Title

Summary of the talk

Since we are still using classical devices for the computation, I will begin with a reminder of the classical logical operations using classical bits.
Then I will introduce the quantum bits (qubits) together with some basic notions of quantum mechanics.
I will continue by showing how qubits can be manipulated and how errors are introduced during this process.
Then I will explain the quantum gates and their combinations in circuits.
I will give en example of quantum circuit and algorithm applied to the teleportation of a single-qubit quantum state.
For the discrete Fourier transform I will show the quantum approach using a more complicated circuit.
I will explain one of the methods for doing cryptography using qubits.
And finally I will try to answer the question of interest “how can we program a quantum device ?”, with an example of the public resources of the IMB Quantum Experience project.

Semiconductors

The idea of using Boolean algebra together with switching devices in circuits is old (C. Shannon, 1930) and its first implementations were massive, noisy, energy-consuming electro-mechanical devices.
The emerging of the modern era of computing was conditioned by two main factors : miniaturization and energy consumption. Both found a solution with the discovery of the transistor based on bipolar junctions.
There are underlying physics phenomena which determine the working of a transistor, described for instance by the Ebers-Moll equations, but for most of the computer scientists this is a hidden information that they do not need to understand or manipulate in order to successfully exploit a computing machine.
The standardization of the components of a computing machine has reduced the complexity of their engineering and functioning into technical data sheets with only the relevant information for the final user: capacity, speed, latency, power consumption, thermal characteristics etc.

The flip-flop circuit (a bi-stable circuit)

This is a bi-stable circuit (or a flip-flop) built with two transistors: on wikipedia the figures are animated, so you can action the two switches and see how the state of the current flow changes.
This is a basic electronic element that can store one bit of information, 0 or 1, represented by one of the two stable states of the circuit.
Of course, a precise definition (standard) of the two “bit states” is necessary: this can be the two levels of a voltage or two frequencies of an oscillating signal.

Inverter ($0 \Rightarrow 1$, $1 \Rightarrow 0$) with transistor-transistor logic (TTL)

A “voltage level bit” inverter is show here: when the base of the transistor is not polarized, all the source voltage drops on the diode, which lights up.
When the base is polarized, such that there is a current flowing between the B and the E, the transistor opens and the voltage applied to the diode drops, turning it off.
Few remarks:
- the transition between the input and the output is not instantaneous, it takes some time until the new working point of the transistor is reached
- the values of the components of this circuit and of the voltage level of the bit can be chosen such that there is a net distinction between the two states of the diode, without any ambiguity
- there should be no intermediate states between 0 and 1, any anomalous value of the diode state will be necessarily interpreted either as 0 or 1

Elementary logic gates: one-bit logic gates

The logic gates can be seen as functions between two sets with elements {0,1} and dimension corresponding to the number of inputs and outputs of the gate. In the case of one-bit gates, this dimension is 1 both for the input and for the output.
The only possible gates in this case are the (trivial) identity and the negate (to which we can add the FANOUT or COPY gate).
The maximum number of one-bit logic gates is 4, but two of them are just constant functions. So we are left with 2 (one being the negation of the other).

Elementary logic gates: two-bit logic gates

For the two-bit logic gates, the dimension of the input is 2 and of the output 1. There are in total 16 possible functions, but half of them are the negation of another function, so we are left with only 8, of which one is the constant function. From the 7 left, only 5 are of interest.
These 5 functions are shown with their gate symbols (using the European convention), their truth table and their Boolean arithmetic expression.

AND, OR, eXclusive-OR,

(next slide) Negated-AND and Negated-OR

A circuit for computing the sum (with carry bit)

This is an example of how to combine one-bit and two-bit logic gates into a circuit which performs the binary addition of two fixed length numbers.
Two AND gates, one OR gate, two XOR gates and four FANOUT gates are combined in a “cell” (left figure) which takes as input the two bits at the same position of the two operands and the “carry” bit from the previous operation (except for the LSB, of course).
By chaining several of such “cells” we can build a full adder for fixed length numbers (right figure).

