

Journées thématiques IN2P3 - Quantum computing: state of the art and applications

2-3 décembre 2019
IPN Orsay

Tutorial session on quantum circuits and quantum algorithms

December 3rd, 2019

using simulation software:

Quantum++ (with Eigen 3), author Vlad Gheorghiu
myQLM, Atos Quantum Lab

Elementary logic gates:
one-bit logic gates

$$f : \{0, 1\} \rightarrow \{0, 1\}$$

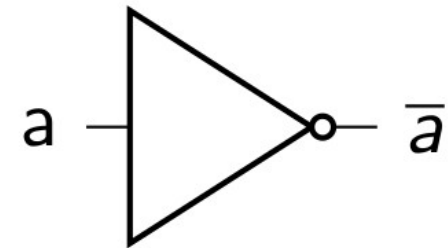
the identity:

$$a = a$$

the NOT gate:

$$\bar{a} = 1 - a$$

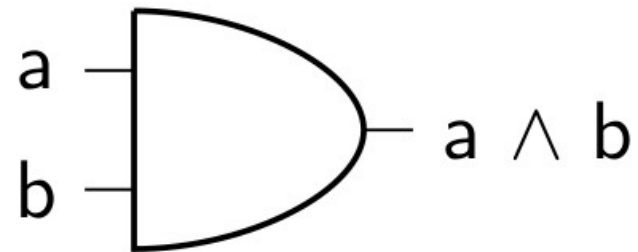
(in binary arithmetic)



a	\bar{a}
0	1
1	0

Elementary logic gates:
two-bit logic gates

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$



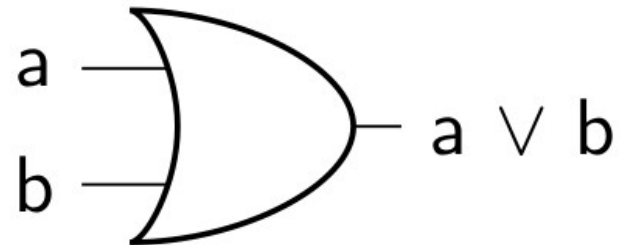
the AND gate:

$$a \wedge b = ab$$

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Elementary logic gates:
two-bit logic gates

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$



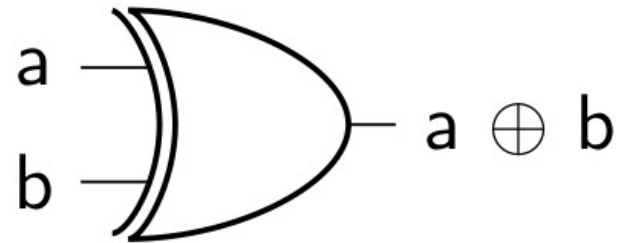
the OR gate:

$$a \vee b = a + b - ab$$

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Elementary logic gates:
two-bit logic gates

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$



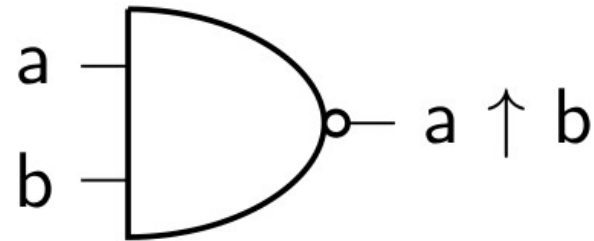
the XOR (exclusive OR) gate:

$$a \oplus b = a + b \pmod{2}$$

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Elementary logic gates:
two-bit logic gates

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$



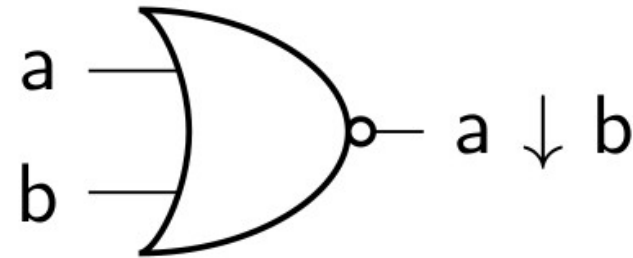
the NAND (negated AND) gate:

$$a \uparrow b = \overline{a \wedge b} = \overline{ab} = 1 - ab$$

a	b	a ↑ b
0	0	1
0	1	1
1	0	1
1	1	0

Elementary logic gates:
two-bit logic gates

$$f : \{0, 1\}^2 \rightarrow \{0, 1\}$$



the NOR (negated OR) gate:

$$\begin{aligned} a \downarrow b &= \overline{a \vee b} = \overline{a + b - ab} \\ &= 1 - a - b + ab \end{aligned}$$

a	b	$a \downarrow b$
0	0	1
0	1	0
1	0	0
1	1	0

How many one-bit and two-bit functions exist ?

one-bit = 4 functions

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

we have the constant functions:

$$f(x) = 0 \quad \forall x$$

$$f(x) = 1 \quad \forall x$$

$$f(a,b) = 0 \quad \forall a,b$$

$$f(a,b) = 1 \quad \forall a,b$$

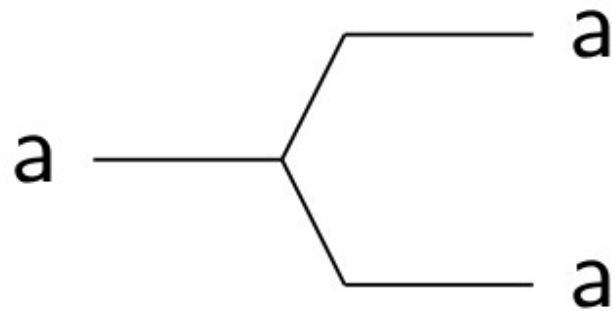
two-bit = 16 functions

a	b	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0

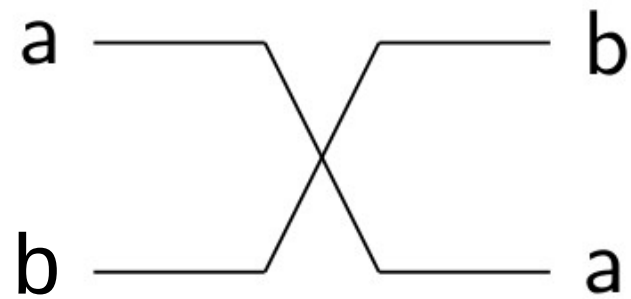
a	b	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Two more special gates, for building circuits

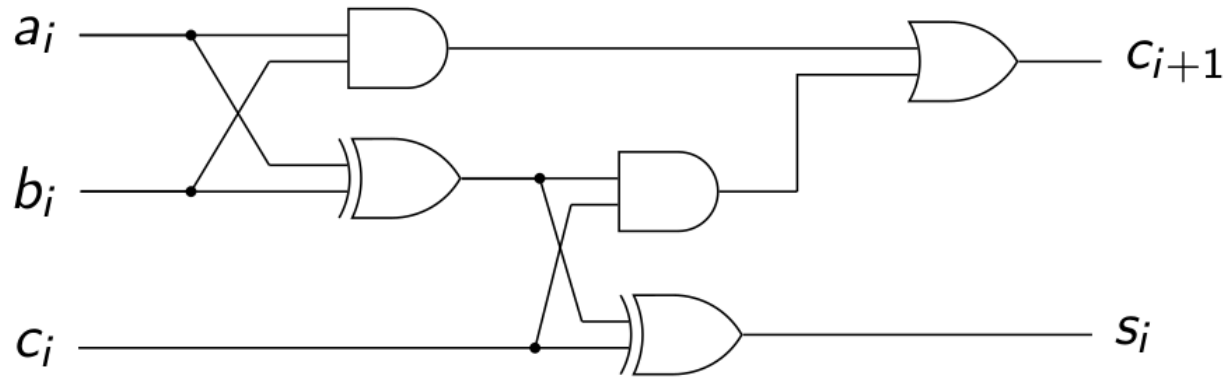
FANOUT (COPY)



CROSSOVER (SWAP)



A circuit for computing the sum with carry



Given the binary representations $a = (a_n, a_{n-1}, \dots, a_1, a_0)$ and $b = (b_n, b_{n-1}, \dots, b_1, b_0)$, the i -th bit of the sum is

$$s_i = a_i + b_i + c_i \pmod{2}$$

where c_i is the carry over from the sum $a_{i-1} + b_{i-1} + c_{i-1}$. The carry over is set to one if two or more of the input bits a_i , b_i and c_i are 1 and 0 otherwise. This circuit can be built with the following elementary gates: 2 AND, 1 OR, 2 XOR and 4 FANOUT.

Universal (classical) gates

Any function $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ can be constructed from the elementary gates:
AND, OR, NOT and FANOUT !

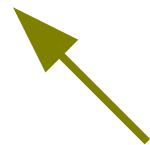
We say that AND, OR, NOT and FANOUT constitute **a universal set of gates** for the classical computation (see the proof in Appendix A).

A smaller universal set is **NAND and FANOUT**:

OR can be obtained from NOT and AND: $a \vee b = \overline{\bar{a} \wedge \bar{b}}$ (De Morgan's identities)

and NOT can be obtained from NAND and FANOUT:

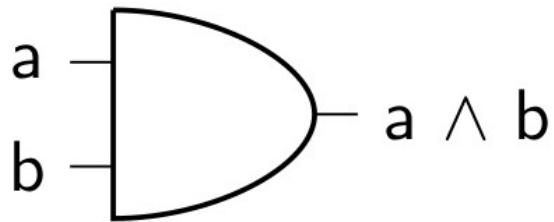
$$a \uparrow a = \overline{a \wedge a} = 1 - a^2 = 1 - a = \bar{a}$$



here we have FANOUT and NAND

Most of the logic gates discussed up to now are *irreversible*.

The Boolean functions $f: \{0, 1\}^2 \rightarrow \{0, 1\}$ erase a bit of information !



a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

$$a \wedge b = 0 \rightarrow a = ? , b = ?$$

Can we embed any irreversible function
into a reversible function ?

