ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

An Introduction to Quantum Computing

IoT-Prism Lab internal seminar

March 31, 2023

Luciano Bononi

Department of Computer Science and Engineering

email: luciano.bononi@unibo.it

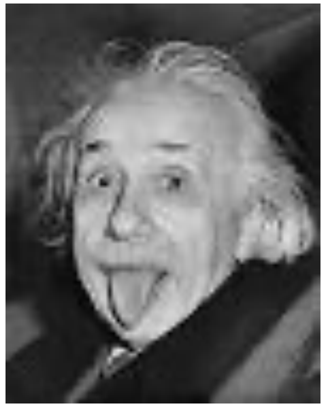
Assumption

There are only 10 types of people in the world... those who understand binary arithmetic and those who don't.

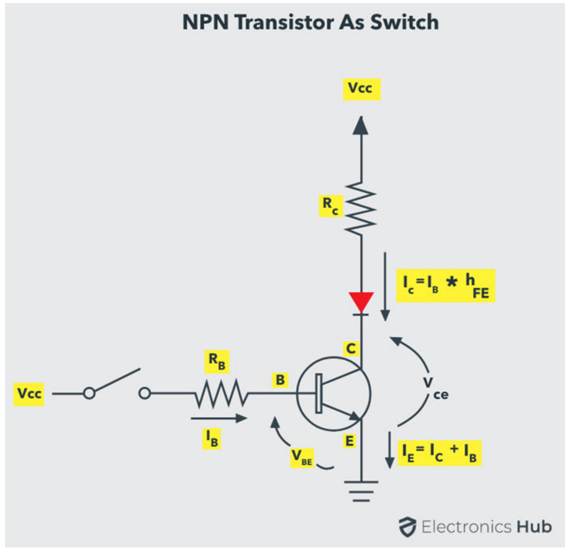
anonymous computer scientist

At the end of this illustration maybe we will reduce the entropy of the computing universe or maybe not:

There are only $\binom{0}{1} \otimes \binom{1}{0}$ groups of computer scientists in the world... split out by how they understand Quantum Computing



from Classical Computing to Quantum Computing (1)



In 1947 Walter Brattain, John Bardeen and William Shockley invented the transistor (Bell labs 1947, Nobel Prize for Physics 1956):
OPEN or CLOSE a circuit electronically by means of a control signal

Corollary: the state of a circuit at time T is either OPEN or CLOSED

0 1

Binary logic: I can represent information using only two symbols (bit values): 0 and 1

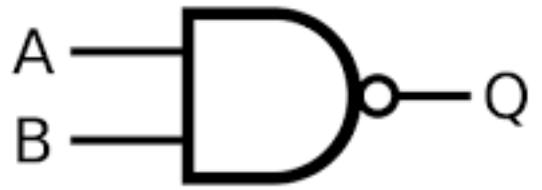
If we associate the states of the circuits (e.g. Open to the value of bit 0 and Closed to the value of bit 1) we have a binary electronic encoding. That is, the state of a machine represents a binary value.

Can we build machines that transform states into different predictable states? Certainly!

The logic gates carry out transformation functions of input signals into output signals according to a pre-defined and DETERMINISTIC truth table (that is, from the same input and operation I always get the same result produced)

from Classical Computing to Quantum Computing (2)

The NAND (Not AND) digital gate is interesting:



truth table (NAND):

A	B	Q (output)
0	0	1
0	1	1
1	0	1
1	1	0

You can demonstrate that by composing NAND gates you can realize all the basic logic ports.
Not (x) = 0 iff x=1, 1 o.w.
OR (x,y) = 1 iff at least one between x and y is 1, 0 o.w.
AND (x,y) = 1 iff both x and y is 1, 0 o.w.
XOR (x,y) = 1 iff ONLY one between x and y is 1, 0 o.w.
etc. etc.
Some logic gates are sufficient to realise all the building blocks for arbitrary complex computations

from Classical Computing to Quantum Computing (3)

If we create a compositional logic for gates and use it for the operation of a controllable computation by binary control signals (the executable program) and run the computation on the input data of our problem we obtain a classical programmable computer which will give us the expected output results for any admissible combination of input.

fact 1: we have only one state at any instant, which represents the evolution of the calculation process (intermediate result) towards the solution. Causal (and deterministic) sequential steps: the result of the next step may ALSO depend on the intermediate result obtained from the previous steps, and is unique and repeatable.

