84 protocol, we will use four different single-qubit states: - the 0-state, here represented by an upward arrow - the 1-state, a downward arrow - and the plus- and minus-states, here depicted as horizontal arrows. On top of this, we will use two possible measurement bases: a "vertical" basis and a "horizontal" basis, which you could think of as 'slits' through which the state passes. We denote the two possible outcomes as 'red' and 'blue'. For those of you who have seen these states before: you probably recognize these as the four cardinal states in the XZ-plane of the Bloch sphere. The two measurement bases are the Z-basis on the left and the X-basis on the right. Furthermore, the outcomes blue and red correspond to outcomes +1 and -1 respectively. If these names do not sound familiar to you, then don't worry: we will not need them in the remainder of this Section. Let us do a short recap of

measuring single-qubit states. Suppose we have a qubit - a yellow arrow - and a measurement basis -the round slits. And now we measure the qubit in that basis.



Tim Coopmans,Quantum Key Distribution

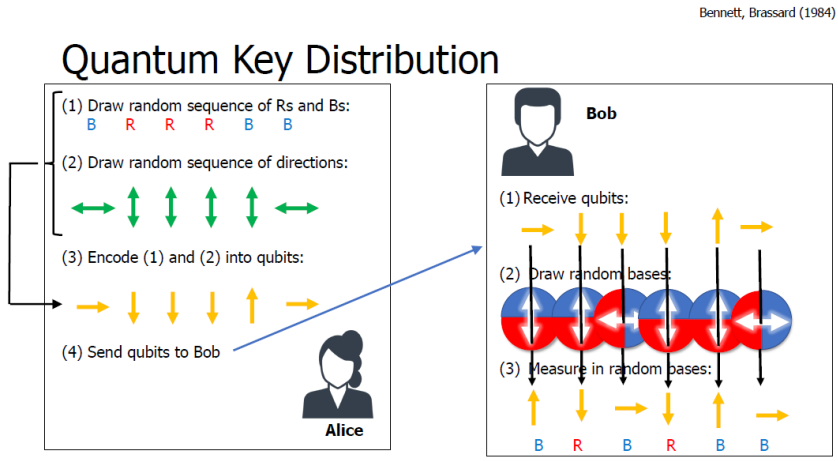
Then if the state of the qubit aligns with the measurement basis, then only a single outcome is possible. Moreover, the state of the qubit does not change, and from the measurement outcome, we learn its state. This situation is depicted in the upper two examples: in the left one, we measure a "horizontal" state in a 'horizontal' basis. The arrow passes through the slit unharmed, and the outcome is 'blue' since the arrow points upward. A similar explanation can be given for the picture in the right upper corner. When the measurement basis and qubit state do not align, the situation is very different. In the example in the left lower corner a horizontal state is measured in a vertical measurement basis. The state then flips to vertical direction: either to up or to down, each with probability 50%. The measurement outcome is precisely related to the direction of the arrow after the flip: blue if the state flipped to 'up', and 'red' if the state flipped to 'down'. Analogously, measuring a vertical state in a horizontal basis also yields a completely random outcome, as can be seen in the lower right corner. Using a similar reasoning, one can find out if the outcome and post-measurement state are completely predictable, or random, for all 8 possible combinations of states and measurement bases. Measuring a single-qubit state can be summarized in the following lines: a single qubit should be measured in an 'aligned' basis to ensure that we extract its original direction. This alignment also influences the state after the measurement.



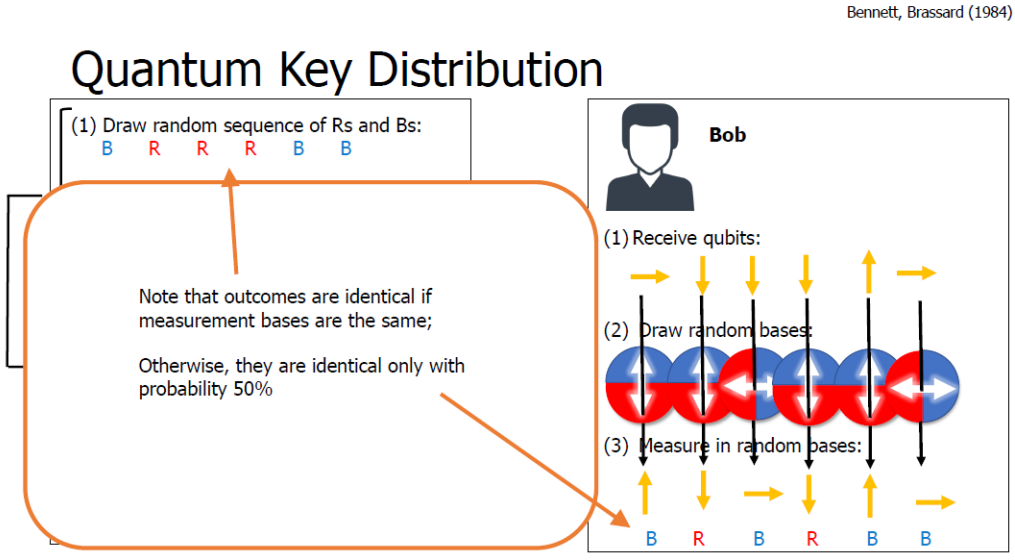
Tim Coopmans,Quantum Key Distribution

Next, we will look at the BB84 protocol for generating a secure shared key. The protocol consists of two parts: in the first part, we use qubits for encoding classical information. And the second part is a post-processing step, which does not involve any quantum operations at all. The setup is as follows: there is Alice, on the left, and Bob on the right. In real life,

they are far away from each other, so they cannot see each other's actions. The BB84 protocol starts by Alice picking a random sequence of bits, which we have written here as Rs and Bs instead of zeroes and ones. These are Reds and Blues. On top of this, she picks a different random sequence of directions: here depicted as vertical and horizontal instead of zeroes and ones. In the third step, she encodes this information into single-qubit states. In a fashion that is compatible with the measurement bases we saw before, she encodes the two sequences bitwise as arrows. For example, she turns blue plus horizontal in a right-pointing state, she turns red plus vertical in a down-pointing state, and so on and forth for the remaining qubits in the random sequences. These qubits, she then sends to Bob. Bob receives the qubits, and draws a sequence of random measurement bases. Note that writing such bases as a direction, as we did here in green arrows on the left with Alice, comes down to precisely the same as writing them as measurement bases here on the right with Bob. In the next step, Bob measures the qubits he received from Alice bitwisely in a randomly chosen bases.