YES

irreversible function: $f : \{0,1\}^m \rightarrow \{0,1\}^n$

reversible function: $\tilde{f} : \{0,1\}^{m+n} \rightarrow \{0,1\}^{m+n}$

defined such that: $\tilde{f}(x, y) = (x, [y + f(x)] \pmod{2^n})$

where x represents m bits, while y and $f(x)$ represent n bits. Since the embedding function is **bijective**, it will be **reversible**! So at the logic level it is possible, with the price of introducing more dimensions in the calculations (**ancillary** bits y).

Why reversible computing ?

This is related to the question :
how much can we increase the
density of the electronic components
on a circuit ? What happens when we
reach the atomic scale ?

Dissipation and noise immunity in computation and communication

Rolf Landauer

IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York 10598, USA

Reversible computers which carry out each step without discarding information can, in principle, dissipate arbitrarily small amounts of energy per step if the computation is carried out sufficiently slowly. This has caused a re-examination of energy requirements in communication and measurement. There also, it is only those steps that discard information which have a lower limit on energy consumption. Such steps can be avoided in the transmission of information.

There is a lower limit per logic operation for the dissipated energy in the case of the irreversible logic gates :

$$\ln(2) \cdot k \cdot T = 4.11 \cdot 10^{-21} J \quad (T = 298 K)$$

(same order of magnitude as the thermal noise)

The number of bits at output is larger than the number of bits at input : the “disappearance” of one bit is seen as a loss of information.

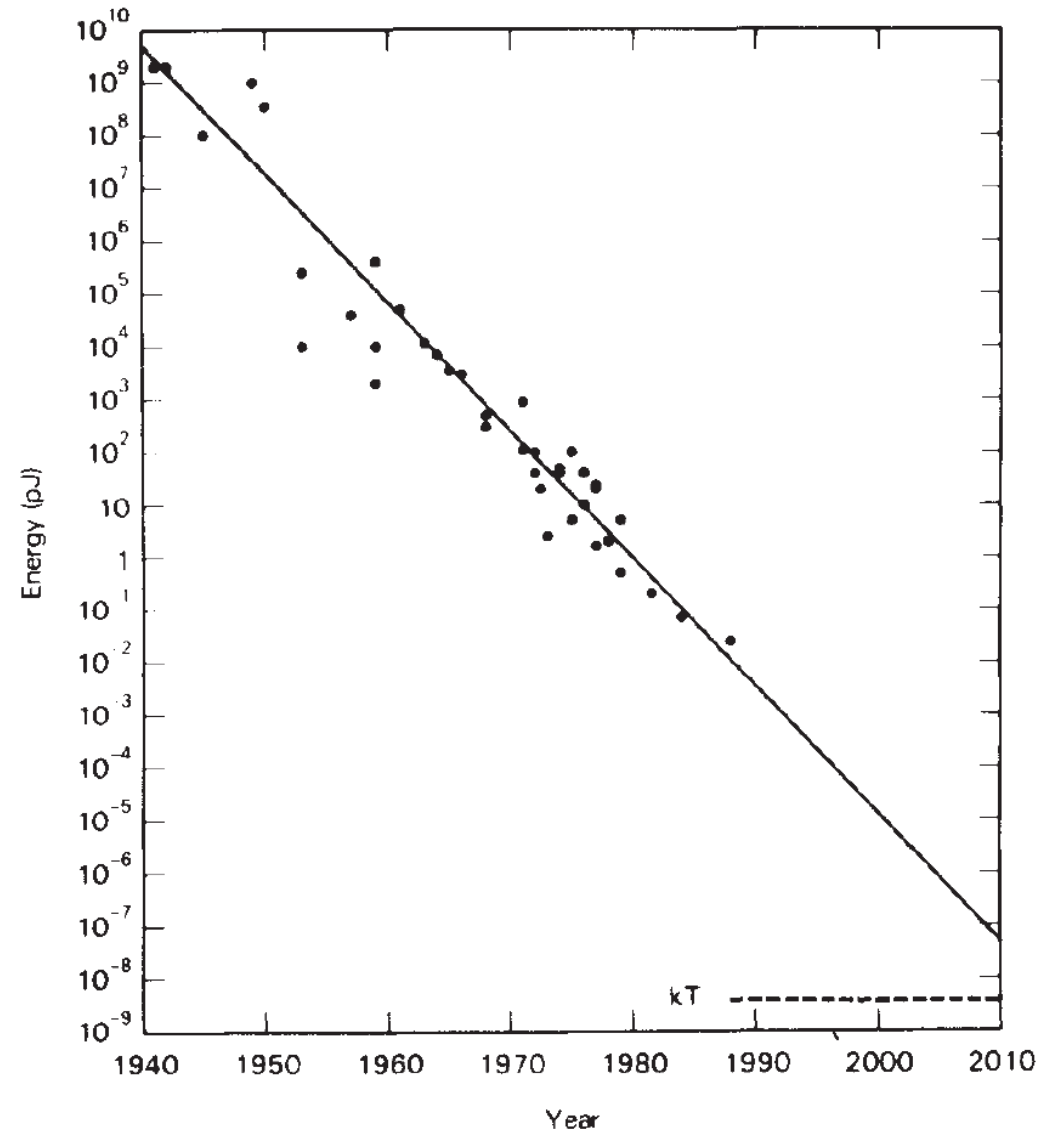


Fig. 1 The decrease in energy dissipated per logic operation over recent decades.

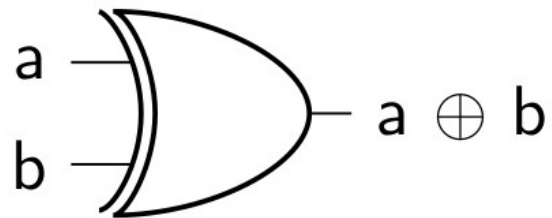
(R. W. Keyes, IBM)

Reversible logic reflects closer a number of fundamental principles of physics, like the reversibility of the fundamental laws of physics (like Newton's equations in classical mechanics).

(Feynman, Fredkin, Toffoli, ...)

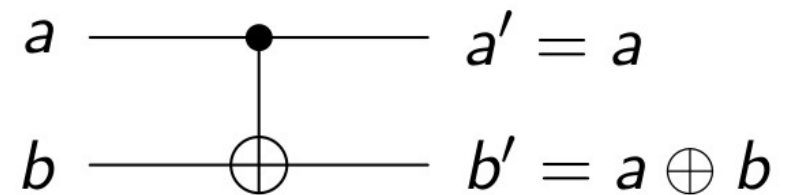
A simple reversible classical gate: the controlled-NOT (CNOT)

The exclusive-OR function (XOR):



a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

The CNOT gate:



a	b	a'	b'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

The classical CNOT gate

the control bit:

$$a \text{ --- } \bullet \text{ --- } a' = a$$

the target bit:

$$b \text{ --- } \oplus \text{ --- } b' = a \oplus b$$



two CNOT gates, applied
one after the other:

$$(a, b) \rightarrow (a, a \oplus b) \rightarrow (a, a \oplus (a \oplus b)) = (a, b)$$

so CNOT is self-inverse:

$$(CNOT)^2 = I \quad , \quad CNOT^{-1} = CNOT$$

If the target bit is set to 0 ($b=0$) then CNOT becomes the FANOUT gate:

$$(a, 0) \rightarrow (a, a)$$

... but two-bit reversible gates are not enough for universal computation !
We can not construct the NAND gate ...

The CNOT classical gate with Quantum++

(see documentation here: qpp-doc/hhtml/index.html)

```
Dynamic_bitset bits{n}; // create n classical bits
                        // in the initial state 000
                        // bit zero is the least significant bit

bits.reset(i);         // set to false bit "i"
bits.reset();          // set to false all bits

bits.set(i, bool value = true); // set bit "i" to "value"
bits.set();            // set all bits to true

bits.rand(i, double p = 0.5); // randomize bit "i"
bits.rand(double p = 0.5);    // randomize all bits
                                // according to a Bernoulli(p)
                                // distribution (coin toss)

Bit_circuit(n);        // construct a circuit with n bits
Bit_circuit(bits);    // construct and initialize
```

The CNOT classical gate with Quantum++

```
bc.CNOT(i, j);           // apply a CNOT gate having bit "i" as control
                          // and bit "j" as target

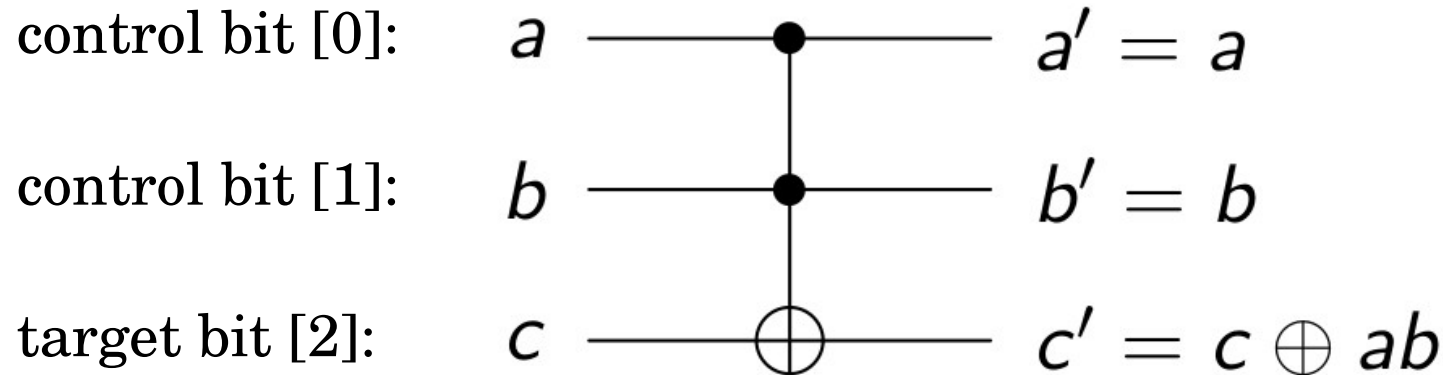
bc.get(i);               // get value of bit "i"

cmat U = gt.CNOT;       // matrix form of CNOT
                          // gt is the singleton instance of qpp::Gates
                          // qpp::cmat is Eigen::MatrixXcd (complex/double)

disp(U);                 // print the matrix
```

Exercise 1: write the circuit of the classical (two-bit) CNOT gate and test it on few values. Apply CNOT two times and check if it is equal to the identity.