Universal (classical) gates

In the most general case of logic gates having as input “m” bits and producing “n” bits of output, the corresponding function can be expressed using some of the previously introduced elementary gates. In the extra slides you find the proof of this, using the set of elementary gates {AND, OR, NOT and FANOUT}.
An even smaller set is found using only two gates: {NAND, FANOUT} and taking into account the De Morgan identities.
We will call this a universal set of gates for the classical computation.

Classical reversible computing

In order to make a step further to the understanding of the quantum approach of computing, we have to make the following remark: the laws of quantum mechanics (with some exceptions) and more general the laws of physics are reversible in time, and so has to be our quantum computing device.
But the two-bit classical logic gates presented before are not reversible: each computation is loosing one bit of information (dissipated into the surrounding medium as equivalent energy).
This dissipation limit does not worry yet the constructors of integrated circuits, because there are other sources of much higher energy loss.
In the classical computing, it is nevertheless possible to “embed” any irreversible function into a reversible function, with the cost of adding additional bits (to equalize the dimension of the input and the output spaces). Those bits do not change their values between the input and the output, they have only a control role and are called ancillary bits.

A simple reversible classical gate: the controlled-NOT (CNOT)

Using the irreversible XOR gate and by adding one ancillary bit we can obtain a 2-bit-to-2-bit new gate which is reversible. The ancillary bit has only a control role, that is why this gate is called a controlled gate. In fact, this gate performs the negation of the active bit "b", only if the controlled bit "a" is 1, otherwise it is just an identity of "b". For this reason it is called a controlled-NOT gate.
There is a variant of this gate which performs the negation on "b" only if "a" is 0.

The circuit representation of the classical CNOT gate

Let’s have a closer look at the graphical representation (a sort of Feynman diagram) of this gate:

- the input and the output are vectors of classical bits, starting, as a convention, with the LSB from below

- the evolution follows the time line from left to right

- the application of this gate occurs at some intermediate moment of time between the input and the output

The controlled-NOT gate has the important property that it is equal to its inverse. Also, if the target bit is set to “0”, then both output bits will have the value of the ancillary bit “a”, so the gate performs a FANOUT gate.
It is possible to show (Preskill 1998) that two-bit reversible gates are not universal gates, so they are not enough for universal computation.

Three-bit reversible gates: the Toffoli gate

Instead, if we build the tree-bit reversible gate known as the controlled-controlled-NOT gate (or the Toffoli gate) it is possible to obtain from it the elementary gates NOT, AND, OR, which, together with FANOUT, form a complete set for universal classical computation as we have seen before.

Quantum bits (qubits)

A qubit is first of all a system which behaves following the laws of quantum mechanics (therefore it has to be microscopic). Since we stick to the idea of doing the computations in the lowest numerical base, 2, this system is chosen such that it has only two possible states. We associate to those states the two classical bit values 0 and 1.
The principles of quantum mechanics say that the most general state of our qubit is a linear superposition of the two basic states, which themselves are orthogonal to each other (as we want them to be, in order to have the maximum separation between the two “qubit values”).

The 1st postulate of quantum mechanics

Quantum mechanics is a very formalized theory, so it is important to understand the postulates which describe how the reality “should” look like for a system which is very-very small.
In the most general case, the interaction with the environment of a quantum system described by its state vector “psi” is modeled by a linear operator (the Hamiltonian) which is sufficient to predict the future evolution in time of the system, if we are able to solve the associated Schrödinger equation.
If we look at the decomposition over the computational basis, we can say that, actually, the two complex coefficients of the linear decomposition vary in time.
The Planck constant, appearing in the Schrödinger equation, gives by its very small magnitude a good idea about the energy scale of the evolution of a quantum system and of the microscopic phenomena in general.

Vector algebra with qubits

Our qubit can be seen as a vector evolving in a 2-dimensional vector space, which accepts several elementary operations, like the multiplication with a complex scalar, the sum and the product with their adjoint vectors (obtained from turning column-like to row-like vector representation).
The direct representation of a vector is called in the quantum formalism a “ket” vector and its adjoint a “bra” vector.
The complex coefficients of a state vector (like “alpha” and “beta”) will take their complex conjugate values when the vector is transformed in its adjoint image.