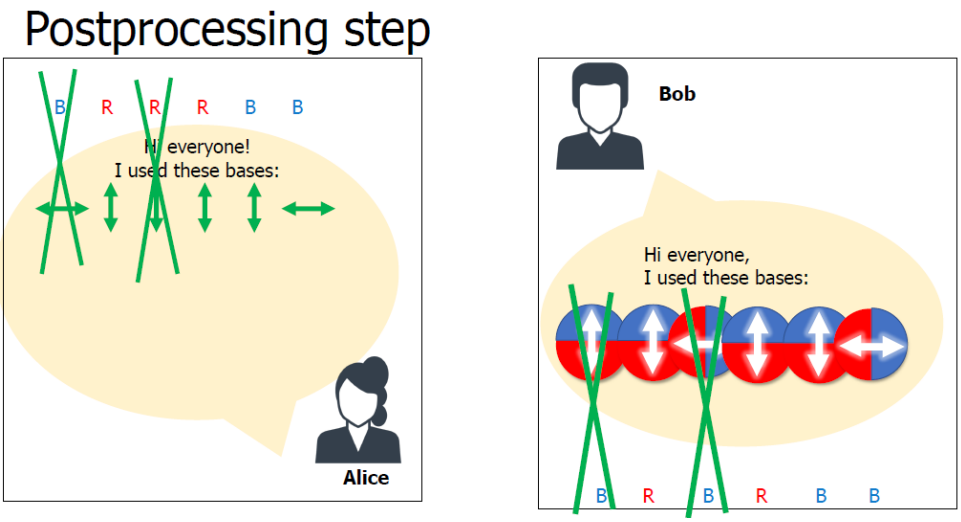


Tim Coopmans,Quantum Key Distribution

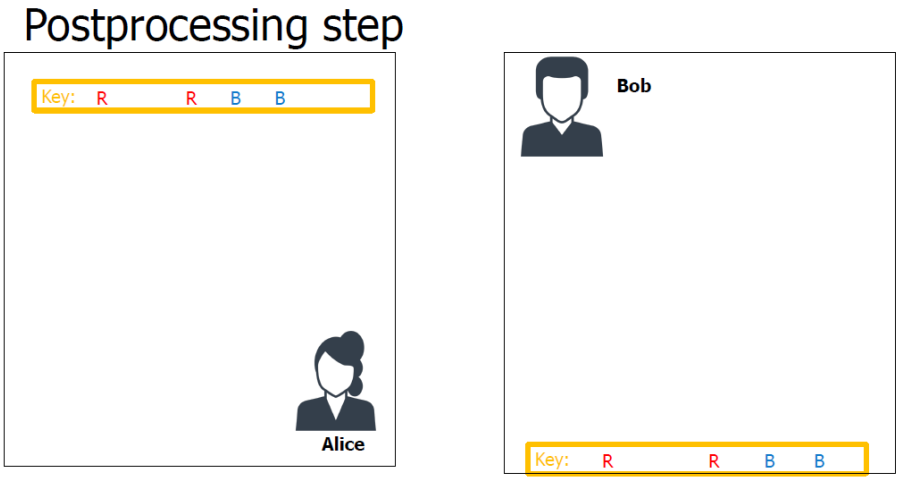


Tim Coopmans,Quantum Key Distribution

Note that this looks surprisingly much like the reverse of what Alice did when she was encoding the qubits. Also note that the outcomes that Bob has are identical to Alice's initial randomly chosen bits, in the case that her chosen direction is identical to Bob's chosen measurement basis. If these two are different, then the outcomes are different with a probability of 50%. If this went a bit too fast, then I encourage you to pause the Section for a bit and convince yourself why the statement here in the orange block is indeed correct. The next part does not involve any quantum any more but only consists of classical post-processing. First, both Alice and Bob publicly announce their measurement bases. Everyone in the world may hear these, even people wishing to eavesdrop on Alice and Bob's communication. After this, Alice and Bob remove all Rs and Bs where their measurement bases are different. The resulting Rs and Bs must necessarily be the same

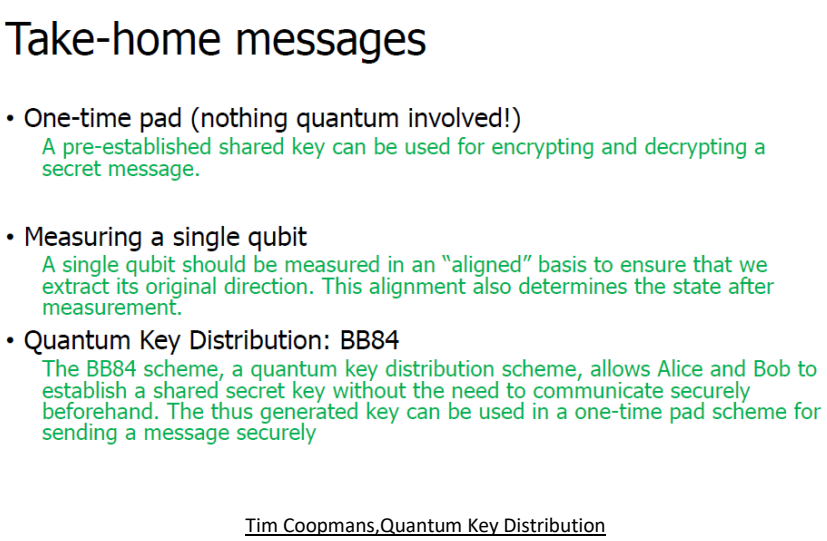


Tim Coopmans,Quantum Key Distribution



Tim Coopmans,Quantum Key Distribution

on both sides. It is important to realize that the measurement bases on both sides have been made public, but that the Rs and Bs are still secret. This means that we can use the Rs and Bs as a secret shared key between Alice and Bob. We finish this Section with the last take-home message. With the BB84 Quantum Key Distribution scheme, Alice and Bob generate a shared secret key without having to communicate securely beforehand. This key can then be used in a one-time pad scheme for sending a message securely.

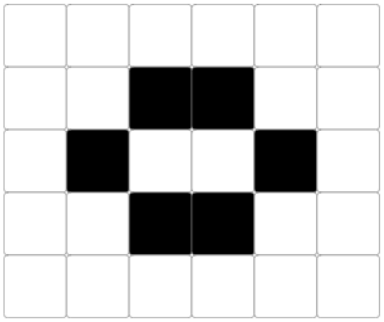


Main takeaways

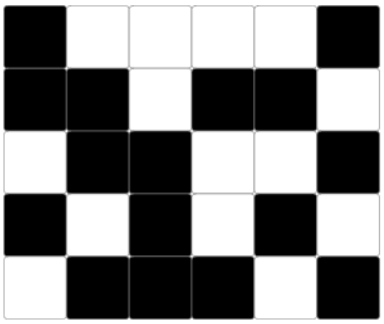
- BB84 is a simple, entanglement-free quantum key distribution protocol, which allows Alice to generate and transmit random numbers such that Bob can successfully share a large portion of them, but Eve cannot intercept them.
- This scheme uses the fact that consecutive measurements in different bases produce random outcomes in order to allow Alice and Bob to detect Eve's presence.
- At long distances, where quantum repeaters become necessary, we use an alternate quantum key distribution protocol called E91, invented by Artur Ekert in 1991.

Practice Questions

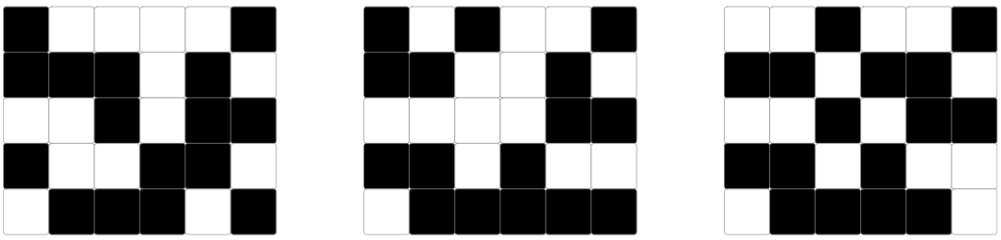
Suppose that Alice and Bob are in independent computer science laboratories, studying Conway's Game of Life. In this game, black-and-white pixel images change through time according to a simple set of rules. Alice is searching for patterns which remain fixed under these rules, and she finds one, the "beehive":



She wishes to communicate this pattern to Bob, so she performs the bitwise XOR with a secret key that she shares with him:



Consider the following patterns:



A B C

One-Time Pad Encryption/Decryption

Which of the patterns above corresponds to the encrypted message which Alice sends to Bob?

An encrypted pattern will have white squares wherever the message and the key have the same colour, black otherwise.

- A
- B
- C
- None of the above

Cardinal States of the Bloch Sphere

In the lecture, Tim mentioned that the states $|0\rangle$, $|+\rangle$, $|1\rangle$, and $|-\rangle$ are cardinal states on the Bloch sphere.

Which angles θ and φ correspond to the states $|0\rangle$ and $|-\rangle$, respectively?

You may refer back to the earlier lecture on ket notation, the Quantum Library, or the Wikipedia article on the Bloch sphere.

- $|0\rangle \mapsto (\theta=0, \varphi=0)$, $|-\rangle \mapsto (\theta=\pi 4, \varphi=\pi)$
- $|0\rangle \mapsto (\theta=\pi 2, \varphi=\pi 2)$, $|-\rangle \mapsto (\theta=\pi 2, \varphi=\pi)$
- $|0\rangle \mapsto (\theta=0, \varphi=2\pi)$, $|-\rangle \mapsto (\theta=\pi 2, \varphi=0)$
- $|0\rangle \mapsto (\theta=0, \varphi=0)$, $|-\rangle \mapsto (\theta=\pi 2, \varphi=\pi)$

Randomness of Encrypted Bits

Suppose that a random bit is produced which is 0 with 50% probability, and 1 with probability 50%.