Three-bit reversible gates: the Toffoli gate (controlled-controlled-NOT, or C²NOT)



```
Bit_circuit bc{3}; // initialize a circuit with 3 bits  
bc.TOF(0, 1, 2); // apply a Toffoli gate
```

Exercise 2: build the NAND function from C²NOT
(hint: set $c = 1$ and verify that $c' = a \text{ NAND } b$)

Quantum bits (qubits)

A **qubit** is a quantum object: a microscopic system whose state and evolution are governed by the laws of quantum mechanics. In order to keep a good resemblance with the classical bit, this system will be chosen to have only two possible states, corresponding to some (measurable) physical property.

The two states are *orthogonal* and any arbitrary state the system can be described as a linear combination (superposition) of those two states:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad |\alpha|^2 + |\beta|^2 = 1 \quad \alpha, \beta \in \mathbb{C}$$

(in the formalism of the quantum mechanics, the notations above are called *ket vectors*)

We will work in a two dimensional Hilbert space (an abstract vector space provided with an inner product, which is also complete).

Vector algebra with qubits

Since we describe our space with two coordinates, we can write the two basis vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and their superposition in the *state vector*: $|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$

normalized
orthogonal
vectors:

$$\left\{ \begin{array}{l} \langle 0|0\rangle = (1 \ 0) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1 \quad , \quad \langle 0|1\rangle = (1 \ 0) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0 \\ \langle 1|0\rangle = (0 \ 1) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \quad , \quad \langle 1|1\rangle = (0 \ 1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1 \end{array} \right.$$

The 1st postulate of quantum mechanics:

the state vector (or wave function) completely describes the state of the physical system.

The evolution in time of the state vector
is governed by the Schrödinger equation:
(H is the Hamiltonian, a self-adjoint operator)

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle$$

Actually, α and β are functions of time:

$$|\psi(t)\rangle = \alpha(t) |0\rangle + \beta(t) |1\rangle$$

Measuring the useful property of the qubit

The 2nd postulate of the quantum mechanics:

We associate with any observable a *self-adjoint operator* on the Hilbert space of the states. The only possible outcome of a measurement is one of the eigen-values of the corresponding operator.

A single-qubit operator can be represented by a 2x2 matrix: $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
(described within a given orthonormal vector base)

$$\sigma_z |0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = +1 |0\rangle$$

$$\sigma_z |1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -1 |1\rangle$$

$|0\rangle$ and $|1\rangle$ are eigen-vectors of the operator σ_z with eigen-values “+1” and “-1”

Measuring the useful property of the qubit

The 2nd postulate of the quantum mechanics:

If we expand the state vector over the *orthonormal* basis formed by the *eigen-vectors* of the operator:

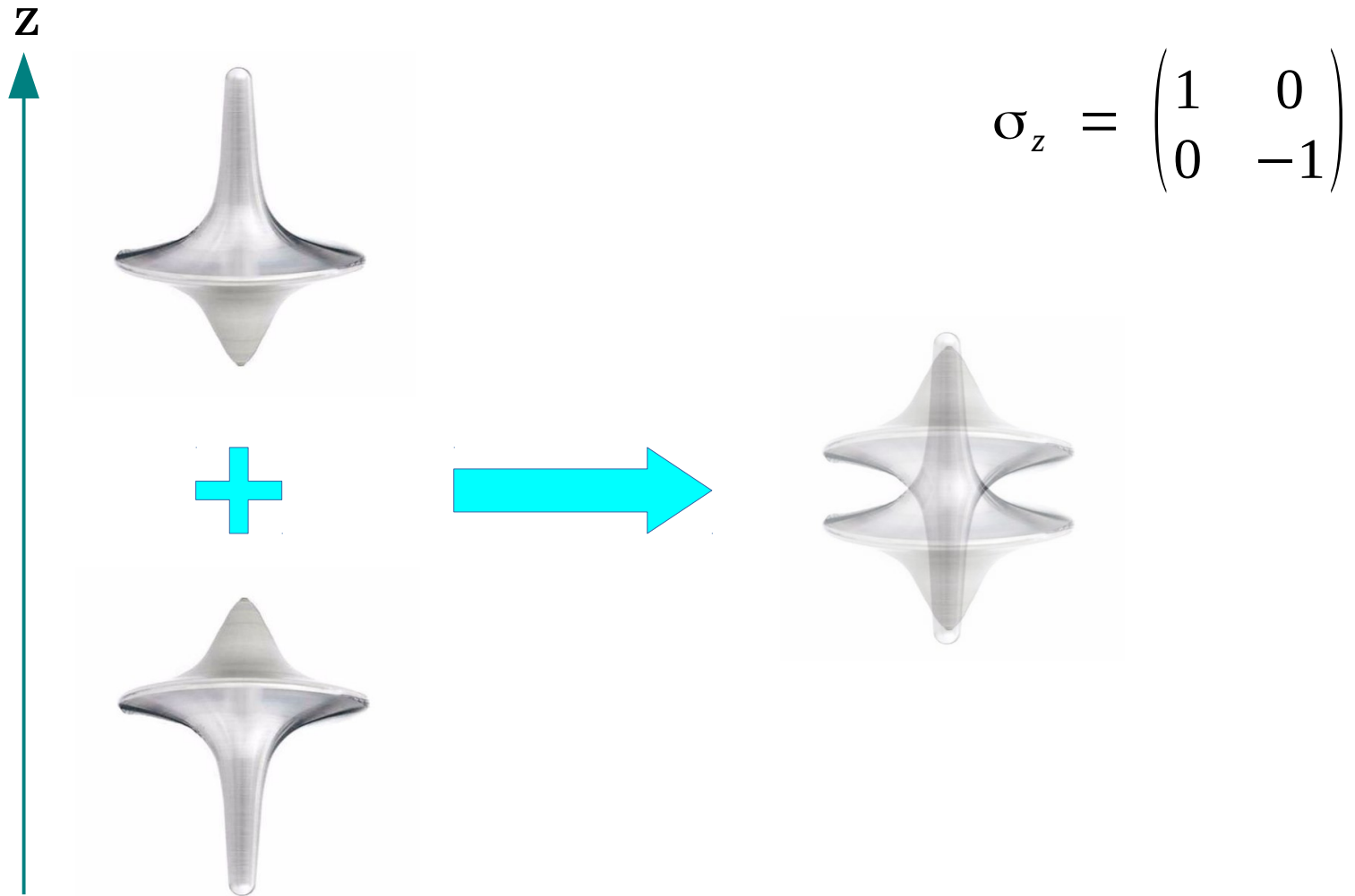
$$|\psi(t)\rangle = \alpha(t) |0\rangle + \beta(t) |1\rangle$$

then the probability that a measurement at time t results in outcome “+1” or “-1” is given by:

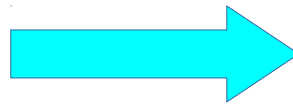
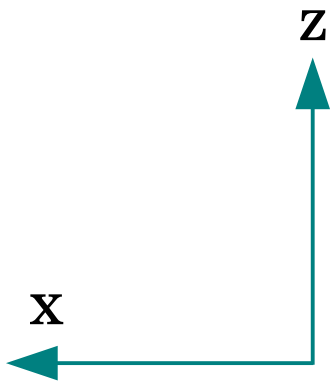
$$p_{+1}(t) = |\langle 0|\psi(t)\rangle|^2 = |\alpha(t)|^2$$

$$p_{-1}(t) = |\langle 1|\psi(t)\rangle|^2 = |\beta(t)|^2$$

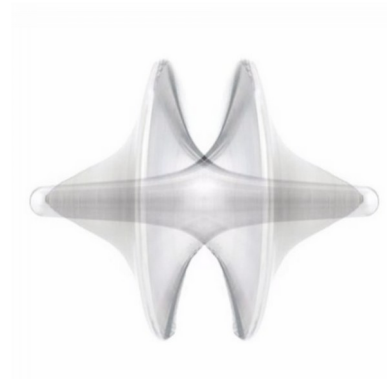
The quantified spin and the choice of the direction of the measurement