<u>The 2nd and 3rd postulates of quantum mechanics</u>

If we want to give a real usage to our qubits, we need first of all to be able to measure them. The second postulate of quantum mechanics defines what a measuring operation is in the quantum world, since we have seen that the system is in a state which is in general neither pure “0” nor pure “1”.
For the “pure” vectors of the computational basis it is rather simple: there is a particular measurement, associated with a particular operator, which applied to a system which is in one of those states, leaves the system in the same state, up to a multiplication factor which can be either “+1” or “-1”. We call these states the eigen-states (or vectors) and the two factors the corresponding eigen-values of that operator.

We can express this particular operator (known as the “z” or the 3rd Pauli operator) in a matrix form and calculate the action on a state vector as the result of the multiplication between this matrix and the column-like vector representation.

The probability of a given measurement outcome

Since this operator corresponds to an observable, a real property of that system, we may ask the following question: in the case of a generic state of a system (a combination of the two pure states), what is the result of one single measurement? The 4th postulate tells us that the result is not exactly predictable, but it is rather expressed by probabilities to obtain one or the other value, and that those probabilities are related to the square moduli of the two complex coefficients, α and β.
An important observation is that global any phase factor (complex numbers with modulus one which multiply the state vector) does not change the prediction of a measurement, understood as the probabilities to have a given outcome.

<u>The quantified spin and the choice of … (1)</u>

We can go further and say that up to now we were talking about a specific physical quantum property of many systems, which is the “spin” (we can imagine it as a spinning movement around a given axis). When we talked about the computational basis corresponding to the two qubit states, we were actually referring to the situation where we measure our qubit along a chosen axis, let’s say the z-axis of a Cartesian system of coordinates.

The quantified spin and the choice of … (2)

But we can equally chose another axis to do the measurement, let’s say the x-axis. In this case, either we rotate the figure in order to bring the x-axis to the vertical and we fall into the previous case, or we introduce a new operator (the 1st Pauli operator, sigma-x) whose matrix, expressed using the same computational basis as before, is shown on this slide.

The quantified spin and the choice of … (3)

And since in the Cartesian space we completely define a geometrical object by three orientations, we can do the same for the other axis, the y-axis, to which we find the corresponding Pauli operator sigma-y.

The quantified spin and the choice of … (4)

In this way, we obtain the three Pauli operators, which are very useful in expressing any arbitrary rotation in the 3-dimensional space.
As for the most general state of our spinning system, the picture becomes more complicated, since it is up to the experimenter to chose which axis he will use for doing the measurement of the spin.

The eigen-vectors of the Pauli operators

Here we write explicitly, using the same computational basis (the eigen-vectors of the sigma-z Pauli operator), the eigen-vectors of the other two Pauli operators: this is easy to check, using the matrix representation of the operators given in the previous slides.
These notations indicate the axis, “x”, “y” or “z” and the orientation of the eigen-vector along that axis, “plus” or “minus”.

The circuit symbol for a measurement

In the graphical representation of a circuit, when we want to show that we do a measurement on a qubit (in general this is the end of the “time line” of that qubit) we use a special symbol where we specify which kind of measurement we do, by noting, for instance, the axis of the Pauli operator: x, y or z (or 1, 2 and 3).
The result of the measurement is a bit of “classical” information, represented here by a double line.

The 5th postulate of quantum mechanics

There is an important feature of the behavior of a qubit after a measurement which shows a fundamental difference from the classical case, with many consequences for the quantum computing.
If we try to extract the value of one observable by doing a measurement on a qubit, we destroy the initial state of the qubit, which collapses into the state described by the eigen-vector corresponding to the value of the observable obtained as outcome of this measurement.
This is somehow counter-intuitive with respect to the classical approach, because it can lead to the idea that we can never have the full description of the initial state from measurements performed on the system.

Before the measurement of the z spin component

Let’s suppose that before the measurement our spin qubit was in some generic state, a completely unknown superposition.

After the measurement

Let’s say that we arrange our experimental setup such that our measurement is along the z-axis. According to the 2nd postulate, we will get the spin value along the z-axis “+1” or “-1”, with probabilities from the 4th postulate. In addition, the 5th postulate tells us that the system state will collapse into one of the base states.