If we flip this bit, exchanging 0 and 1, what is the probability that it is now 0?

- 0%
- 25%
- 50%
- 75%

Finding Measurement Outcomes

In order to analyse protocols such as BB84, it is necessary to calculate "sandwich" products, of the form $\langle\psi|O|\psi\rangle$, in order to determine measurement outcomes.

For small systems, it is convenient to write out these products using vectors and matrices.

For example:

$$\langle 0|\, Z\, |0\rangle = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = 1.$$

Which of the following expressions for $\langle +|\, X\, |-\rangle$ is correct?

- $\frac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}$
- $\frac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}$
- $\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix}$
- $\frac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}$

Secret Key produced by BB84

Suppose that Alice generates the following ten BB84 states:

$|0\rangle, |-\rangle, |+\rangle, |0\rangle, |1\rangle, |+\rangle, |-\rangle, |0\rangle, |+\rangle, |1\rangle$

She sends them to Bob, who measures them in the following measurement bases:

X,X,Z,Z,X,Z,X,Z,Z,X.

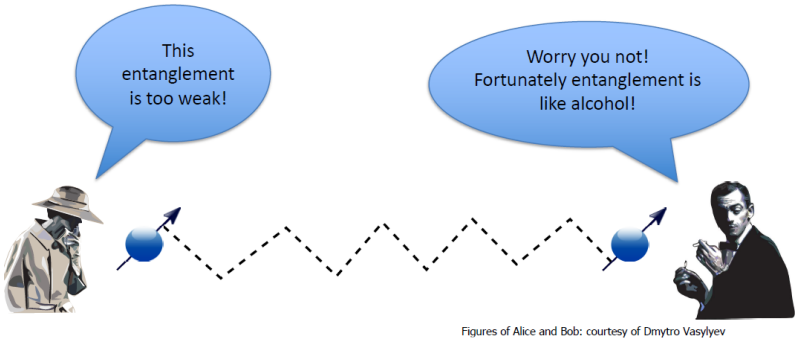
After Alice and Bob reveal their measurement bases, what is their shared secret key?

The states $|0\rangle$ and $|+\rangle$ result in 0 when measured; $|1\rangle$ and $|-\rangle$ result in 1.

- 0000
- 1010
- 1110
- 001100

Introducing entanglement distillation

We've seen that to build a quantum internet, we need to be able to generate entangled states distributed across different nodes. This is a nontrivial task where hardware can be a limiting factor. In this lecture, Filip Rozpedek explains the concept of entanglement distillation, a method to filter out non-entangled states which can be generated accidentally.



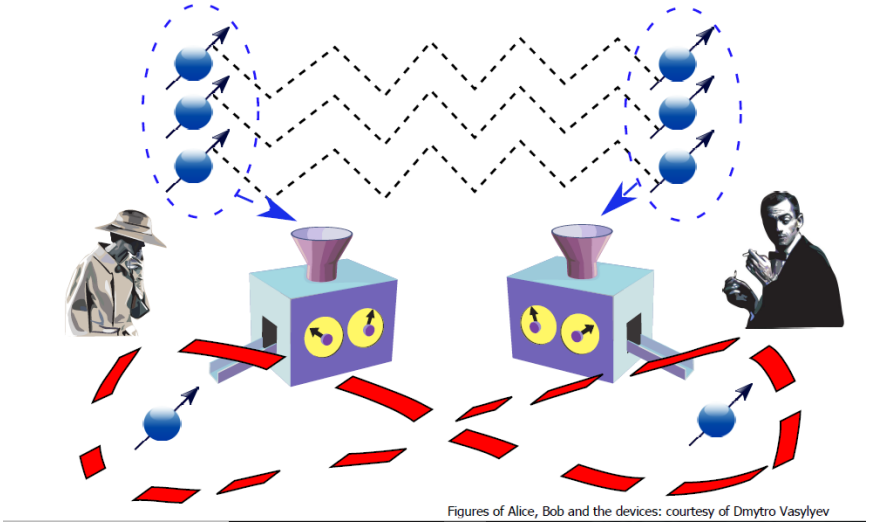
Filip Rozpedek,Introducing entanglement distillation

In this Section I will introduce the concept of entanglement distillation which is a crucial building block of a large scale quantum internet. We will start with a short motivation. Let us consider a scenario where Alice and Bob are separated by a long distance and they have access to certain experimental setups that allow them to generate long distance entanglement. Unfortunately, fully entangled states which are perfectly correlated are a great idealization and from an experimental perspective almost impossible to create. In general, there can be many reasons for this, e.g. their experimental equipment used to generate this long distance entanglement isn't perfect or they cannot maintain their quantum systems long enough. Additionally, to overcome the problem of losses over such a long distance, there might be additional repeaters placed between Alice and Bob which allow them to significantly increase the rate of generating those states. However, such repeater nodes perform then additional imperfect operations, which further decrease the quality of the resulting entanglement between Alice and Bob.

Filip Rozpedek,Introducing entanglement distillation

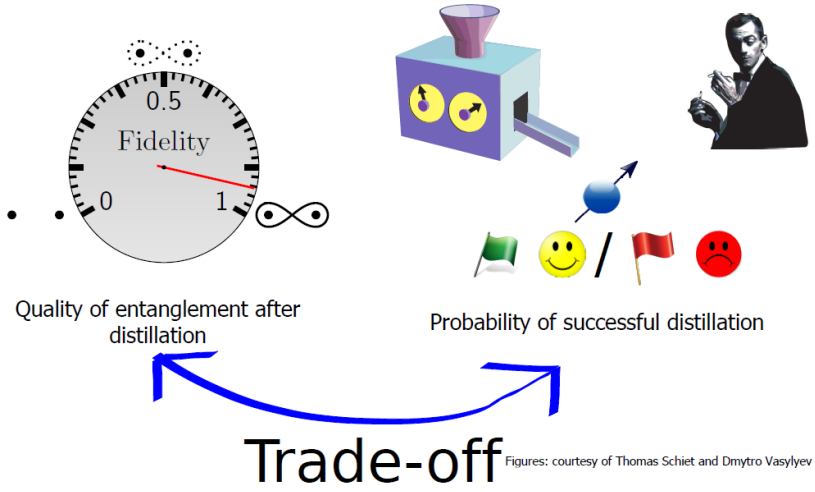
Effectively, the quantum correlations become then weaker and completely diluted in a mixture of various other quantum states. This means that the quality of the generated entanglement might be too low for specific applications. For example, too weak correlations would not allow Alice and Bob to generate any shared secret key. Fortunately, Bob, as most physicists, has a lot of experience with alcohol and he expects that some of the corresponding practical skills could also be applicable in this situation. Specifically, in his previous life, Bob used to work in a distillery. There, a very weak form of alcohol is stored in the massive stills as the ones on the picture. This weak beer is a mixture of pure alcohol, water and

some other substances. For the high-quality whisky Bob needed to extract this pure ethanol from everything else. By evaporating alcohol and later condensing it outside of the stills he was able to separate it from the "contamination" that should not end up in any high-quality whisky. It was then possible to separate such alcohol from the water to the desired degree by running multiple rounds of this distillation procedure.



Filip Rozpedek,Introducing entanglement distillation

Bob then decided to apply such a distillation procedure also to entanglement. In entanglement distillation, Alice and Bob will use multiple copies of weakly entangled states. In this procedure, although each state has some, possibly small amount of entanglement, it is possible to combine those copies together, and eliminate the contamination by extracting only the entangled part, so that at the end smaller number of more strongly entangled states can be extracted. Guided by his experience in the whisky distillery, Bob knows that for realistic procedures, a single run of such an entanglement distillation would in most cases not be enough. But repeated procedure will be necessary. Also similarly to alcohol distillation, where the total amount of alcohol cannot be increased, the local operations and classical communication used by Alice and Bob allow them for concentrating existing entanglement into a smaller number of copies, without increasing the total amount of entanglement. The main goal of such an entanglement distillation procedure is to increase the quality of the resulting entanglement as much as possible. Here we can for example use fidelity as a measure of closeness of the resulting state to the maximally entangled one.