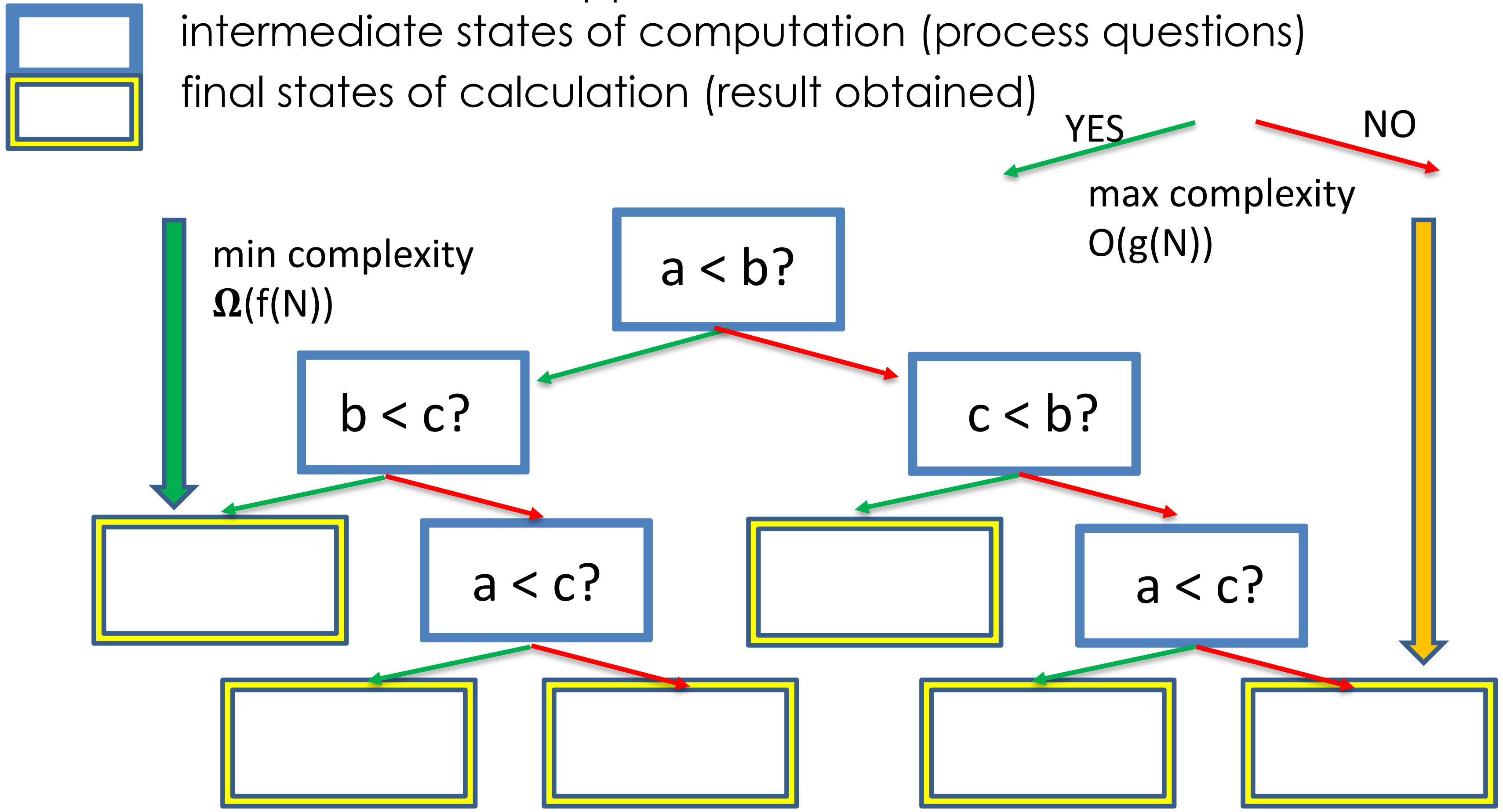
Fact 2: When will we complete the calculation process? When the calculator has performed all the necessary steps in sequence.