The Stern-Gerlach experiment

Such an arrangement was tested experimentally by Walther Gerlach in 1922, following an idea of Otto Stern from 1921.
He let a beam of particles with spin to pass through a region with a strong magnetic field gradient, which produced the split of the main beam in two secondary beams whose impacts he recorded on a detector plate.
The two spots corresponded to the particles which collapsed in either one or the other spin state shown in the previous slide. The passing of the particles through the magnet field and their subsequent detection can be seen as a measurement of the spin along one axis (the z-axis, by convention).

Selection of one z-state

Next, we put a hole in the detector screen allowing only the passage of particles being in a given z-state, let’s say the spin “+1”, and we guide this secondary beam through a similar magnet as the first one, but which we have turned around the beam axis with 90 degrees, such that now it corresponds to the x-axis of the coordinates system.

After the second measurement

If we use a second detection screen to record the impact of the particles, we notice that we have again two spots, corresponding this time to a split of the incoming beam according to the two eigen-states along the x-axis.

Selection of one x-state

If we repeat the trick with the whole in the detection screen and let pass only one part of the split beam (with x-spin “+1”), guiding it further through a similar magnet but rotated back to the initial position, then we have a result difficult to comprehend, but which is in perfect agreement with the principles of quantum mechanics.

After the last measurement

After this last measurement, along the z-axis, we will see again that the population of particles in the beam is distributed between the two states along the z-axis, although in the first selection we have eliminated the particles with negative spin along this same axis.

The no-cloning theorem

Before talking about manipulating qubits, we must insist on a fundamental property of the qubits as quantum systems which forbids the integral copy (or cloning) of a generic state of a qubit. This would correspond to the FANOUT gate from the classical case.
This not only a limitation, but it has important consequences, as we will see in the discussion about the quantum cryptography.
The impossibility of the cloning is also a consequence of the universal validity of the uncertainty relation of Heisenberg, which is one of the pillars of the interpretation of the quantum reality.

Flipping a qubit using a constant magnetic field

A qubit changes its state if there is an action from outside and the evolution of the state vector (or wave function) is described by the Schrödinger equation (the 6th postulate).
In other terms, we can describe the state vector at some time “t” as the result of the application of an operator (called the time-evolution operator) upon the state vector at time “zero”.
If the action on the qubit does not depend on time (the Hamiltonian operator is constant), then the time-evolution operator has a simple expression and shows a very important property which is the unitarity.
For a spin interacting in a constant magnetic field, the Hamiltonian operator is proportional to the scalar product between two vectors: the magnetic field intensity H and the one formed by the three components of the particle spin.

Flipping a qubit with a constant magnetic field (cont.)

Using some additional notations, we can write the matrix expression of the time-evolution operator. The time dependence has been now isolated in the linear function “alpha”, which appears as the argument of trigonometric functions. This suggests that the resulting movement of the particle with spin in a constant magnetic field is just a rotation.
The unitarity of the time-evolution operator, introduced in the previous slide, can now be interpreted as the property to conserve the angle between two vectors when the same rotation is applied to both of them.

Flipping a qubit with a constant magnetic field (cont.)

If the magnetic field is oriented along the x-axis, the expression of the time-evolution operator simplifies.
In particular, we can ask to make the state vector “zero” evolve into the state vector “one” by using this operator. If we do the matrix multiplication and solve the system of equations, we find that this is possible only for a special value of the “time” (actually an infinity of values, do to the periodicity of the trigonometric functions).
We have obtained the equivalent of the classical gate NOT applied to the “zero” state vector of a qubit.

A first observation must be done: this “transition” between the two states is obtained with a very precise tuning of the magnetic field and its action time. In practice this is rather difficult to achieve, for several reasons. Therefore, it is difficult to obtain a “clean” change of a qubit state as in the classical situation, where the errors are extremely rare and, of course, can have only a binary nature.

Unitary errors

A quantum algorithm contains in general sequences of unitary operators applied to one qubit. We stay in the case of the strict unitarity of all operators, which means that we neglect any uncontrolled coupling to the environment.
As we have seen in the case of the qubit flip in a pulsed constant magnetic field, it is very difficult to obtain experimentally the right values of the magnetic field and of the pulse length. But even if we are wrong, the result will still be a unitary operator, only the final state vector will not be the desired one.
We can express the action of this new operator “Vi” by introducing a residual vector “Ei”.