Filip Rozpedek,Introducing entanglement distillation

However, any practical entanglement distillation procedure is normally probabilistic which means that there is only a finite probability that the resulting state that Alice and Bob obtain is more entangled than the input copies. These procedures are fortunately heralded, which means that the devices of Alice and Bob raise flags that tell them, whether the procedure

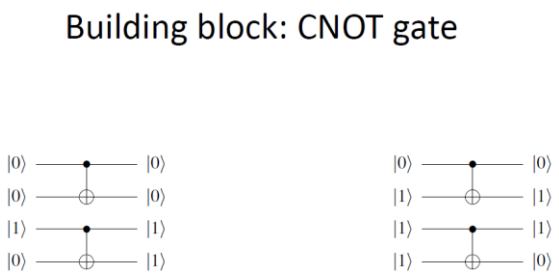
has succeeded or not and therefore whether they should keep or discard the resulting state. As one could expect, there is a trade off between the two effects. In particular, if one designs a procedure that allows us to greatly improve the fidelity, such procedure will generally succeed with very small probability. Conversely, if one wants to focus on a procedure that succeeds with high probability, the corresponding gain in fidelity will in general be small.

Main takeaways

- Creating ideal maximally-entangled states is hard.
- Repeaters can help creating entangled states over large distances, but they also introduce noise.
- We need entanglement distillation to filter out states that are not maximally entangled.
- Entanglement distillation does not increase the amount of entanglement, just as alcohol distillation does not increase the amount of alcohol. In both distillation schemes, quantity is reduced, so that quality can be enhanced.

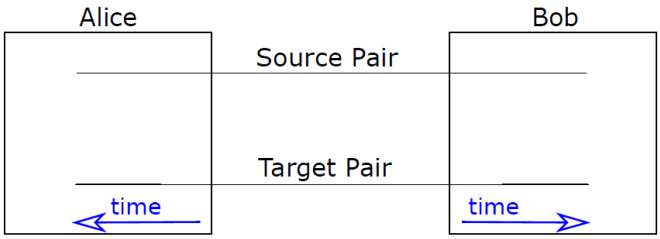
Entanglement distillation example

To get an idea of how entanglement distillation will work in practice, it is helpful to look at an example of this process in action. In this lecture, Filip Rozpedek outlines a simplified entanglement distillation protocol which uses individual measurements with Alice and Bob's devices to determine if they have accidentally been given a separable state. These measurements are reminiscent of quantum error correction, discussed in Module 3. The main difference here is that, when Alice and Bob detect that an error has occurred, they don't need to correct it, they can simply discard their bits and try again.

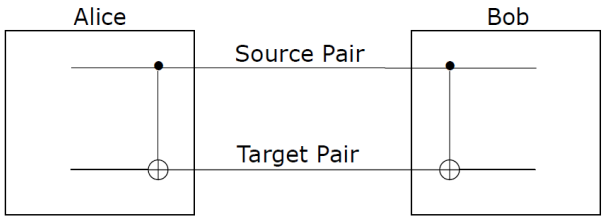


Filip Rozpedek,Entanglement distillation: example

In this Section we will discuss an example class of very common entanglement distillation protocols, which operate on two entangled copies and aim at extracting a single more entangled copy. We will then also consider a specific distillation procedure from this class, used to deal with a very particular type of errors. The procedures that we will consider here include a controlled NOT gate which has already been discussed before. This quantum gate is very analogous to its classical counterpart.



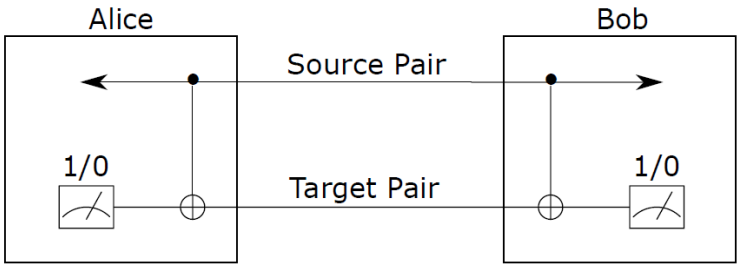
Filip Rozpedek,Entanglement distillation: example



Filip Rozpedek,Entanglement distillation: example

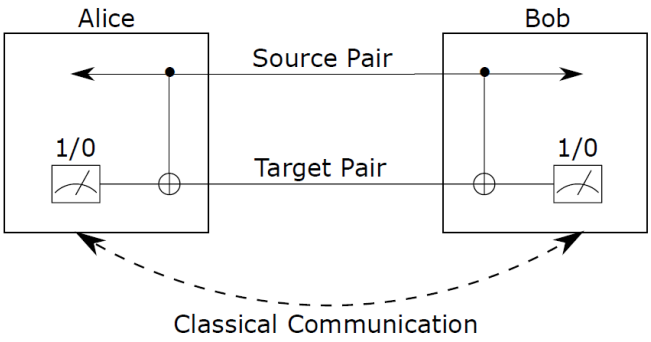
In the classical scenario depending on the value of the first bit, the second bit is flipped or not. That is if the first bit is 0, the second bit value remains untouched, as can be seen on the top two circuits. If the first bit is 1, the second bit value gets flipped, as can be seen on the bottom two circuits. Now, in the quantum case, our first “controlling bit” can be in

superposition of 0 and 1, in which case in superposition we flip and don't flip the second bit. This effectively allows us to correlate the two qubits, which in other words transfers certain information about the first qubit to the second one. This correlating property will be crucial in our distillation procedure.



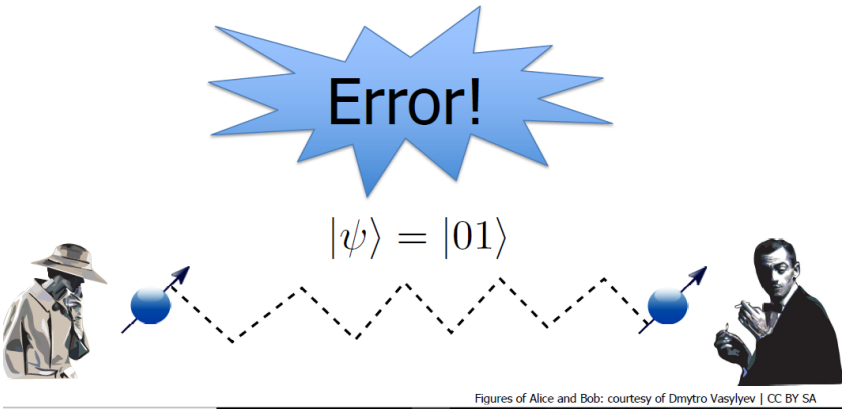
Filip Rozpedek,Entanglement distillation: example

The general setup of the procedure is as follows. We have Alice and Bob, whose labs are depicted with the black squares. They share two entangled pairs, depicted by the two horizontal lines. The source pair is the one whose entanglement we will be trying to increase using the second target copy. We also note that the corresponding operations of Alice and Bob will be depicted such that time flows to the left in Alice's lab and to the right in Bob's lab.



Filip Rozpedek,Entanglement distillation: example

In the first step both Alice and Bob perform local CNOT gates as shown on the slide. These CNOTs effectively correlate the two copies. Next, Alice and Bob measure the two qubits of the target copy locally. Because the target and the source copy were correlated, by collapsing the correlations of the target copy they effectively project the source copy in different states depending on the outcomes of their measurement. Finally they use a classical communication channel to compare these classical outcomes. For outcome configuration in which the source pair becomes projected into a more entangled state, the source pair is kept. It is discarded if the configuration of outcomes of Alice and Bob doesn't match the success condition, as in that case the measurements have actually decreased the entanglement in the source pair.