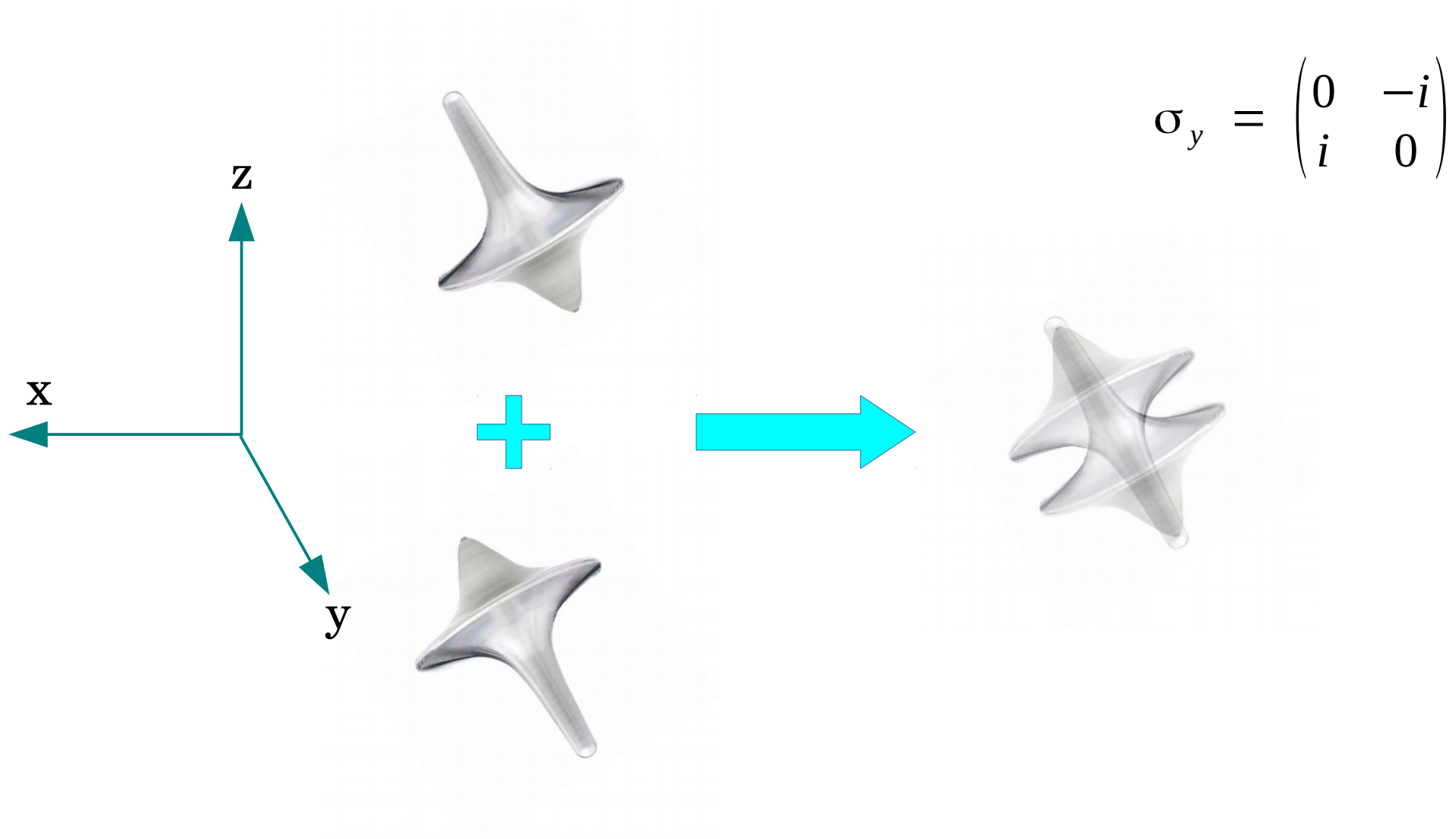
The quantified spin and the choice of the direction of the measurement



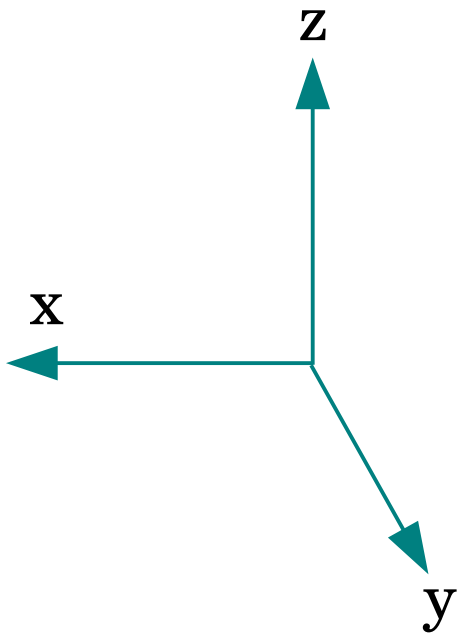
$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



The quantified spin and the choice of the direction of the measurement



The quantified spin and the choice of the direction of the measurement



σ_x , σ_y , σ_z Pauli matrices (operators), also σ_1 , σ_2 , σ_3

The eigen-vectors of the spin operators (Pauli)
corresponding to eigen-values “+1” and “-1”

$$\sigma_x : \quad |+\rangle_x = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad , \quad |-\rangle_x = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\sigma_y : \quad |+\rangle_y = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \quad , \quad |-\rangle_y = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

$$\sigma_z : \quad |+\rangle_z = |0\rangle \quad , \quad |-\rangle_z = |1\rangle$$

Example of a measurement

Let's suppose that a spin is in the eigen-state $|0\rangle$ of the σ_z operator.

This state can be expanded within the orthonormal vector basis given by the two eigen-vectors of the σ_x operator like this:

$$|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle_x + |-\rangle_x)$$

and the probabilities to measure “+1” and “-1” along the “x” axis will be:

$$p_{+1,x} = |\langle +|_x |0\rangle|^2 = \frac{1}{2}$$

$$p_{-1,x} = |\langle -|_x |0\rangle|^2 = \frac{1}{2}$$

This is a reference to the Stern-Gerlach experiment.

The 3rd postulate of quantum mechanics

If a system is described by the wave vector $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and we measure σ_z obtaining the outcome s_n ($s_1 = +1$ and $s_2 = -1$), then immediately after the measurement the state of the system is given by:

$$\frac{P_n|\psi\rangle}{\sqrt{\langle\psi|P_n|\psi\rangle}}, \quad n = \{1, 2\} \quad ; \quad P_1 = |0\rangle\langle 0| \quad , \quad P_2 = |1\rangle\langle 1|$$

P_1 and P_2 are called *projection operators* and we have $P_1 + P_2 = I$.

The probability of obtaining the outcome s_n is given by: $p_n = \langle\psi|P_n|\psi\rangle$

and the average value of the observable σ_z is given by:

$$\langle\sigma_z\rangle = \sum_n s_n p_n = \sum_n s_n \langle\psi|P_n|\psi\rangle = \langle\psi|(\underbrace{\sum_n s_n P_n}_n)|\psi\rangle = \langle\psi|\sigma_z|\psi\rangle$$

the diagonal representation of an operator

Single qubits with Quantum++

```
 $|0\rangle$  ket sigz0 = st.z0; // st = singleton instance of qpp::States
// qpp::ket is Eigen::VectorXcd
 $|1\rangle$  ket sigz1 = st.z1;
 $|+\rangle_x$  ket sigx0 = st.x0;
 $|-\rangle_x$  ket sigx1 = st.x1;
 $|+\rangle_y$  ket sigy0 = st.y0;
 $|-\rangle_y$  ket sigy1 = st.y1;
```

Exercise 3: print the components of the 6 vectors (using the function `disp(ket v)`). Note that, for instance, `sigx0[0]` is the complex coefficient multiplying the basis vector $|0\rangle$.

Exercise 4: calculate the probabilities of the Stern-Gerlach experiment, as explained on the previous slide.

Single-qubit gates σ_x , σ_y , σ_z

$$\sigma_x |0\rangle = |1\rangle$$

$$\sigma_x |1\rangle = |0\rangle$$

$$\sigma_y |0\rangle = i |1\rangle$$

$$\sigma_y |1\rangle = -i |0\rangle$$

$$\sigma_z |0\rangle = |0\rangle$$

$$\sigma_z |1\rangle = -|1\rangle$$

Exercise 5: write a code to verify these transformations through the Pauli operators.

A measurement with Quantum++

```
cmat basisZ = hevects(gt.Z); // form a matrix with the
                             // eigen-vectors of the sigma-z gate

dyn_col_vect<double> evalsZ = hevals(gt.Z); // the eigen-values
                                           // of sigma-z
```

Build a qubit in a generic state in the σ_z basis:

```
ket sigma = alpha * st.z0 + beta * st.z1;
```

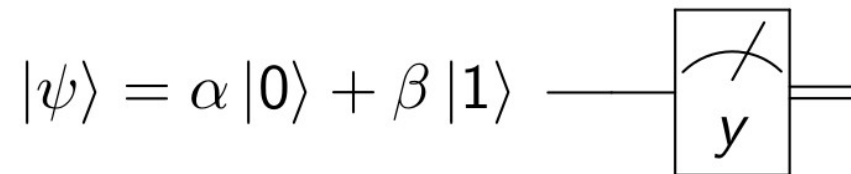
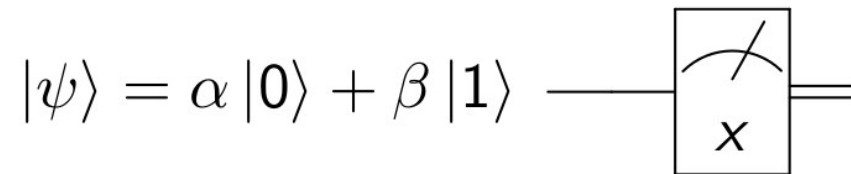
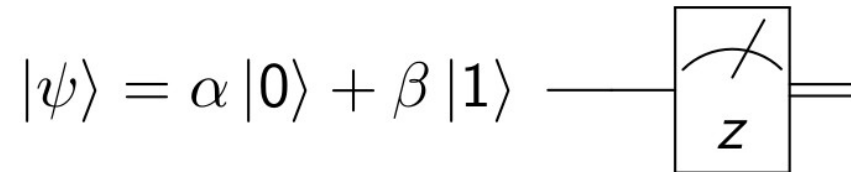
and do a measurement using the σ_z operator:

```
auto meas_sigz = measure(sigma, basisZ);
```

After one measurement, the result is given by:

```
idx res = std::get<RES>(meas_sigz);
double value = evalsZ[res];
```

Exercise 6: build a generic state and do several measurements in order to verify the second postulate of the quantum mechanics (the probability of the possible outcomes of one measurement).



Other single-qubit gates

The Hadamard gate:

$$H |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \equiv |+\rangle_x$$

$$H |1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \equiv |-\rangle_x$$

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

$$|x\rangle \text{ --- } \boxed{H} \text{ --- } (-1)^x |x\rangle + |1-x\rangle \quad , \quad |x\rangle = \{|0\rangle, |1\rangle\}$$

Transforms the computational basis: $|0\rangle, |1\rangle \rightarrow |+\rangle_x, |-\rangle_x$

The generic state of a qubit in spherical coordinates

we can write

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle = \begin{bmatrix} \cos \frac{\theta}{2} \\ e^{i\phi} \sin \frac{\theta}{2} \end{bmatrix}$$

Because:

- the two coefficients α and β are complex
- we have the total probability normalization condition
- a state vector is defined only up to a global phase of no physical significance (we can take one of the coefficients pure real)

$$p_{+1,z} = |\langle 0, \psi \rangle|^2 = \cos^2 \frac{\theta}{2} \quad , \quad p_{-1,z} = |\langle 1, \psi \rangle|^2 = \sin^2 \frac{\theta}{2}$$

Other single-qubit gates

The phase-shift gate: $R_z(\delta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix}$

$$R_z(\delta) |\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix} \begin{bmatrix} \cos \frac{\theta}{2} \\ e^{i\phi} \sin \frac{\theta}{2} \end{bmatrix} = \begin{bmatrix} \cos \frac{\theta}{2} \\ e^{i(\phi+\delta)} \sin \frac{\theta}{2} \end{bmatrix}$$

$|x\rangle$ — $\boxed{R_z(\delta)}$ — $e^{ix\delta} |x\rangle$, $|x\rangle = \{|0\rangle, |1\rangle\}$

Unitary transformations

An operator U is said to be unitary if: $UU^\dagger = U^\dagger U = I$

The *adjoint operator* is obtained by transposition followed by complex conjugate of all the matrix elements of the operator matrix.