Unitary errors (cont.)

If we write the product of the operators leading to the final state vector but using those “error affected” operators, we can proceed by recurrence in order to express the final state vector as the sum of the desired final state vector and several terms. In these terms, products of several unitary operators Vi are applied to the residual vectors Ei.

<u>Unitary errors (cont.)</u>

Since a product of unitary operators is also unitary and by applying it to a vector we don’t modify the amplitude of that vector (it is just a rotation), we can limit the errors given by the residual vectors Ei by a single upper bound value ε. If we calculate the overall error on the final state vector, we see that it increases linearly with the number of operations, while in the classical case, we know that this increase is driven only by the square root of the number of operations.

Single-qubit gates (Pauli operators)

With the Pauli operators we have up to now introduced three single-qubit gates. This is the table of their action on the vectors of the computational basis corresponding the the sigma-z operator.

The Hadamard gate

Another single-qubit gate is the Hadamard gate. This gate transforms the two vectors of the computational basis of the z-axis Pauli operator into the ones of the x-axis Pauli operator. The symbol is the capital letter H inside a box. Generally, a single-qubit operator will be drawn in a circuit by enclosing its symbol inside a box which is placed on the time line of that qubit.
We also say that the Hadamard operator transforms a pure state of the computational basis into a uniform superposition of states.

The exponential power of the states superposition

Here is one example of Hadamard gates in action, which illustrates an important advantage of computing with qubits.
If we start with three qubits, all in “zero” state, organized in such a way that we can “simultaneously” apply a Hadamard gate on each of them, the final state of the three qubits system will be expressed by the direct (or tensor) product of the three single-qubit state vectors.
This state is a superposition with equal weights of the eight basis vectors of the system of three qubits. If we look at the composed indices as binary representations of the numbers from 0 to 7, we may say that here we have stored on 3 qubits the eight integers from 0 to 7.
This exponential power of the data storage using qubits is one of the advantages of the quantum computing.

The generic state of a qubit in spherical coordinates

Instead of working with the two complex coefficients “alpha” and “beta”, there is an alternative representation using spherical coordinates. Here we take into account that the sum of the squared moduli of the coefficients is one (probability normalization) and that we can consider the first coefficient as being real, since a global (complex) phase has no physical significance.
This allows a simple interpretation of the transformation of a qubit (by mapping it on the Bloch sphere) and lets us introduce another single-qubit operator.

The phase-shift gate

The phase-shift gate, as the name suggests, when applied to a generic state, only adds up a phase angle, seen in the complex part of the second coefficient in the spherical coordinates representation. Its symbol is R (rotation) with the indication of the axis and of the value of the rotation angle.

Universality of Hadamard and phase-shift gates

We can now formulate a first theorem which has no equivalent for the classical bits: any unitary operation on a single qubit can be constructed using only Hadamard and phase-shift gates. As a consequence, we can reach any generic state by using only Hadamard and phase shift gates and starting from the “zero” state of the computational basis. It is important that we represent the state using the spherical coordinates “theta” and “phi”.
Nevertheless, in practice, preparing an arbitrary state is not so easy to do.

Two-qubit states and gates

Two qubits can be described by a vector space with dimension 4, so a generic state will be characterized by four complex coefficients.
Notice that for more than one qubit, for the vectors of the computational basis we can use different notations, those three from this slide indicating explicitly the single-qubit states entering the tensor product.

The quantum CNOT gate

This is the quantum equivalent of the controlled-NOT gate, whose operating mode is obvious. The circuit diagram is the same as in the classical case. The XOR operation is applied to the symbols “0” and “1” associated to the vectors of the computational basis. We give also the matrix representation of this gate which acts on a tensor product of single-qubit 2-dimensional vectors.
As in the classical case, we also have the version of this gate where the target qubit flips only if the control qubit is in the “0” state.