Filip Rozpedek,Entanglement distillation: example

Let us now consider a very specific distillation example for a very specific type of noise in the system. In particular let us assume that Alice and Bob want to generate the state Phi Plus as shown on the slide. Unfortunately from time to time an error occurs.In which case Alice and Bob unknowingly share the state psi which is an unentangled product state that is orthogonal to the desired maximally entangled state phi plus. Let us now go through the distillation procedure on the two copies of such a state as described before. Let us firstly consider the case in which both copies of the state are actually the desired phi plus state. Then, Alice and Bob apply locally their CNOT gates, so that the CNOTs are performed from qubit A1 to A2 and from B1 to B2. You can check yourself that in this case the total state remains unaffected. Let us now look at the case where the second, target copy has an error. In this case the CNOTs transform the two copies as shown. Similarly we can obtain the corresponding result if the source copy has an error. And if both copies have an error. You can now pause the Section to verify these calculations. The final part of the distillation is the measurement of the target qubits A2 and B2. Let us look here for which cases, the qubits A2 and B2 can both give Alice and Bob outcome 1. In the first case Alice and Bob apply their measurement to the state Phi plus. Hence, with probability 50% they measure the desired 11 and with 50% probability they measure 00. In the second case it is never possible to obtain 11. Neither in the third. Nor in the 4th. Hence, if Alice and Bob keep the source state on registers A1B1 only if the target copy was measured to be 11, they are guaranteed to extract the perfect maximally entangled state.

$$|\Phi^+\rangle_{A_1B_1} \otimes |\Phi^+\rangle_{A_2B_2} \rightarrow \text{bilocal CNOTs} \rightarrow |\Phi^+\rangle_{A_1B_1} \otimes |\Phi^+\rangle_{A_2B_2}$$

Measure "11" on A_2B_2: ✓ / ✗

$$|\Phi^+\rangle_{A_1B_1} \otimes |01\rangle_{A_2B_2} \rightarrow \text{bilocal CNOTs} \rightarrow \frac{1}{\sqrt{2}}\left(|00\rangle_{A_1B_1} \otimes |01\rangle_{A_2B_2} + |11\rangle_{A_1B_1} \otimes |10\rangle_{A_2B_2}\right)$$

Measure "11" on A_2B_2: ✗

$$|01\rangle_{A_1B_1} \otimes |\Phi^+\rangle_{A_2B_2} \rightarrow \text{bilocal CNOTs} \rightarrow |01\rangle_{A_1B_1} \otimes \frac{1}{\sqrt{2}}\left(|01\rangle_{A_2B_2} + |10\rangle_{A_2B_2}\right)$$

Measure "11" on A_2B_2: ✗

$$|01\rangle_{A_1B_1} \otimes |01\rangle_{A_2B_2} \rightarrow \text{bilocal CNOTs} \rightarrow |01\rangle_{A_1B_1} \otimes |00\rangle_{A_2B_2}$$

Measure "11" on A_2B_2: ✗

Filip Rozpedek,Entanglement distillation: example

Of course we see that there are plenty of possibilities to obtain different outcomes on the target copies. All those non-11 cases will classify as failure and in those cases both copies will be discarded. This was an example scenario in which with finite probability a perfect maximally entangled state could be extracted just from two copies of the noisy state. In

most cases however, the noise that arises in the experiment cannot be completely eliminated when performing distillation only on two input copies.

Main takeaways

- By creating 2 entangled pairs one can use a gate and measurement scheme to get a measurement result which verifies an entangled state.
- Instead of 2 pairs, we end up with only 1 entangled state. In general, one can create the protocol such that this single entangled state is more entangled than either of the two initial states.

Practice Questions

Properties of Distillation

The name 'entanglement distillation' establishes chemical distillation as an analogous process.

What are the main similarities between chemical distillation and entanglement distillation?

Select all correct answers.

- [ ] Both chemical distillation and entanglement distillation consume large amounts of energy.
- [ ] Like chemical distillation, entanglement distillation involves keeping only a small fraction of the initial resource.
- [ ] Like chemical distillation, entanglement distillation is a bulk process, and becomes more efficient the larger the input.
- [ ] Both chemical distillation and entanglement distillation increase the quality of the input resource.

Effect of Bilocal CNOTs

Suppose that Alice and Bob are trying to distill entanglement, and that their apparatus may produce the state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ or $|11\rangle$ (this is in contrast to the apparatus discussed in the lecture, which produces $|01\rangle$). Assume that Alice and Bob produce two pairs, initially in the state $\frac{|00\rangle_{A_1B_1}+|11\rangle_{A_1B_1}}{\sqrt{2}} \otimes |11\rangle_{A_2B_2} \equiv \frac{1}{\sqrt{2}}\left(|0011\rangle_{A_1B_1A_2B_2} + |1111\rangle_{A_1B_1A_2B_2}\right)$.

What is the resulting state after the bilocal CNOTs?

- $\frac{1}{2}\left(|0000\rangle_{A_1B_1A_2B_2} + |0011\rangle_{A_1B_1A_2B_2} + |1100\rangle_{A_1B_1A_2B_2} + |1111\rangle_{A_1B_1A_2B_2}\right)$
- $|1111\rangle_{A_1B_1A_2B_2}$
- $\frac{1}{\sqrt{2}}\left(|0011\rangle_{A_1B_1A_2B_2} + |1100\rangle_{A_1B_1A_2B_2}\right)$
- $\frac{1}{\sqrt{2}}\left(|0011\rangle_{A_1B_1A_2B_2} + |1111\rangle_{A_1B_1A_2B_2}\right)$

Learn more

- In this news article **One step closer to the quantum internet by distillation** and the accompanying Section **Entanglement Distillation Explained**, you can learn more about a recent accomplishment in entanglement distillation and how this method is essential to realize a trustworthy quantum network between several quantum nodes.
- You can also read the scientific paper **Entanglement Distillation between Solid-State Quantum Network Nodes,** which was published in Science Magazine in 2018, to learn more about how researchers in Delft and Oxford have now managed to distill a strong entangled link by combining multiple weaker quantum links into one.

Module 6: Wrapping up The Building Blocks of a Quantum Computer Part 2

Introduction to Module 6

We have now come to the end of our ascent through the layers of the quantum stack, and to the end of our discussion about the development and applications of quantum networks. Along the way, you've had the opportunity to ask us many questions through the forums, and we hope that you've found the answers enlightening.

In this final module of the course, we've prepared Sections to respond to three questions in our "Ask Me Anything" unit in Module 3. In these responses, Ben Criger will explain how to calculate the parameters of a surface code from the frequently-used diagrams we've seen so far, how we can construct small quantum Fourier transforms and where you can find software libraries to study quantum computing. This module ends with the final exam, where we re-cap the topics we've covered in the course. This is preceded by a practice exam, which you can use to refresh your knowledge before taking the final. We hope you have enjoyed this treatment of quantum computers and quantum networks.

Parameters of a surface code

In Module 3 you've had the chance to ask a question to be answered by one of the teachers. Some of you proposed requests in this thread and the three most upvoted questions will be answered by Ben Criger. This first Section will explain how you can calculate the parameters of a surface code.

Researchers in quantum information often use a simplified diagram in order to talk about surface codes. It can be a little bit difficult for people who are beginners in the field to know exactly what is being discussed in terms of the number of physical qubits, the number of logical qubits and the distance of a surface code given the diagram alone. So, we are going over the calculation method quickly just to see how everything works.

Now to start from the very basics, let's recall that X times Z is equal to minus Z times X, where X and Z are the two by two Pauli matrices that we have been discussing throughout the course. We can very quickly, and if you are not following along don't worry about it, we can define stabilizer codes to be a set of code states Psi in the code C, such that S times Psi is equal to Psi for every operator s in a big S, which is a set of stabilizer operators. So, we say that s stabilizes Psi because Psi is the plus 1 eigenstate of s, for all of the little s's in this big group of s's. And that is nice, and you can do a lot of classically efficient computations with this, but the really nice property is that this allows you to detect errors.

Let's assume that first of each of these s's is going to be some multiqubit Pauli operator, and that E, an error that may or may not occurred, is also a multiqubit Pauli operator. What happens if we try to measure S and there is an E here that anticomuttes with S, the same way X does with Z. If you go through this math for some specific example or if you do it, in general, using only the fact that E times S is equal to minus S times E and that E squared, like al multiqubit Pauli's, is the identity. You will end up getting minus 1. Now if this error had not occurred, not producing the state E Psi, but rather Psi this would be plus 1. And this means that as long as we are working with the stabilizer code, errors are going to have a physically measurable effect on the system.

So, how do we make a surface code? We begin with a square array of qubits that we layout here, and then we start defining stabilizers. Here is a weight two stabilizer of the form Z Z on these two qubits. We have a weight four stabilizer of the form X X X X on these four. And similarly, local stabilizers are defined all over the lattice. This arrangement of stabilizers and data qubits becomes this surface code picture. And typically, people will include a little legend to let you know which of these letters corresponds to which of these colors because it is not clear a priority.