Unitary operators act on vector in Hilbert space in a way analogous to rotations in Euclidean space: they preserve both the length of a vector and the angle between two vectors.

In quantum mechanics unitary operators are very important, because if the Hamiltonian from the Schrödinger equation is time independent, the solution can be written as:

$$|\psi(t)\rangle = U(t, t_0) |\psi(t_0)\rangle \quad , \quad U(t, t_0) = \exp \left[-\frac{i}{\hbar} H(t - t_0) \right]$$

The time independence of the Hamiltonian

The Hamiltonian operator corresponds to the sum of the kinetic and potential energies of all the particles in the system, i.e. the total energy.

If the Hamiltonian operator does not depend on time, it means that the total energy of the system remains constant in time, it is conserved (which happens when the system is isolated from the outer world).

This conclusion belongs to the formalism of both classical and quantum mechanics.

Universality of Hadamard and phase-shift gates

Any unitary operation on a single qubit can be constructed using only Hadamard and phase-shift gates. In particular, the generic state can be reached starting from $|0\rangle$ as follows:

$$e^{i\frac{\theta}{2}} |\psi\rangle = e^{i\frac{\theta}{2}} (\cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle) = R_z(\frac{\pi}{2} + \phi) H R_z(2\theta) H |0\rangle$$

Exercise 7: obtain $|+\rangle_y$ from $|0\rangle$ using the above method.

$$|+\rangle_y = \frac{1}{\sqrt{2}} (|0\rangle + i|1\rangle)$$

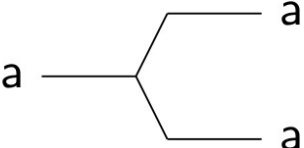
An important remark

Because the generic state $|\psi(t)\rangle = \alpha(t)|0\rangle + \beta(t)|1\rangle$ can take an infinity of values, (α and β are complex numbers) we will have an infinity of possible single-qubit quantum gates. For the classical bit we can have only the identity, the NOT and the COPY gates.

This has a consequence in the presence of the **errors**, since all the real numbers involved in the construction of the unitary operators are internally represented with a limited precision.

The no-cloning theorem

Contrary to the classical case, it is not possible to clone (COPY or FANOUT) a **generic** quantum state.

The equivalent of this:  does not exist in the quantum case.

It is impossible to build a machine that operates unitary transformations and is able to clone the generic state of a qubit (see Appendix B for a proof).

This has important consequences and leads to interesting consequences like quantum cryptography.

The possibility of cloning would also invalidate the *uncertainty relation of Heisenberg*: we would be able to simultaneously measure with infinite precision two physical properties of the system.

A generic two-qubit state

$$|\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$$

$$|ij\rangle \equiv |i\rangle |j\rangle \equiv |i\rangle \otimes |j\rangle \quad i = \{0, 1\}, j = \{0, 1\}$$

$$\alpha^2 + \beta^2 + \gamma^2 + \delta^2 = 1$$

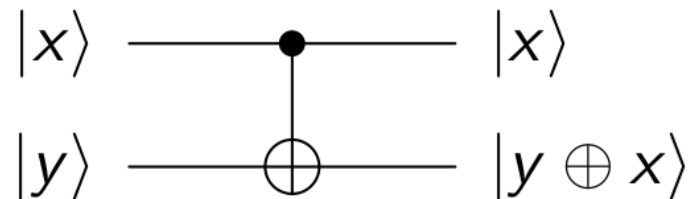
Two total vector space of the two qubits is the result of a tensor product, the computational base of the resulting space is given by the 4 possible combinations by tensor product of the computational basis of each of the two qubits.

The quantum (two-qubit) CNOT gate

It acts on the computational basis of the system of two qubits like this:

$$|00\rangle \rightarrow |00\rangle, \quad |01\rangle \rightarrow |01\rangle, \quad |10\rangle \rightarrow |11\rangle, \quad |11\rangle \rightarrow |10\rangle$$

The circuit diagram:

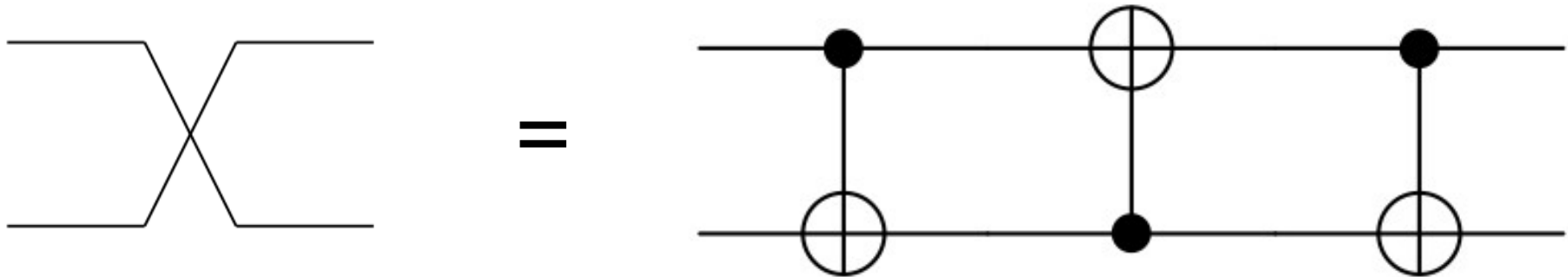


The 4×4 unitary matrix:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The state of target qubit (y) flips only if the control qubit (x) is in the $|1\rangle$ state.

A circuit for the SWAP gate using CNOT gates



(see proof in Appendix C)

Entanglement of two qubits

Contrary to the single-qubit gates, the CNOT gate can generate entangled (non-separable) states:

$$CNOT(\alpha |0\rangle + \beta |1\rangle) \otimes |0\rangle = \alpha |0\rangle \otimes |0\rangle + \beta |1\rangle \otimes |1\rangle$$

Up to now, the largest objects which can be obtained in entangled states are small crystals.

Exercise 8: built a system with two qubits and apply the CNOT gate (`gt.CNOT`) to different combined states. Compose CNOT from the flip gate `gt.X` (the Pauli x-operator) applied in controlled mode:

```
ket result = applyCTRL(psi, gt.X, {0}, {1});
```

See in Appendix D the Bell basis made of four entangled states.

A word about storing data on qubits

$$|0\rangle \text{ --- } \boxed{H} \text{ --- } \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|0\rangle \text{ --- } \boxed{H} \text{ --- } \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|0\rangle \text{ --- } \boxed{H} \text{ --- } \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

a network of 3 qubits: the application of the 3 Hadamard gates is synchronized and in the total product state we have a superposition of the values from 0 to 7.

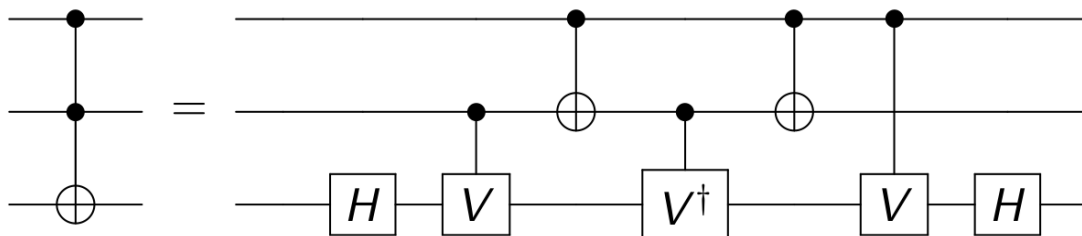
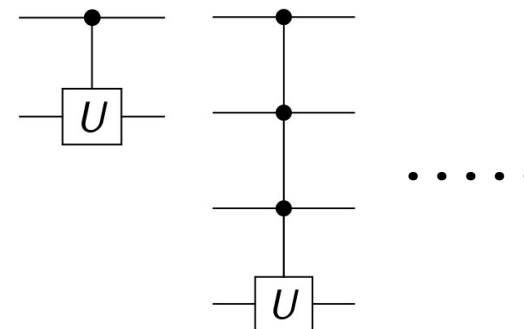
$$= \frac{1}{2^{3/2}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$$

$$= \frac{1}{2^{3/2}}(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle)$$

Universal quantum gates

Any unitary operation in the Hilbert space of n qubits, $U^{(n)}$ can be decomposed into one-qubit and two-qubit CNOT gates.