And if by chance we take the wrong way (we write a wrong program) to get the expected calculation? We would get a wrong result or we will never know the result...and we will never know that we will never know (the halting problem is undecidable).

Classic computational tree

example: we want to sort three integer elements a, b, c in order from smallest to largest (by comparisons), then what program should I write?

... and when I run it what happens? What if I have to order N items?


intermediate states of computation (process questions)
final states of calculation (result obtained)
YES
NO
max complexity O(g(N))
min complexity Ω(f(N))
a < b?
b < c?
c < b?
a < c?
a < c?

I carry out the whole computational path from the root of the tree to a leaf (result), choosing which path to follow for each choice, and hoping not to make a mistake so as not to have to go up and down from other paths.

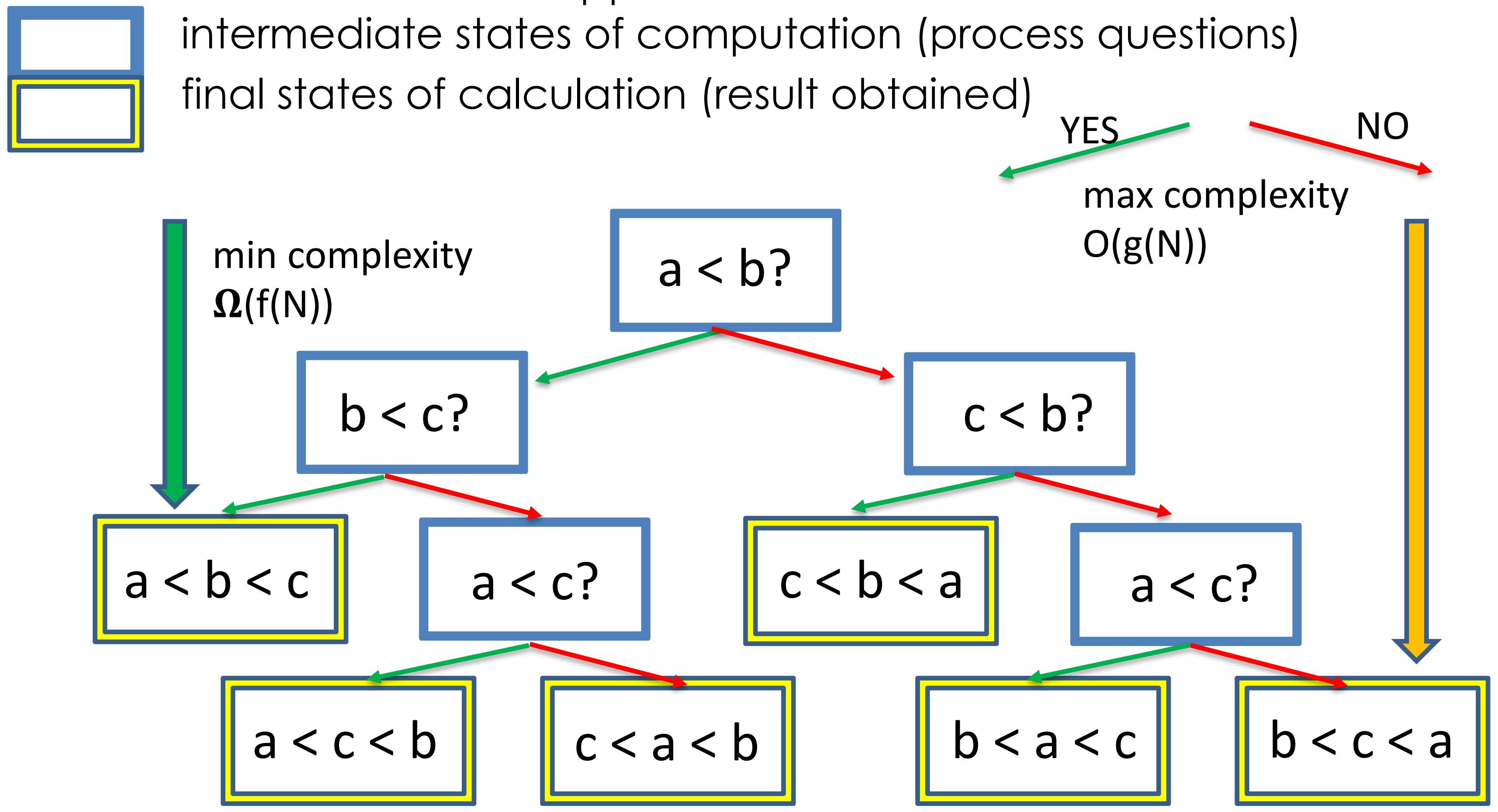
If I have to go up and down the tree (the program is inefficient or wrong) the calculation path gets longer and may never arrive at the result.