So, here I have used black tiles to host Z stabilizers and I have used white tiles to host X stabilizers. Here we can see the weight two stabilizers that are going around the perimeter and the weight four stabilizers that are on the interior. The logical operators for this code run all the way over one side of the square or the other. Depending on which color is which you might have different logical operators on different sides, but here the logical Z operator runs down the right side of

the tile and the logical X operator runs across the top. And they are both weight three because we are dealing with a three by three square.

How do you calculate n, k, and d for a surface code? n is simply the length in the x-direction times the length in the z-direction. So, in this case, it is 3 times 3, which is 9. An important thing to remember is not to count the number of squares that are in a row, but rather the number of qubits that are in a row. So, here we have 9 total. k, the number of logical qubits is always 1 for these codes. There are some ways to get k not to be 1, but they are too advanced to cover in this course. And d, the code distance, is the minimum side length of this square. Which is either lx or lz, whichever smaller the minimum. And in this case that is 3. So, as you can see, square tiles are nice when error rates are equal, and you want to have an equal distance for both X and Z logical operators.

Main takeaways

- n = The length in the x-direction times the length in the z-direction.
- k = The number of logical qubits.
- d = The length of the side with the shortest length (the code distance).

Small quantum Fourier transforms

The second most popular question that was proposed by you, was "Could you please explain an example of the application of the QFT and the inverse QFT, in a simple circuit?" Ben will explain a small quantum Fourier transform in this next Section.

It can be difficult, at first, to understand the quantum Fourier transform, as well as its circuit decomposition. And here we are going to take a look at a geometric picture for deriving the coefficients of the quantum Fourier transform, and a little example on two qubits. So, let's begin with this mathematical formula which tells us what the quantum Fourier transform on k qubits is doing. It's taking a state that represents an integer z. So, for instance on three qubits the state 7 would be 1 1 1, the state two would be 0 1 0, etcetera. And we map the integer z to a superposition that is uniform in magnitude but varies in phase. And here is the formula for evaluating that phase. And we have a superposition over all the computational basis states b. The states from 0 0 0 up to 1 1 1, in the regular old set eigenbasis. This can be a little bit opaque and hard to understand, but it gets easier if we look at an example on two qubits. Here we have the matrix for the quantum Fourier transform on two qubits. It's just a half. Which is 1 over 2^k with k equals 2, when squared rooted it cancels out the 2, you just get 2. And we have the columns 1 1 1 1, which is what happens if I multiply every integer b by zero and then take e^(2 pi i) times zero. And then to understand the rest of the coefficients. We can understand z, the input, as a speed at which the quantum Fourier transform proceeds around the unit circle. So, if we have 0 1, the one state. The quantum Fourier transform will take 1 step at a time on this diagram. So, it will be 1 i -1 -i, as we see here 1 i -1 -i. If we take two steps at a time for the input state 2, it were 1 0 on two qubits. We end up with 1 -1 1 -1, exactly as we see on the two-state here. If we take 3 steps at a time on a circle with four points on it, that is actually the same as 1 step backwards. And we can see that we have 1 -i -1 i in the coefficients in the final column of the matrix. Now we went over a little bit over how to decompose this into a circuit, but I figured I do it concretely here for you. We have a Hadamard gate, a controlled square root of Z, another Hadamard gate on the target qubit of the controlled square root of Z, and then a swap gate at the end. Now the swap gate at the end wouldn't always appear in different figures that you might see in research papers or textbooks, or courses for example, because it is often assumed that you can do swap gates like this one classically just by relabeling which qubits are which at the end. But for the sake of completeness, I included it here. Now we can calculate the result of performing these four gates in order simply by putting the matrices in the reverse of the order in which they appear in the circuit. So, we have swap, identity tensor Hadamard, the controlled square root of Z, and Hadamard tensor identity here. And I invite you to multiply these four matrices by yourself and observe that they do indeed recreate the quantum Fourier transform on two qubits. Now if that is too easy, let's do another challenge or let's have you do another challenge. In which you decompose, into a circuit, the quantum Fourier transform on one qubit. Now on one qubit, you have 1 over root 2, because k is just 1. And then you can either proceed around the unit circle 0 steps at the time, staying at one, or one step at the time but missing this i. So, you would just go from 1 to minus 1. And if you can find a circuit decomposition for that, we may have already discussed it in the course, then you are well underway to understanding the quantum Fourier transform.

Quantum libraries

The final Section in this series of "Ask the teacher", will answer the question where to find quantum libraries. One of the things that's very valuable in quantum computing and in science in general, is the ability to write out a little bit of software and then never have to look at that problem again. For example, you can look at condensed matter physicists, who like to diagonalize large matrices, and they don't rewrite their diagonalization routines from scratch every time. Unless maybe they're going for an educational exercise or something. In a more professional context or in research, they will use some prepackaged software in order to make their lives easier. This is also the case in quantum computing, where there are a lot of precompiled libraries that you can gain access to, and really the number and efficacy of these libraries have been exploding recently. Let's take a look at a few of them. Wikipedia has a nice list of these under the quantum programming page, where they assemble a large series of these algorithms, libraries, toolkits, and indeed a lot of domain specific and specialized languages that you can use to study quantum computing. There is ProjectQ, which is from Zurich and/or Microsoft, Qiskit from IBM, Forest from Rigetti, and the list goes on. It's mostly language focused. If you wanted a specific library that does one thing and does it well, there are other places that you can go look. Like for example, there's the venerable Quantiki, which is the quantum information wiki that's been around for a while. And if you like to find a job in quantum information or find out when the next conference is, Quantiki is a great place to look. But they also have a list of quantum computation simulators and there are simulators broken down by language. So, here we can see that there are some from C and C++, there's quite a few, including the QX Simulator, which is developed at Delft. And if we scroll down, we can find that there's some more exotic languages like CaML, OCaml, Coq, F#. Let's see if there's something special on the list. There are a few in Julia. So, if you're more accustomed to Julia, if you have been working in JuliaBox in the quantum cryptography "MOOC" which is also developed around here, then you might want to take a look at QuantumOptics.jl which is a more hardware-oriented library, or QuantumWalk, which is nice for studying fundamental quantum information processes. And there are many of these libraries. There is a third big list that I have found last week after I contacted the hive mind on Twitter to let me know where all of these big lists are stored. And it is from Mark Fingerhuth, apologies if I've pronounced your name wrong, and these libraries are divided by categories. So, they have full-stack libraries, simulators, libraries to do quantum annealing, but also libraries to do what is called Experimental quantum computing, which is mostly, you know, domain specific languages and something called Generic lab tools. I would like to conclude by calling your attention to one of the well developed libraries in quantum computing which is QuTiP, which a lot of researchers already have installed. So, if I open up ipython here, and I wait of course for ipython to get started, then I can import everything from QuTiP, and output some of the simple quantum objects that QuTiP has ready for you, in memory. So, for example, the Pauli-X matrix is stored here as a quantum object, which is a custom datatype that include metadata's such as the dimensions of the object, the shape of the object, its type; whether it's a state or an operator, for instance, which can be difficult to track if you are using matrices, whether it's supposed to be Hermitian, so that you can check and make sure that there are no errors in your code, making something that's supposed to be Hermitian non-Hermitian, and then the matrix itself.

Practice exam

This is a practice exam. It will not influence your final grade, though the questions are similar to the final exam (next page). You may use this practice exam to test your knowledge before taking the final exam.Questions are arranged in the same order as the modules of the course.

Good luck!

We begin with Module 1 about the Introduction to the Building Blocks of a Quantum Computer and the layer Quantum to Classical Interface.

Practice question 1: Abstraction in the quantum stack

Which of the following layers in the stack is the highest (the most abstract, having the least direct interaction with the physical qubits)?

- Quantum-to-Classical
- Quantum Error Correction
- Quantum compiler
- Quantum algorithms

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 2: Measurement of spin qubits

Which electronic components are necessary in order to read out a spin qubit? Assume that we don't need to input any microwave signals during measurement.

Mark all that apply.