- we need few more special gates, like the controlled-U gate, where the U operator is applied to the target qubit only if the control qubit is in the $|1\rangle$ state.
- the controlled-U gate can be generalized to the C^k -U gate, with k control qubits.
- a particular C^k -U is the C^2 -NOT gate, or Toffoli gate; this a circuit implementing the Toffoli gate, using CNOT, Hadamard and the unitary operator V :



where $V = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

Universal quantum gates

and finally we come to the conclusion:

- 1) a generic operator $U^{(n)}$ can be decomposed by means of C^k -U gates
- 2) any C^k -U gate ($k > 2$) can be decomposed into Toffoli and controlled-U gates
- 3) the C^2 -NOT gate (Toffoli) can be implemented using CNOT, controlled-U and Hadamard gates
- 4) for any single-qubit rotation U, the controlled-U operation can be decomposed into single-qubit and CNOT gates

Unitary errors

Any quantum computation is given by a sequence of quantum gates applied to some initial state:

$$|\psi_n\rangle = \prod_{i=1}^n U_i |\psi_0\rangle$$

If the errors are unitary (no coupling to the environment, but any realistic implementation of a unitary operation will involve some error, since unitary operators form a continuous set), instead of operators U_i we apply slightly different operators V_i :

$$|\psi_i\rangle = U_i |\psi_{i-1}\rangle \quad V |\psi_{i-1}\rangle = |\psi_i\rangle + |E_i\rangle$$

Therefore, after n iterations we obtain:

$$|\widetilde{\psi}_n\rangle = |\psi_n\rangle + |E_n\rangle + V_n V_{n-1} |E_{n-2}\rangle + \cdots + V_n V_{n-1} \cdots V_2 |E_1\rangle$$

$$\left\| |\widetilde{\psi}_n\rangle - |\psi_n\rangle \right\| < n\epsilon$$

Unitary errors accumulate at worst linearly with the length of the quantum computation n , while stochastic errors are randomly directed and give a more favorable growth \sqrt{n} .

Quantum information: teleportation

Alice owns a two level system in some unknown state: $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and wishes to send this qubit to Bob using only a classical communication channel (we know that Alice can not clone that state into a quantum copy).

Alice can not simply measure the state, because it will immediately destroy that state with the price of obtaining only one bit of information (describing the generic state requires an infinite amount of classical information).

Quantum teleportation is possible, providing that Alice and Bob share an *entangled pair of qubits*.

For instance, starting from the computation base we can create

the entangled state using: $CNOT(H \otimes I) |01\rangle = |\psi^+\rangle \longrightarrow |\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$

Quantum information: teleportation

The three qubit state is given by the tensor product:

$$|\psi\rangle \otimes |\psi^+\rangle = \frac{\alpha}{\sqrt{2}}(|001\rangle + |010\rangle) + \frac{\beta}{\sqrt{2}}(|101\rangle + |110\rangle)$$

Alice will let her qubit interact with her half of the Bell pair, which means that she will perform a measurement not in the computational basis but in the Bell basis. The three-qubit state can be written in the Bell basis after some transformations:

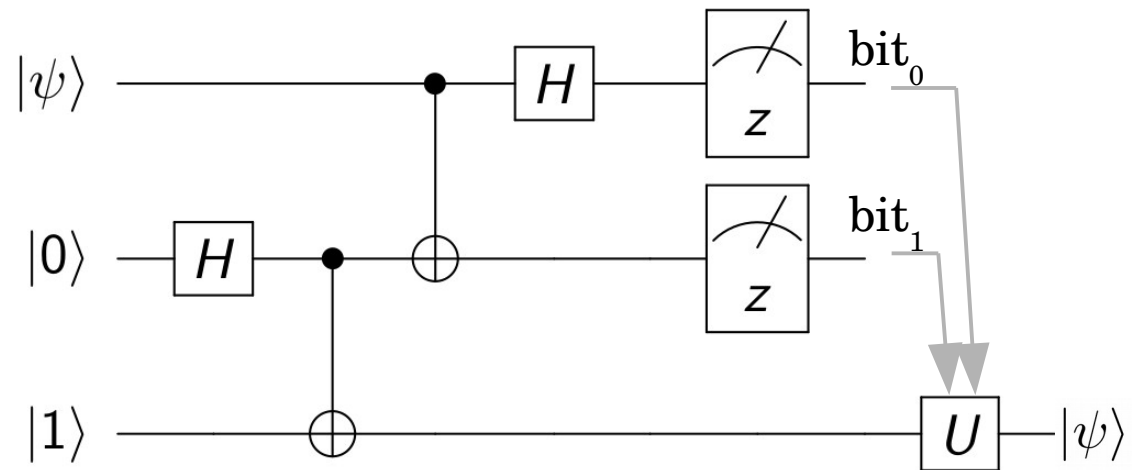
$$\begin{aligned} |\psi\rangle \otimes |\psi^+\rangle = & \frac{1}{2} |\psi^+\rangle (\alpha |0\rangle + \beta |1\rangle) + \frac{1}{2} |\psi^-\rangle (\alpha |0\rangle - \beta |1\rangle) \\ & + \frac{1}{2} |\phi^+\rangle (\alpha |1\rangle + \beta |0\rangle) + \frac{1}{2} |\phi^-\rangle (\alpha |1\rangle - \beta |0\rangle) \end{aligned}$$

and after the application of the two last gates $(H \otimes I)CNOT$ we obtain:

$$\begin{aligned} |\psi\rangle \otimes |\psi^+\rangle = & \frac{1}{2} |01\rangle (\alpha |0\rangle + \beta |1\rangle) + \frac{1}{2} |11\rangle (\alpha |0\rangle - \beta |1\rangle) \\ & + \frac{1}{2} |00\rangle (\alpha |1\rangle + \beta |0\rangle) + \frac{1}{2} |10\rangle (\alpha |1\rangle - \beta |0\rangle) \end{aligned}$$

Quantum information: teleportation

this is the circuit for teleportation:



Finally, Alice makes a measurement on his two qubits and sends the result to Bob, in the form of two classical bits (0, 1) which correspond to the computational basis. If Bob chooses to apply a unitary operator U to his qubit according to the pair of bits sent by Alice as in next table, he will obtain exactly the initial generic state which Alice wanted to transmit:

Alice measures	Bob gets the bits	and applies to his qubit
$ 01\rangle$	0,1	I
$ 11\rangle$	1,1	σ_z
$ 00\rangle$	0,0	σ_x
$ 10\rangle$	1,0	$i\sigma_y$

Quantum information: teleportation

Remark: before the $|\psi\rangle$ state “materializes” at Bob's side, the measurement at Alice's side destroys its “original” (the no-cloning theorem).

Quantum information: teleportation

```
ket psi_a = randket(2);           // a random generic state
ket phi_AB = st.b00;             // the psi+ Bell state
ket input_aAB = kron(psi_a, phi_AB); // create the global input
                                   // state with a direct
                                   // (tensor) product
auto results = measure_seq(input_aAB, {0, 1}); // measure only a
                                                // part of the
                                                // multi-part state
                                                // vector
```

Exercise 9: write the code for the teleportation.

Hint: construct the last operator to be applied to Bob's qubit by using the three Pauli operators and their relations:

$$\sigma_x \sigma_y = i \sigma_z, \quad \sigma_y \sigma_z = i \sigma_x, \quad \sigma_z \sigma_x = i \sigma_y$$

Function evaluation

(see Appendix A for the classical case)

This is the basic task performed by a classical computer!

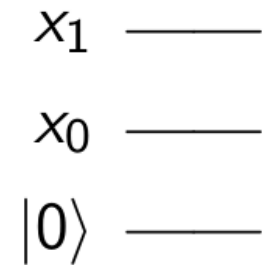
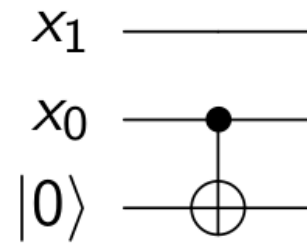
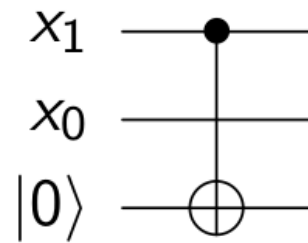
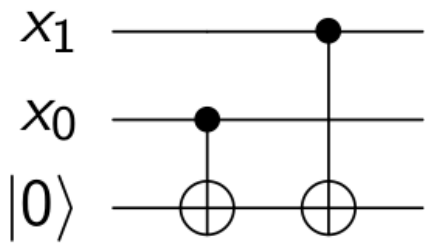
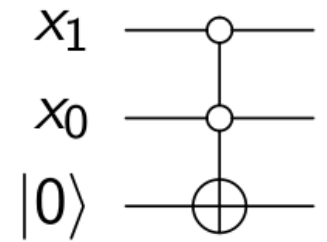
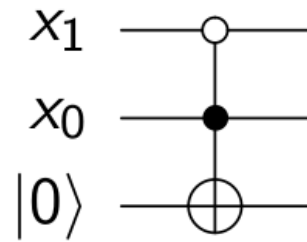
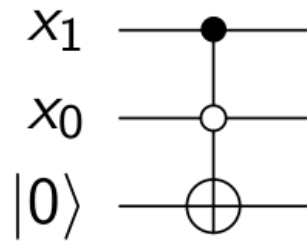
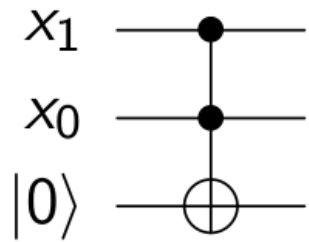
$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

If we start from the classical case, each *minterm* is implemented in a quantum computer by a generalized Cⁿ-NOT gate. In general the number of *minterms* grows **exponentially** with the number n of (qu)bits and we can not evaluate the function f efficiently (i.e. with a number of elementary gates polynomial in n).

Embedding the irreversible function f into a reversible function is equivalent with finding the appropriate unitary transformation U_f using an ancillary qubit $|y\rangle$:

$$U_f |x_{n-1}, x_{n-2}, \dots, x_0\rangle |y\rangle = |x_{n-1}, x_{n-2}, \dots, x_0\rangle |y \oplus f(x_{n-1}, x_{n-2}, \dots, x_0)\rangle$$

Function evaluation



Quantum circuits implementing the two-bit binary functions: only 8 functions are shown (from a total of 16) since $f_{15-i} = \bar{f}_i$ for the other 8 functions, by applying a NOT (σ_x) to the ancillary qubit.