<u>I have to make correct and efficient programs to minimize the computational path</u>

Classic computational tree

example: we want to sort three integer elements a, b, c in order from smallest to largest (by comparisons), then what program should I write?

... and when I run it what happens? What if I have to order N items?



I carry out the whole computational path from the root of the tree to a leaf (result), choosing which path to follow for each choice, and hoping not to make a mistake so as not to have to go up and down from other paths.

If I have to go up and down the tree (the program is inefficient or wrong) the calculation path gets longer and may never arrive at the result.

<u>I have to make correct and efficient programs to minimize the computational path</u>

from Classical Computing to Quantum Computing (4)

So today we have classic computers and we try to program them well, but we have to wait many steps before (perhaps) having the results.

idea! Let's move fast! i.e. we use fast Clocks.

We need an even faster drummer of the "computing band" on a chip.

But to do that we need faster logic gates.

Physical limit: speed of light (c = 300.000.000 m/sec).

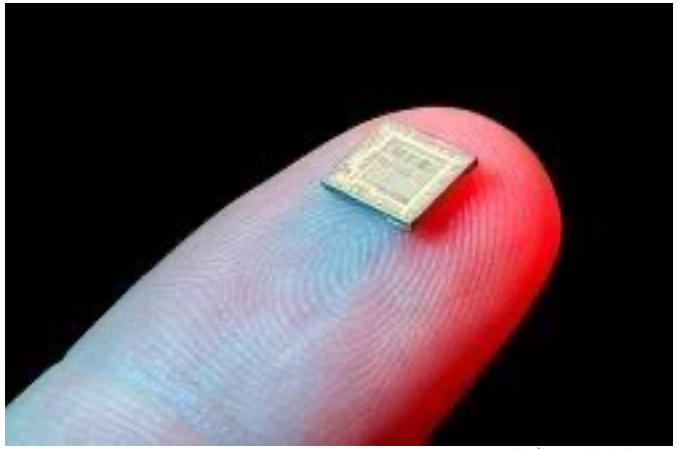
Let us assume a computation requires a sequence of one billion logic gates to be traversed on a Chip to transform signals in sequence from input to output results. Each logic gate occupies 10 nanometers. How far does the electric current travel step by step?

10 nanometers of distance per gate x 1 billion gates = 10 meters

Even by traveling at speed c, it will not take less than:

time = space / speed = 10 / 300.000.000 = 33.3 nanoseconds.