Obtaining a SWAP gate from CNOT gates

One trivial but sometimes useful gate is the SWAP gate, which switches the positions of two qubits. We can obtain it as shown here, using three Hadamard gates.

With the controlled-NOT gate we can introduce a very important aspect of the quantum world, which is the possibility to entangle the states of two quantum objects. We do it here by applying the CNOT gate on two qubits, one qubit which is in a generic states and a second which is in the "0" state. From the distributive property of the operator over the sum of vectors, we obtain the expression on the right side, which is one example of entangled state, meaning that we can not separate this state in a product of single-qubit states.

Universal quantum gates

Just like in the classical case, we are looking for a small set of quantum gates which is universal, that is, it allows to build any other unitary operator we can imagine, acting on any number of qubits we want.
The answer to this is that such operator can be decomposed in one-qubit gates and two-qubit CNOT gates.
As for the one-qubit gates we have already seen that the Hadamard and the phase-shift operators form a universal set of gates.

In order to find a solution to this, we need to introduce special gates like the controlled-U gate, where U is an arbitrary unitary operator, with a generalized version, the k-times-controlled-U.

In particular, when the U operator is the NOT operator and we have two control qubits, we get the Toffoli gate.

Universal quantum gates (cont.)

For the Toffoli gate we have a solution: it can be implemented using Hadamard gates and some special single-qubit operator V (with his controlled version), which in turn can be obtained from Hadamard and phase-shift gates.

Universal quantum gates (cont.)

The full proof is more complicated, but it is summarized here.

- a generic operator U on n qubits can be decomposed by means of k-times-controlled-U gates

- any k-times-controlled-U gate (with $k > 2$) can be decomposed into Toffoli and controlled-U gates

- the Toffoli gate is obtained from CNOT, controlled-V and Hadamard gates (V is also a unitary operator); we have a similar conclusion for the 2-times-controlled-U gate

- the controlled-U operation can be decomposed into single-qubit operators (A, B, C) and CNOT gates

Quantum information, teleportation

(see slide)

Quantum information, teleportation (cont.)

The entanglement property of quantum objects allows nevertheless to do the teleportation. The only condition is that Alice and Bob, in some way, at the beginning of the experiment, share a pair of qubits which are set in an entangled state, and each will have his half of this pair to take along.

In this example, the pair of qubits “q0” and “q1” is in the initial state “10” (starting from the LSQB) and after the application of a Hadamard and a CNOT gate the result is what is known as a Bell pair of (entangled) states.

Quantum information, teleportation (cont.)

One thing to mention about qubits and circuits is that in order to apply multi-qubit gates there has to be a physical connection between those qubits, which in the information technology language means that they have to be in the same register.
So, Alice’s qubit and the two qubits of the Bell pair must be at the beginning in the same register.
The 3-qubit state vector will be expressed like this, in the computational basis of the 3-qubit space.
For the next operation we need to change the vector basis in which this state is expressed, because Alice wants to do a measurement in the Bell basis, not in the computational basis. As we have seen before, if we do a measurement of a spin along a different axis that “z”, we have to change the representation of the spin state into the new basis, in order to extract predictions about the outcomes of this new measurement.

Quantum information, teleportation (cont.)

At this moment Bob can get away with his half of the Bell pair because Alice will not touch it anymore. He will wait “news” from Alice and do nothing until.

On her side, Alice applies another two gates to her pair of qubits, reversing the ones used to create the Bell pair. Because “q0” and “q1” are entangled, the action of these last operators will affect all three qubits and we will see how, by magic, the “q0” qubit (Bob’s qubit) contains now different “versions” of the initial qubit from Alice.
The two qubits of Alice are found to be in pure eigen-states, which will simplify the measurements she will do in the computational basis, this time.

Quantum information, teleportation (cont.)

The last contribution of Alice is to do a measurement in the computational basis on the two qubits in her possession and send the values obtained (two classical bits of information) via a classical transmission channel to Bob.
At this point Alice does not have any more the initial quantum state of her qubit, but this is OK, since this state will soon be recreated on Bob's side, using his qubit and the two classical bits.

Quantum information, teleportation (cont.)