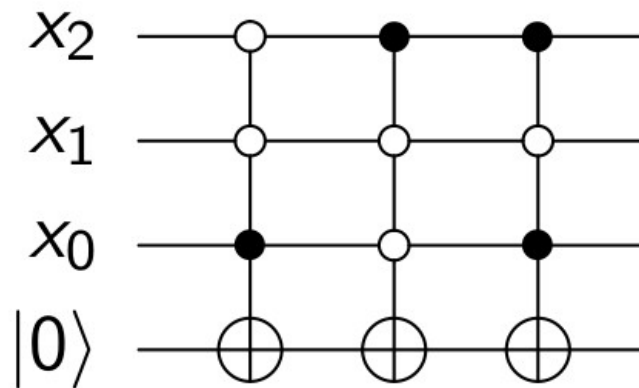
Function evaluation

Example: a binary function of three qubits which equals to 1 only for 3 combinations of values out of the 8 possible (there will be only 3 minterms to evaluate):

$$\bar{x}_2 \wedge \bar{x}_1 \wedge x_0 \quad , \quad x_2 \wedge \bar{x}_1 \wedge \bar{x}_0 \quad , \quad x_2 \wedge \bar{x}_1 \wedge x_0$$

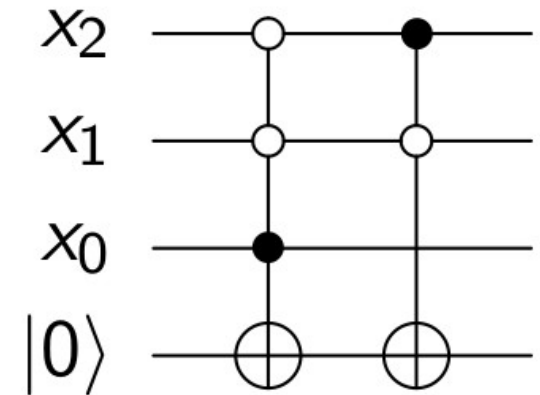
x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a) circuit with 3 C^3 -NOT gates



=

(b) simplified circuit $x_0 + \bar{x}_0 = 1$



Evaluation of x^2 for 2-qubit input

Note that: if we need $n = \log_2 N$ qubits to load
 an integer $x \in [1, N]$, then
 we need $2n = \log_2 N^2$ qubits to store
 the output $x^2 \in [1, N^2]$

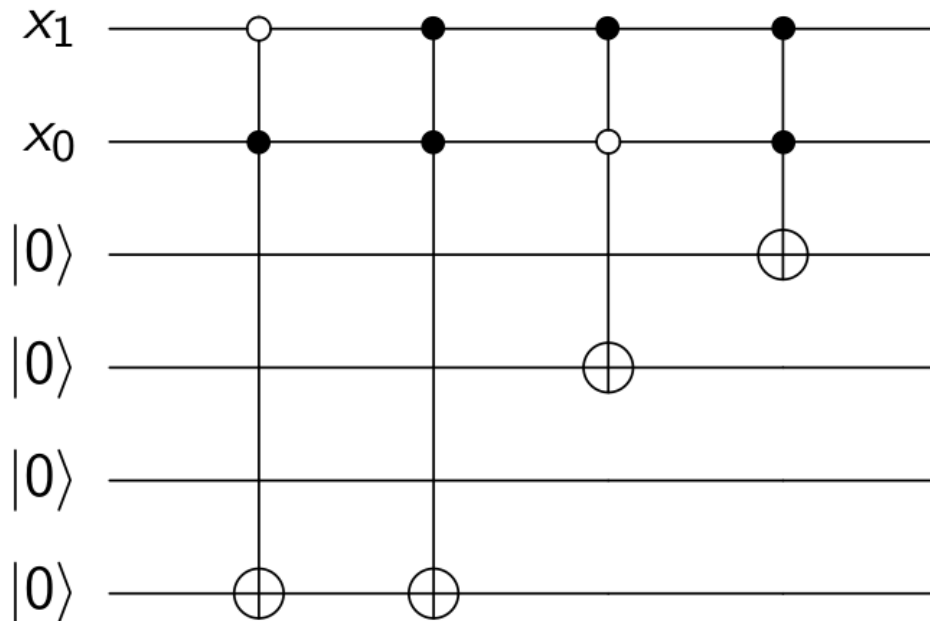
This corresponds to 4 binary functions (as in previous slides), which can be evaluated reversibly using 4 ancillary qubits.

Evaluation of x^2 for 2-qubit input

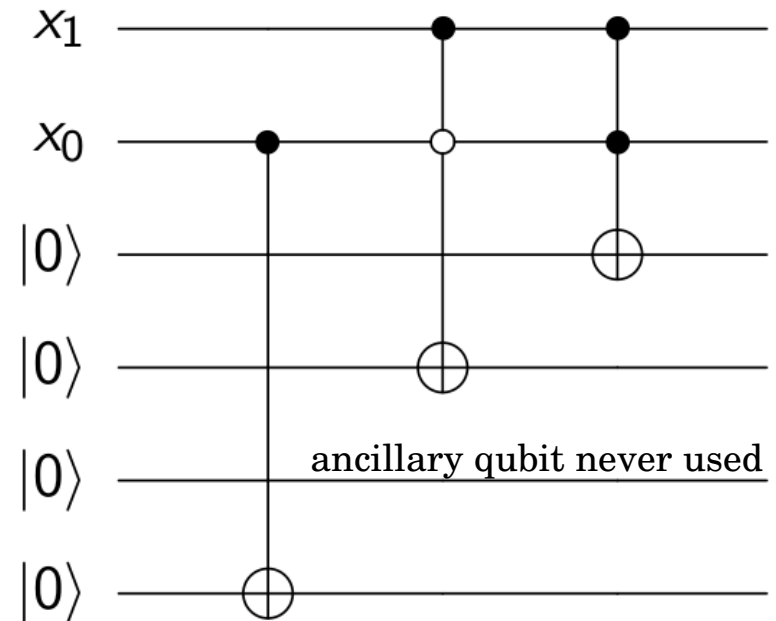
The function table:

x_1	x_0	x	x^2	x^2	x^2, g_0	x^2, g_1	x^2, g_2	x^2, g_3
0	0	0	0	0000	0	0	0	0
0	1	1	1	0001	0	0	0	1
1	0	2	4	0100	0	1	0	0
1	1	3	9	1001	1	0	0	1

(a) with the 4 minterms



(b) simplified



Real run on IBM Q processors with Qiskit

IBM Q - quantum computing for researchers, www.ibm.com/quantum-computing/

Qiskit - open-source quantum computing software development framework, qiskit.org

IBM Q account: qiskit.org/ibmqaccount

Tutorials: github.com/Qiskit/qiskit-ibmq-tutorials.git

See [3_the_ibmq_account.html](#) for information about the providers and the backends they offer (properties and configuration).

[1_getting_started_with_qiskit.html](#) is the HTML export of the jupyter notebook doing the “getting started” tutorial example.

Real run on IBM Q processors with Qiskit

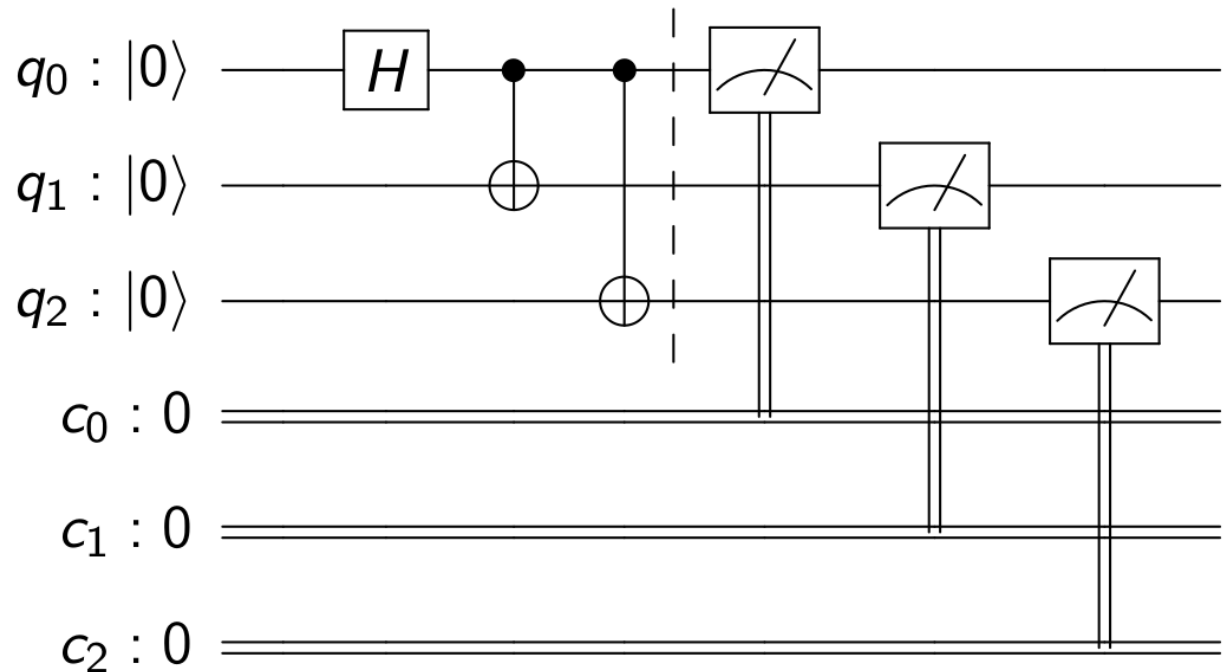
The “getting started” example from the Qiskit tutorials with IBM Q backend (ibmqx2):

- 5 qubits, 1024 shots

```
import numpy as np
from qiskit import *
```

```
circ = QuantumCircuit(3)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
```

```
meas = QuantumCircuit(3, 3)
meas.barrier(range(3))
meas.measure(range(3), range(3))
qc = circ + meas
```



creates a 3-qubit entangled state, GHZ (Greenberger-Horne-Zeilinger):

$$|GHZ\rangle = \frac{|0\rangle^{\otimes 3} + |1\rangle^{\otimes 3}}{\sqrt{2}} = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$$

Real run on IBM Q processors with Qiskit

```
from qiskit import IBMQ

IBMQ.load_account()

IBMQ.providers()

provider = IBMQ.get_provider(group='open')

provider.backends()

backend = provider.get_backend('ibmqx2')

job_exp = execute(qc, backend = backend)

job_monitor(job_exp)

result_exp = job_exp.result()

counts_exp = result_exp.get_counts(qc)

plot_histogram(...
```