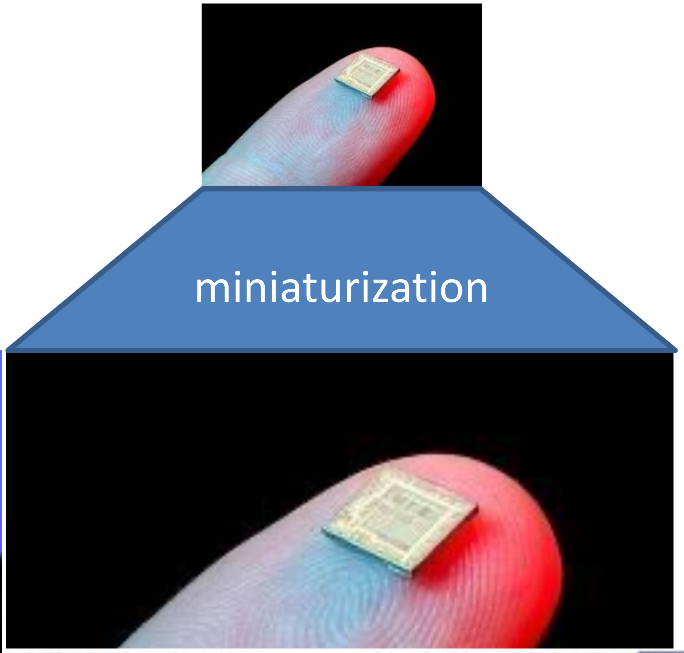
But we are even more in a hurry, so what?



from Classical Computing to Quantum Computing (5)

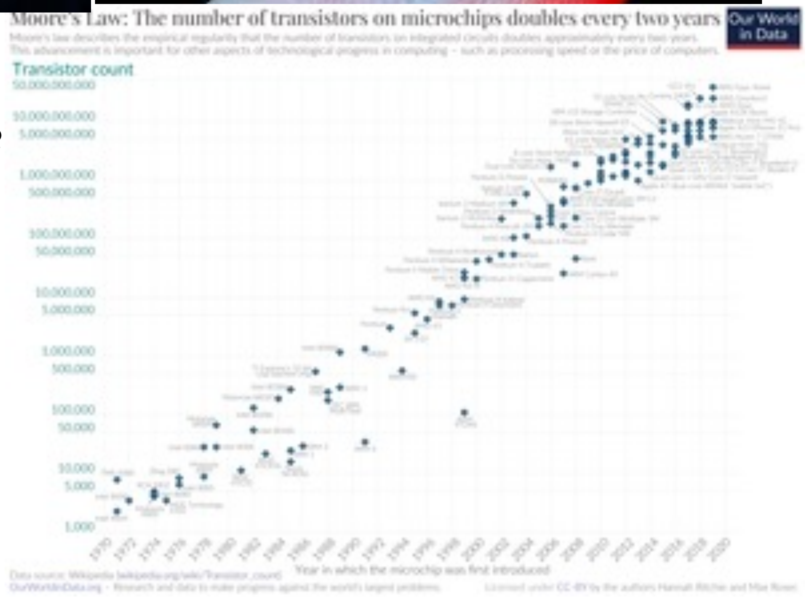
If we can't send faster-than-light signals, the only way to make faster steps along the computational path is to reduce the total distance ahead.

idea! Miniaturize further more the logic gates on the Chip.



Moore Law: number of transistors in the same chip space doubles every two years (in 2024 we will be around 3-2 nanometers)

Gordon E. Moore
1929 - 2023



Ok, but is there a physical limit that we will reach sooner or later?
Will we ever be able to make transistors on less than an atom of a chip?

So what?
Let’s be creative... parallelize calculation processes where possible, etc. (not possible/easy for many problems)

physical limits of classical computing (summary)

We will not be able to accelerate the clocks beyond a limit related to the physical size of the chips

We won't be able to miniaturize chips indefinitely

We won't always be able to invent faster parallel algorithms and solutions

Got stuck?.... but there is a new way.