The only thing Bob has to do is to select one operator (one of the three Pauli operators or the identity) from this table, according to the values of the two classical bits sent by Alice. When this operator will be applied to his qubit (his half of the Bell pair), he can be sure that this will evolve into the initial state of the qubit which Alice wanted to send by teleportation. End of story.

The Fourier Transform (FT), continuous and discrete

The Fourier transform is an integral transform acting on functions of a real variable, in general associated with time. The result is a complex function defined on a domain called the frequency domain. Its inverse is also an integral transform which recovers the initial function defined on the time domain.

In the case of discrete values, such as those obtained from the sampling of a continuous signal, the integrals must be replaced by finite sums.

The discrete quantum Fourier transform (QFT)

Suppose we have “N” complex values stored as the elements of a vector “f” and we want to calculate the elements “f-tilde” obtained by the Fourier transform of those “N” values.
To do this, we need only “n” qubits, where “n” is the logarithm in base 2 of “N”, which we build in a generic state where the “N” complex values “f” are the coefficients of the computational basis vectors (notice how the computational size is exponentially reduced).
Remember that a vector of the computational basis is a tensor product of “n” single-qubit vectors, and it is written here using the short notation with a single index.

The discrete quantum Fourier transform (cont.)

Next, we define a unitary operator “F”, by describing how it acts on each of the “n” vectors of the computational basis: the operator “F” applied to the vector “j” is here decomposed using the vectors “k” of the computational basis. For a generic state “psi”, if we replace the decomposition from the previous slide we obtain a superposition of the vectors of the computational basis, whose coefficients are exactly the discrete transform we were looking for.
So this is reduced now to the problem of finding the implementation of such a unitary operator “F”.

The discrete quantum Fourier transform (cont.)

Before, we must specify how we formally treat the product of the indices “jk” appearing in the definition of the operator “F”.
By taking into account that a general index “j” can be understood in his binary representation, whether it is larger that one or smaller than one, we can re-write the terms of the sum in order to express it as a tensor product where we have only single-qubit state vectors.

The discrete quantum Fourier transform (cont.)

(see slide)

The discrete quantum Fourier transform (cont.)

And this is the quantum circuit implementing the transform: it is using “n” Hadamard gates and “n(n-1)/2” phase shift gates.
This exponential acceleration, compared to the classical case, is not very effective, because the algorithm extracts information from the amplitudes of the state vectors and, as we have seen, it is difficult to prepare a state in a well defined generic state. Also, the final values are amplitudes of the state vectors which have to be evaluated from measurements, but one measurement destroys the state, so we loose information about the other coefficients. The power of the quantum algorithms consists in extracting useful information not from the amplitudes but from the wave function itself.
Nevertheless, the QFT is a key ingredient of exponentially efficient quantum algorithms, like the Shor algorithm for integer factorization.

The unbreakable cipher

The Vernam cipher “is a symmetrical stream cipher in which the plain-text is combined with a random or pseudo-random stream of data of the same length, to generate the cipher-text, using the Boolean "exclusive or" (XOR) function” (definition from wikipedia).
This class of ciphers has been proven to be unbreakable, under the following conditions (C. Shannon, 1949): “the key must be truly random, as large as the plain-text, never reused in whole or part, and kept secret.”
In this case the problem is actually shifted to the generation and transmission of a very large number of secret keys.

The BB84 quantum protocol (Bennett and Brassard, 1984)

The BB84 protocol uses two alphabets and a set of rules to code classical bits “0” and “1” into eigen-states of the two Pauli operators, sigma-z and sigma-x.
For instance, if the z-alphabet is used, the “1” classical bit value is coded in the “-1” spin state of the “z” Pauli operator. With the x-alphabet, it would be coded in the “-1” corresponding spin state of the “x” Pauli operator.

The first part of the BB84 protocol

(see slides)

The IBM Q project (launched in March 2017)

We are going to talk now about one implementation of quantum processors which can be used to build simple circuits with a small number of qubits.
The IBM Quantum Experience project was launched in March 2017 with the goal to provide access to quantum processors to a broad audience.
The processors are publicly accessible through the IMB cloud structure.
Each QX architecture has a small number of qubits configured according to a coupling-map, which defines on which pairs of qubits we can apply two-qubit gates and in which order: for instance, in the case of QX2, we can apply the CNOT gate between the qubits “Q0” and “Q1” having “Q0” as control qubit and “Q1” as target, but not the other way around.