Real run on IBM Q processors with Qiskit

Exercise 10: write the Quantum++ code for the “getting started” Qiskit example and check with simulations the correct results of the measurement of an GHZ state.

```
QCircuit qc{3, 3};           // create a circuit with 3 qubits and
                              // 3 classical bits (for the read-out)

qc.measureZ(0, 0);          // make a measurement (sigma-z) of
                              // qubit "0" and transfer the result
                              // to the classical bit "0"

QEngine engine(qc);         // initialize a quantum engine

engine.execute(shots, false); // execute a number of "shots"

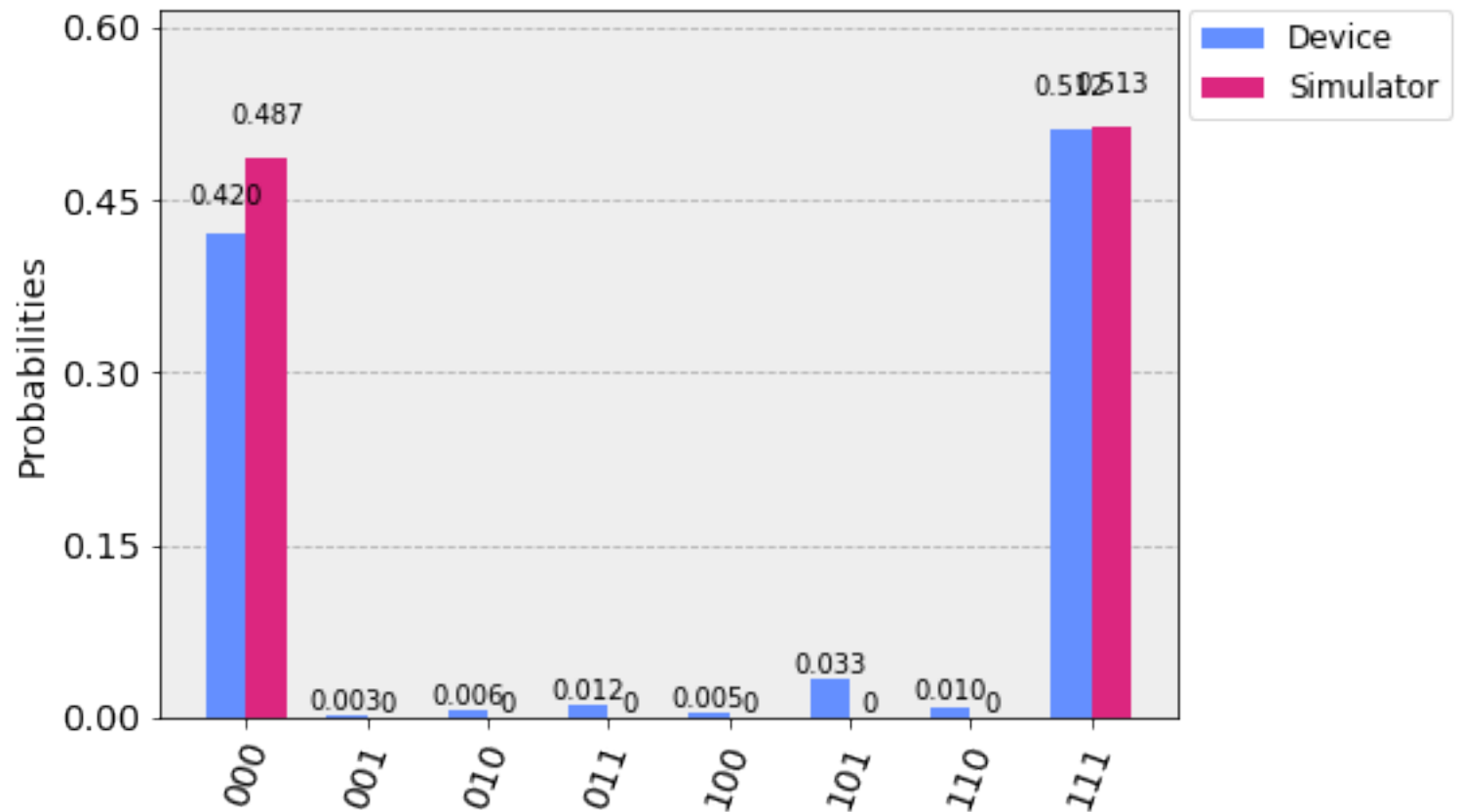
auto stats = engine.get_stats(); // get the statistics at the end
                              // of the series of shots
```

Real run on IBM Q processors with Qiskit

Qiskit provides also a QASM simulator (QASM = A Quantum Programming Language)

www.quantum-inspire.com/kbase/qasm/

We should have only states (000) and (111) but in reality we see with small probability other states too.



The GHZ state

Appendix

Appendix A

AND, OR, NOT and FANOUT constitute a universal set of gates for classical computation.

Proof.

The m -bit function is equivalent to m one-bit (or Boolean) functions

$$f_i : \{0, 1\}^n \rightarrow \{0, 1\}, \quad (i = 1, 2, \dots, m)$$

where $f = (f_1, f_2, \dots, f_m)$. For any values of the input argument $a = (a_{n-1}, a_{n-2}, \dots, a_1, a_0)$, one way to compute the boolean function $f_i(a)$ is to consider the *minterms* $f_i^{(l)}(a)$, defined as

$$f_i^{(l)} = \begin{cases} 1, & \text{if } a = a^{(l)} \\ 0, & \text{otherwise} \end{cases}$$

Appendix A

for instance, if the particular value of $a^{(l)} = 110100 \dots 001$, then $f_i^{(l)}$ can be defined as follows

$$f_i^{(l)} = a_{n-1} \wedge a_{n-2} \wedge \bar{a}_{n-3} \wedge a_{n-4} \wedge \bar{a}_{n-5} \wedge \bar{a}_{n-6} \wedge \dots \wedge \bar{a}_2 \wedge \bar{a}_1 \wedge a_0$$

the one-bit function f_i can be calculated for all possible a values as follows

$$f_i(a) = f_i^{(1)} \vee f_i^{(2)} \vee \dots \vee f_i^{(k)}$$

as the logical OR of all k minterms, with $0 \leq k \leq 2^n - 1$ (2^n is the number of all possible values of the input a). The FANOUT gate is required to feed the input a to the k minterms.

Appendix A

Consider the Boolean function $f(a)$, where $a = (a_2, a_1, a_0)$ defined as follows

a	a_2	a_1	a_0	$f(a)$
$a^{(1)} = 1$	0	0	1	1
$a^{(2)} = 3$	0	1	1	1
$a^{(3)} = 6$	1	1	0	1
$a^{(4)} = 0$	0	0	0	0
$a^{(5)} = 2$	0	1	0	0
$a^{(6)} = 4$	1	0	0	0
$a^{(7)} = 5$	1	0	1	0
$a^{(8)} = 7$	1	1	1	0

$$f^{(1)} = \bar{a}_2 \wedge \bar{a}_1 \wedge a_0$$

$$f^{(2)} = \bar{a}_2 \wedge a_1 \wedge a_0$$

$$f^{(3)} = a_2 \wedge a_1 \wedge \bar{a}_0$$

$$f(a) = f^{(1)}(a) \vee f^{(2)}(a) \vee f^{(3)}(a)$$

Note: we may have up to $2^3 = 8$ minterms.

Appendix B

The no-cloning theorem

Let us consider two qubits in the states $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ (generic) and $|\phi\rangle$ (ancillary qubit), the cloning machine in the initial state $|A_i\rangle$ and suppose there is a unitary transformation U such that:

$$U(|\psi\rangle |\phi\rangle |A_i\rangle) = |\psi\rangle |\psi\rangle |A_{f\psi}\rangle = (\alpha |0\rangle + \beta |1\rangle)(\alpha |0\rangle + \beta |1\rangle) |A_{f\psi}\rangle$$

but at the same time we can write:

$$U(|\psi\rangle |\phi\rangle |A_i\rangle) = U((\alpha |0\rangle + \beta |1\rangle) |\phi\rangle |A_i\rangle)$$

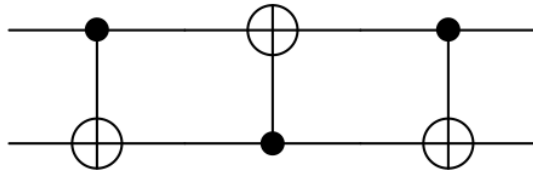
if we invoke the linearity of quantum mechanics we obtain:

$$\alpha U(|0\rangle |\phi\rangle |A_i\rangle) + \beta U(|1\rangle |\phi\rangle |A_i\rangle) = \alpha |0\rangle |0\rangle |A_{f0}\rangle + \beta |1\rangle |1\rangle |A_{f1}\rangle$$

which is the entangled state clearly different from the desired cloned state. 75

Appendix C

Obtaining the SWAP gate from CNOT gates



$1 \times 2:$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$(1 \times 2) \times 3:$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Appendix C

if we start with this:

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

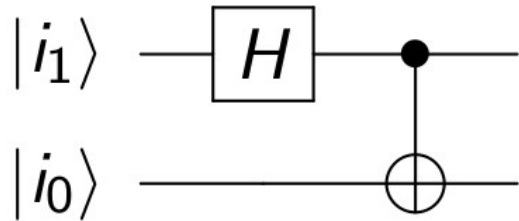
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |1\rangle \otimes |0\rangle$$

we have this at the end.

Appendix D

The Bell (EPR) basis

This circuit:



transforms the computational
basis states into the Bell states:

$$\left\{ \begin{array}{l} |00\rangle \rightarrow |\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |10\rangle \rightarrow |\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ |01\rangle \rightarrow |\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |11\rangle \rightarrow |\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{array} \right.$$

Appendix E

Exercise	qpp file
1	cls-CNOT
2	cls-NAND-Toffoli
3	qubit-1
4	
5	pauli-1
6	qubit-meas
7	unitary-H-PS
8	qnt-CNOT
9	teleportation
10	qiskit-start