Quantum Computing

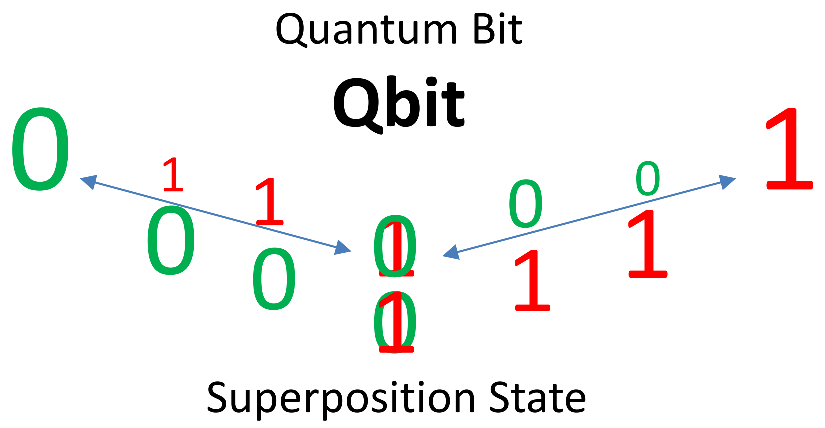
from Classical Computing to Quantum Computing (5)

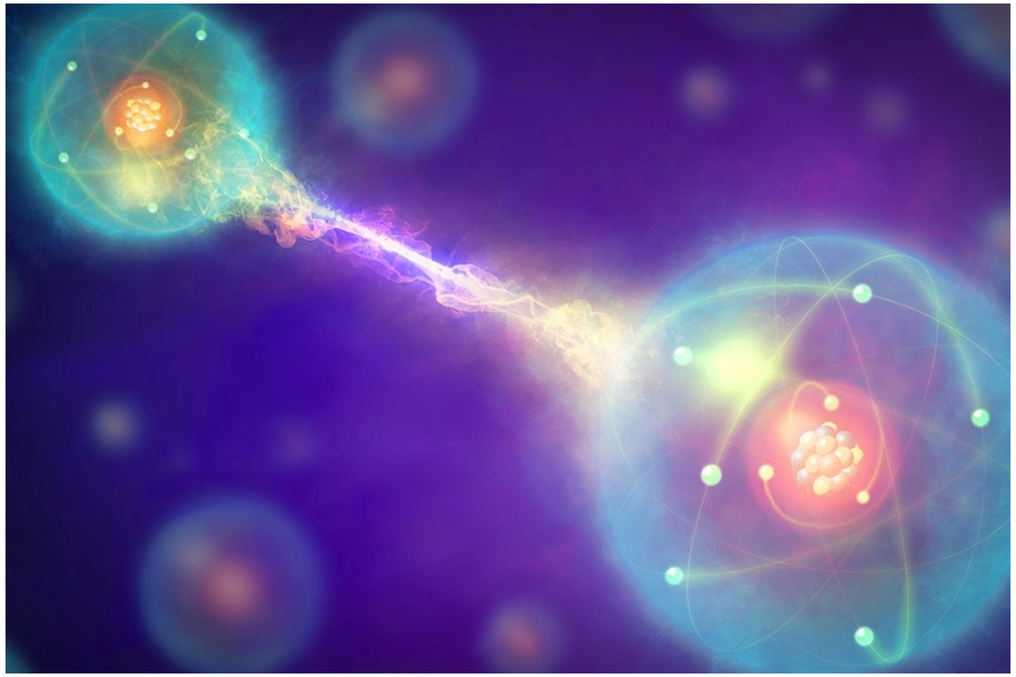
So let's take advantage of a new PHYSICS: quantum mechanics.

- dimensional order of the (subatomic) particles constituting the atom (from 10^{-15} and below). E.g. electrons, photons, etc

- A universe made of surprising physical phenomena.

- new bits (Qbit), which actually have the two states 0 and 1, but can be in both states at the same time (superposition principle).