- Analog-to-digital converters
- Digital-to-analog converters
- Low-noise amplifiers
- On-chip charge sensors

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 3: Refrigeration

In order to realise quantum computations, we need powerful refrigerators to cool our qubits to extremely low temperatures, sometimes as low as 20 mK.

Which one of the following statements about refrigeration is true?

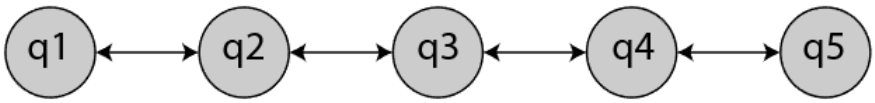
- As operating temperature decreases, cooling capacity remains constant.
- As operating temperature decreases, cooling capacity increases.
- As operating temperature decreases, cooling capacity decreases.

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

The following questions 4, 5 and 6 are related to Module 2 about the Micro-architecture and the Compiler & Programming language.

Qubit routing

Consider the following arrangement of qubits:



Where q_i represents a qubit. Each qubit can only do a 2-qubit gate (such as a CNOT) with its neighbours, indicated by the arrows.

Practice question 4: Qubit routing - algorithms

We want to use this structure to run an algorithm. Let's assume for now that the algorithms only consist of 2-qubit CNOT gates. We'll use the notation [q1,q2],[q3,q4],[q4,q5] to denote a circuit consisting of CNOT(q1,q2), followed by CNOT(q3,q4), followed by CNOT(q4,q5).

Which of the following algorithms requires the fewest SWAP gates to execute?

Assume that CNOTs must be performed in the exact order stated, and that SWAP gates must be used to route qubits.

- [q2,q4],[q1,q3],[q4,q5],[q3,q4]
- [q5,q3],[q4,q1],[q2,q3],[q4,q5]
- [q1,q3],[q2,q4],[q3,q5],[q1,q5]

Practice question 5: Qubit routing - algorithm, continued

Now we would like to run a specific algorithm on our qubits. The algorithm goes as follows:

CNOT q3,q5

CNOT q4,q2

CNOT q3,q1

CNOT q1,q5

CNOT q4,q1

CNOT q4,q3

CNOT q1,q2

If we want to run the gates in this exact order, how many SWAP gates would we need for routing?

- 0
- 3 or less
- 6 or less
- 8 or less

Practice question 6: Qubit architecture

When implementing a quantum algorithm, we have control over the initial ordering of the qubits. The compilation step that selects the ideal ordering is called *qubit placement*. Good placement can reduce the amount of operations used in routing.

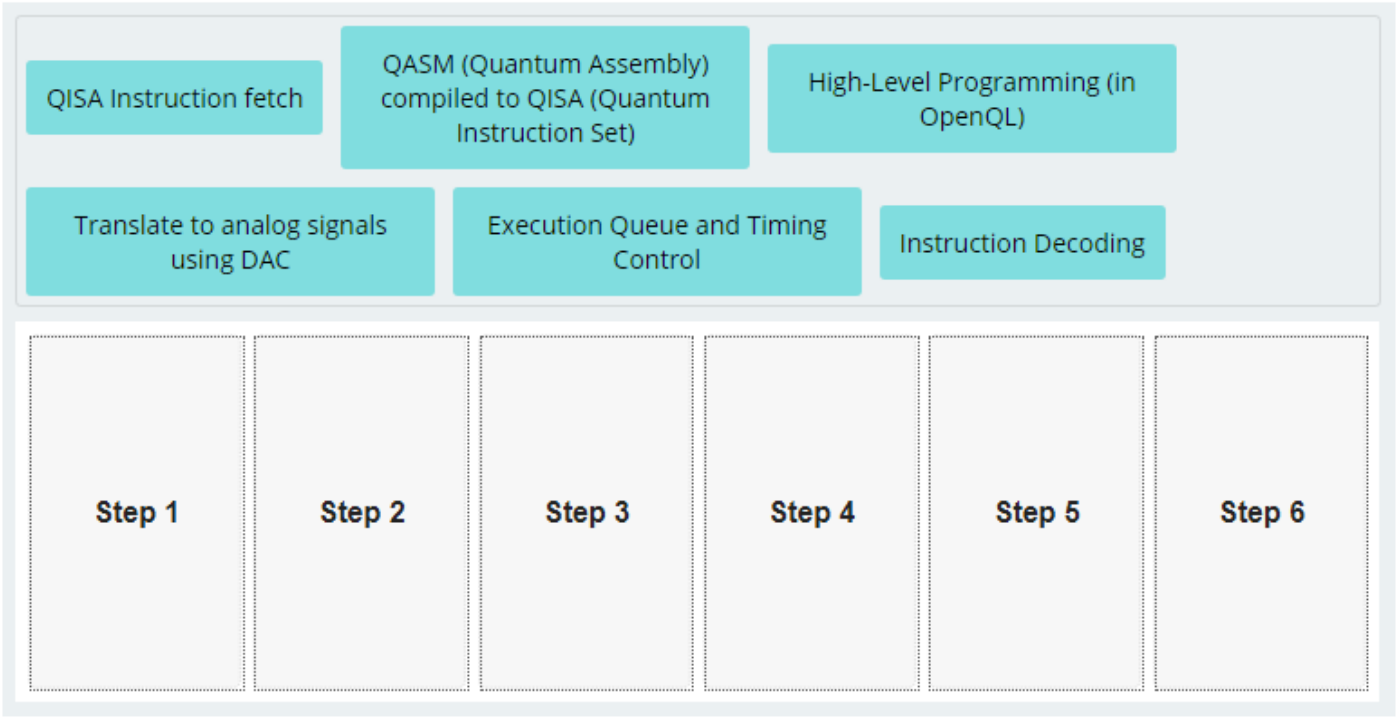
Which of the following initial placements would result in the fewest SWAP gates in order to run the algorithm from the previous question?

- [q5,q3,q1,q4,q2]
- [q2,q4,q1,q3,q5]
- [q1,q5,q3,q2,q4]
- [q4,q3,q5,q1,q2]

Practice question 7: Steps of execution in microarchitecture

PROBLEM

Drag and Drop the boxes according to the steps of their execution.



The following three questions are related to Module 3: Quantum algorithms & Quantum Error Correction.

Practice question 8: Controlled-Z

0 points possible (ungraded)

Recall the definition of the controlled-phase, or CZ gate: $CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$.

Suppose that this gate is applied to two qubits, 1 and 2.

Which of the following statements is true?

- Qubit 1 is the control qubit, and qubit 2 is the target qubit.
- Qubit 2 is the control qubit, and qubit 1 is the target qubit.
- The controlled-Z gate has no well-defined control or target qubit.

Quantum Fourier Transform

The Quantum Fourier Transform, F_N of dimension N is given by the following matrix:

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega_N^{0\cdot 0} & \dots & \omega_N^{0\cdot(N-1)} \\ \vdots & & \dots \\ \omega_N^{(N-1)\cdot 0} & \dots & \omega_N^{(N-1)\cdot(N-1)} \end{pmatrix}$$

where, \omega_{N} = e^{i\frac{2\pi}{N}} is the N^{th} root of unity.

Given below are four options. Select the correct option for the following question.

(A) $F_4 = \frac{1}{2}\begin{pmatrix} 1 & i & 1 & 1 \\ 1 & -1 & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$ **(B)** $F_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$

(C) $F_4 = \frac{1}{2}\begin{pmatrix} 1 & -1 & 1 & 1 \\ 1 & i & -1 & -i \\ -1 & -1 & -1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$ **(D)** $F_4 = \frac{1}{2}\begin{pmatrix} 1 & -i & 1 & -1 \\ 1 & i & -1 & -i \\ i & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$

Practice question 9: Quantum Fourier Transform

Which one of the matrices above is equivalent to F4, the quantum Fourier transform on two qubits?

- A
- B
- C
- D

Practice question 10: Shor's 9-qubit code - logical operators

A few months after proposing his famous factorization algorithm, Peter Shor proposed the first quantum error correcting code often called the 9-qubit code, this code has codewords as the following states :

$$|0\rangle_L = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$$

$$|1\rangle_L = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)$$

We would like to implement a logical X gate, which maps the |0⟩L state to the |1⟩L state and vice versa.