Approximating continuous gates with discrete gates

This is the good place for an additional remark to the universality of the set of gates introduced earlier in this talk.
One of the gates necessary for describing any unitary operation on a set of qubits is the phase-shift gate, which, as we know, is a continuous gate (a rotation). Its practical implementation will therefore raise technical problems, for the reasons discussed before.
However, it is possible to approximate such a transformation with an arbitrary small accuracy ε using a discrete set of quantum gates. It is possible to show that using Hadamard and T gates (where T is a π/4 phase-shift around the z-axis) we can approximate any single-qubit rotation in a number of steps given by this formula.

The IBM QX processors can implement the single-qubit gates Hadamard and T and the two-qubit gate CNOT.

The programming language

For the programming of his quantum devices, IBM has developed a SDK named Qiskit, which can be installed in Python with one command. Qiskit has several components, or elements (here with their Latin names):
Terra is a set of tools for composing quantum programs at the level of circuits and pulses, optimizing them for the constraints of a particular physical quantum processor.
Aer provides a high performance simulator framework, with realistic noisy simulations of the errors that occur during execution on real devices.
Ignis is a framework for understanding and mitigating noise in quantum circuits and systems.
Aqua contains a library of quantum algorithms upon which applications can be built.

Running an example on IBM QX2

This is a first example to begin with from the Qiskit tutorial repository.
The circuit shown here starts with a register grouping the first 3 qubits of the QX2 device, prepared all in the initial state “zero”. After the application of a Hadamard gate on qubit “0” followed by two CNOT gates on qubits “1” and “2”, controlled by qubit “0”, we obtain a Greenberger-Horne-Zeilinger (GHZ) state.
This entangled state contains only the states “000” and “111”. The 6 remaining states corresponding to the tensor product of the three qubits are not present in the GHZ state.

Running an example on IBM QX2 (cont.)

The circuit is executed 1024 times on the QX2 device, starting each time from the "000" initial state and the classical bits resulting from the measurements are shown in a histogram as the contributions of the 8 possible final states formed with the three qubits.

Because it is a real quantum device, affected by errors, we see not only the two states of the GHZ state, but also the six others, with much smaller probabilities.

The histogram shows the frequencies of each of those states, together with the results from "Aer" simulations (without considering errors).

The Qiskit code

The Qiskit code is provided as a Jupyter notebook, since Qiskit has an application programming interface for Python.
With this piece of code we first build a quantum circuit with 3 qubits and place the three gates shown before which bring the “000” initial state into the GHZ state.
With the help of the matplotlib module we can obtain schematic drawings of this circuit.

The Qiskit code (cont.)

If we want to do only a simulation, then we need to specify what exactly this simulation will calculate, by choosing a “back-end”: in this case it is a simulator of the final states of the system of three qubits.
If we don’t specify, noise will not considered in the simulations, so there are no errors in the final states and the two allowed states appear with equal weights (1/sqrt(2)).

<u>The Qiskit code (cont.)</u>

If we want the simulation to be closer to the hardware architecture, then we have to use the OpenQASM back-end.
First we add to our circuit with three qubits three classical bits, which will be connected by measurement to the three qubits. To this we add the previous circuit where the gates have been placed.
The OpenQASM back-end is a language closer to the hardware description, which translates the API functions into a kind of assembly instructions. The example shows few instructions corresponding to the application of the gates and the measurement of our circuit.

The Qiskit code (cont.)

In this case, the simulation will finish with 1024 measurements, so (again without errors) we will read the states “000” and “111” statistically a number of times, in this case 515 and 509, respectively.

The Qiskit code (cont.)

And finally, we execute this circuit on a quantum processor. We need first to create an account (a free account has limited resources and low execution priority), login, see which providers are available corresponding to my user status, chose one of the devices and execute the circuit on this device. A monitoring module will take care of the submission of the job in the queue and allow to retrieve the output once the job finishes, from which we can extract the number of counts for each result of the measurement of the final state of the three qubits.