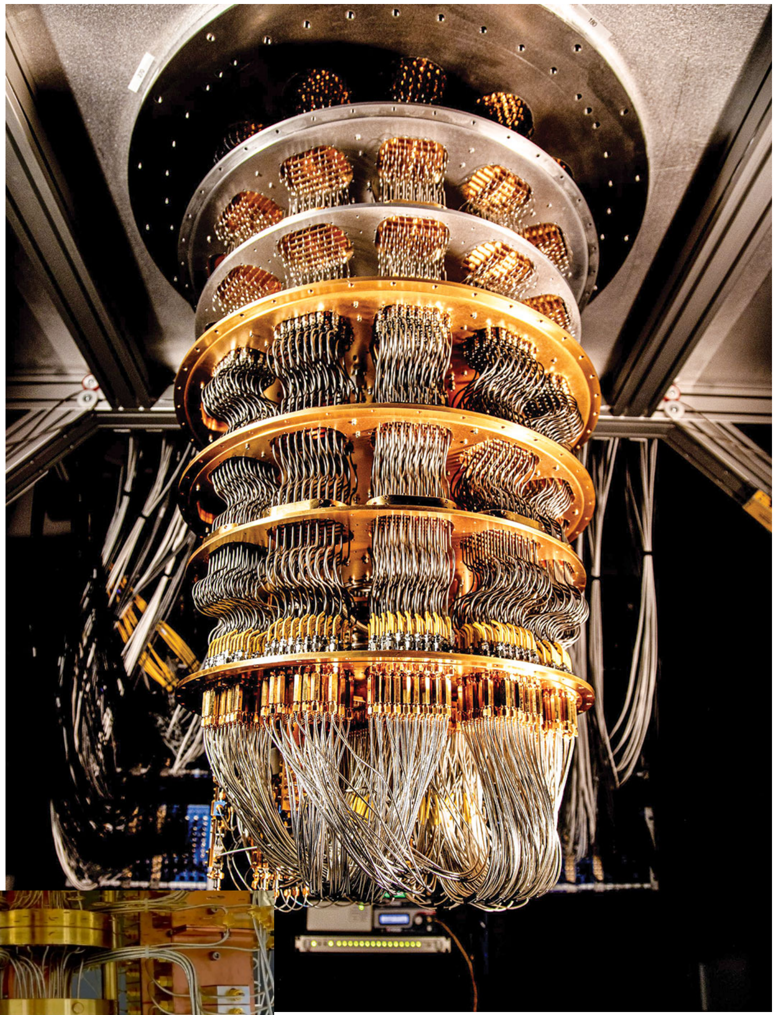

credits: wired italia

from Classical Computing to Quantum Computing (6)

1) Ok, but can we control Qbits states analogously to electric currents with a transistor?

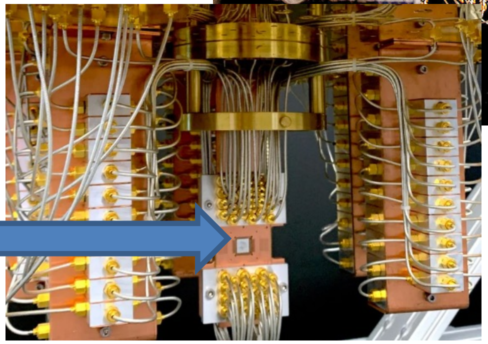
- Yes, but we have to act on superfluid particles close to absolute zero (that are stable and without «noise»).

Quantum Computer: first of all a large REFRIGERATOR.



2) Are we therefore able to build new functions or "logical gates" of Quantum Computing from whose composition it is possible to realize the programming of quantum algorithms?

- Yes, by exploiting a new probabilistic programming approach, linked to "unexplained" but demonstrable quantum phenomena such as quantum tunneling, entanglement, superposition, etc.