Which of the following operators will accomplish this?

- [ ] IIXIIXIIX
- [ ] IIZIIZIIZ
- [ ] ZZZZZZZZZ
- [ ] XXXXXXIII

The final six questions of this exam are related to Modules 4 & 5: Quantum Internet, Quantum Key Distribution and entanglement distillation.

Practice question 11: Quantum channels and their attenuation lengths

Alice and Bob want to communicate over a lossy quantum channel 400 km long, with an attenuation length of 20 km. Alice thinks she can overcome the losses thanks to her ability to locally generate entanglement at a rate of 10 GHz. As a photon is sent to Bob, she immediately starts to generate more entangled pairs locally.

On average, how long would Alice and Bob need to wait to have an entangled pair?

If the length of the channel is L, and the attenuation length is La, the transmission probability is 10−LLa.

- ○ 1 year
- ○ 173 years
- ○ 317 years
- ○ 756 years

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 12: Fidelity of quantum states

In module 5, we discussed the concept of fidelity as a measurement of the similarity of the two quantum states. In the case of two pure states $|\psi\rangle$ and $|\varphi\rangle$, the fidelity is defined as: $F=|\langle\psi|\varphi\rangle|2$.

What is the fidelity of the two states: $|\psi\rangle=|0\rangle+|1\rangle2\sqrt{}$ and $|\varphi\rangle=13\sqrt{}|0\rangle+2\sqrt{}3\sqrt{}|1\rangle$?

- 0.97
- 0.81
- 0.42
- 0.38

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 13: Byzantine agreement

Byzantine Agreement is a problem in multi-party communication, where the goal is to decide the value of a bit in presence of possible faulty nodes, or nodes that may transmit false information.

Using quantum resources, how does the number of rounds necessary to reach an agreement scale with the number of nodes n?

- it does not scale, it is a constant.
- $n--\sqrt{}$
- n
- n2

Practice question 14: Quantum repeater

What is a quantum repeater needed for?

- To constantly repeat Alice signal until Bob takes the time to measure it.
- To duplicate quantum states such that more than 2 parties can exchange entangled qubits simultaneously.
- To create superpositions between distant parties.
- To increase the maximum length over which entanglement can be distributed using lossy channels.

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 15: Communication over noisy channels

The transmission channel used for quantum key distribution is noisy. Alice and Bob may be unaware of this, and consider their channel to be perfect.

If Alice and Bob make this mistake, what is likely to occur in QKD?

- For attempts in which their bases correspond, they can still have mismatched bits, as if an eavesdropper were present.
- Alice and Bob will not compare their bases, as this is not necessary for a perfect channel, which they imagine they have.
- Entanglement will not remain over long periods, and Bob will have to constantly recalibrate his detector.
- Since a noisy channel does not allow superpositions, their communication will be entirely classical.

You have used 0 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Practice question 16: BB84

Consider the protocol BB84, discussed in module 5. Suppose that, on average, Alice and Bob do 2350 runs every time they want to communicate a secure key. Of the bits that remain after comparing their bases, they decide to always use 100 randomly selected bits to check for an eavesdropper.

What is the average length of the bitstring that Alice and Bob have left to use for their secure key if their communication has not been eavesdropped?

Take into account that both Alice and Bob have their bases picked randomly.

Try using the quantum computer yourself!

Congratulations! You have now finished the course Building Blocks of a Quantum Computer - part 2.

Try it yourself!

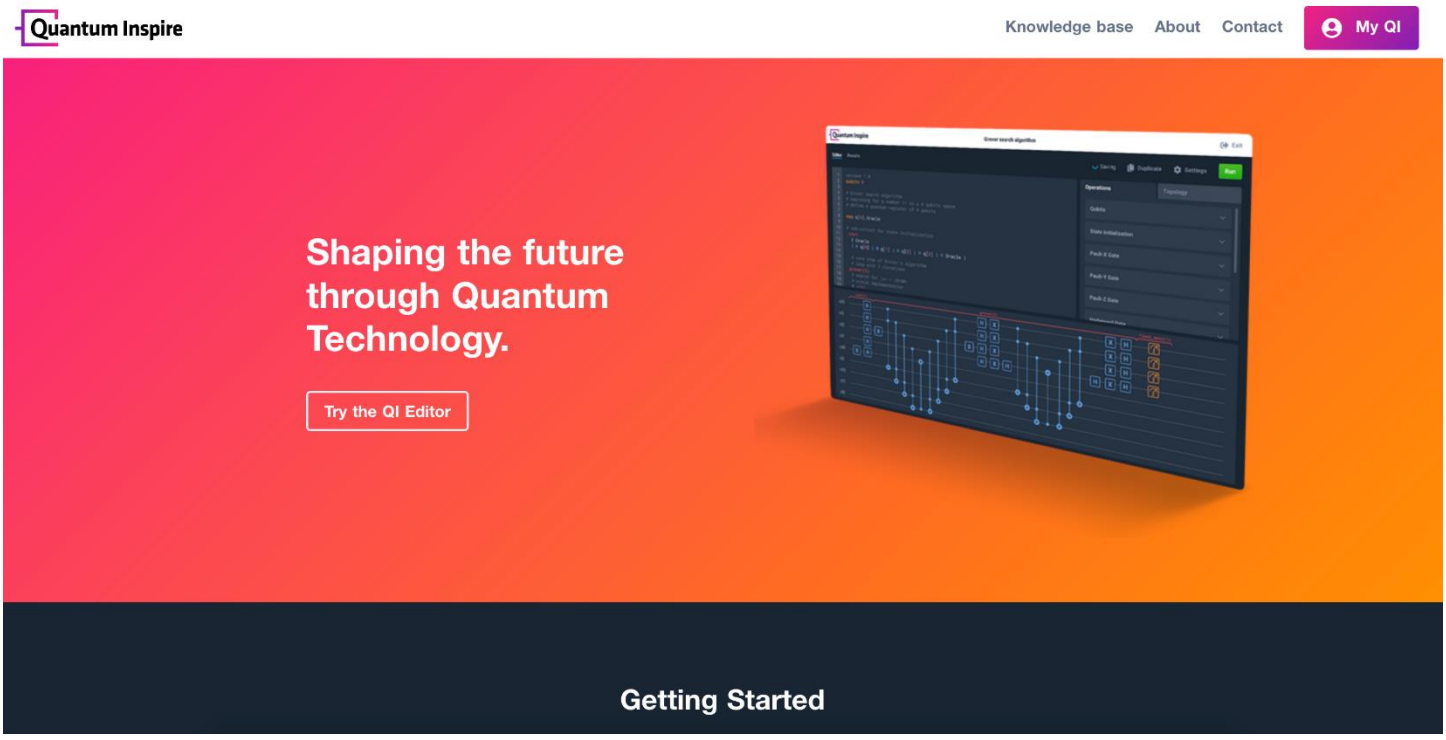
Now you have learned about all layers of a quantum computer, you might want to try running an operation on one. QuTech has launched Quantum Inspire. This quantum computing platform is designed to support your first steps into the fascinating world of quantum computation and to help you explore the opportunities that the power of quantum technology will offer in the future. A great chance to put to practice what you've learned in the courses on the Building Blocks of a Quantum Computer!

Running Grover's algorithm in Quantum Inspire

Quantum Inspire provides users a variety of ways to program quantum algorithms, execute these algorithms and examine the results. It provides a graphical interface to program in QASM (Quantum Assembly Language) and to visualize operations in circuit diagrams. With the Quantum Inspire Editor you can write quantum algorithms with support of automatic syntax checking. The Quick Guide explains how. The output of a quantum algorithm can be examined using our built-in data viewer. Users can also download the raw output of quantum algorithms for more detailed analysis and examination of results. If you wish to get started, this Section explains how to run Grover's algorithm in Quantum Inspire might be a first great exercise.

If you would like to continue your learning about quantum technology, we invite you to try our EdX course Quantum Cryptography. In this course, you will learn how to use quantum effects, such as quantum entanglement and uncertainty, to implement cryptographic tasks with levels of security that are impossible to achieve classically.

Good luck!



Reference

- https://www.edx.org/course/architecture-algorithms-quantum-computer-internet