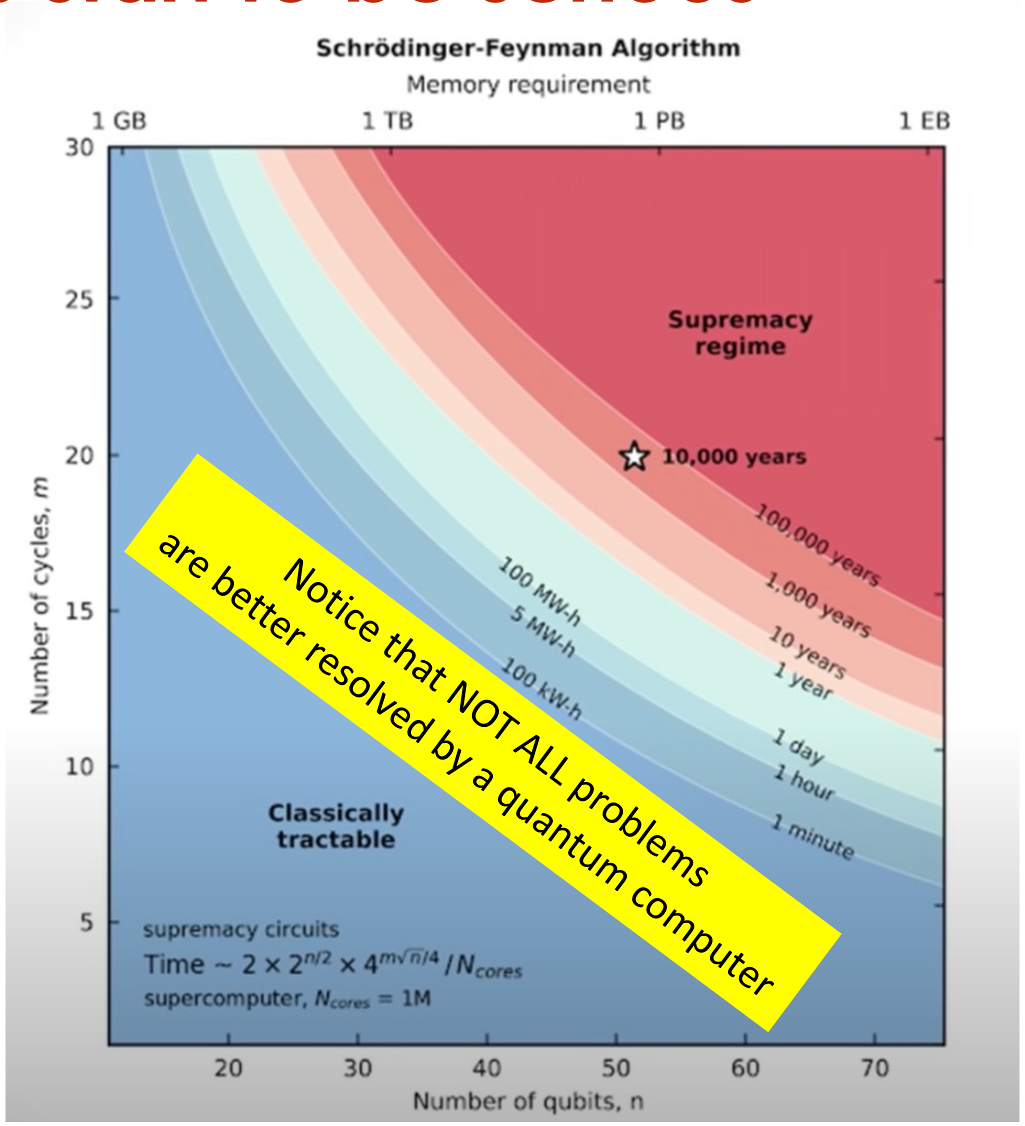
A new quantum processor (inside the refrigerator)

end of jokes... let’s start to be serious

The **Feynman's algorithm** is an algorithm that is used to simulate the operations of a quantum computer on a classical computer.
It is based on the Path integration formulation of quantum mechanics which was formulated by Richard Feynmann.

We will use something similar to express the algebraic equivalence between two computers: a classical and a quantum computer.

Then we will demonstrate the «quantum supremacy» on a very simple algorithm...
«a small step for a quantum computer.... but a giant leap for makind!»



Notice that NOT ALL problems are better resolved by a quantum computer

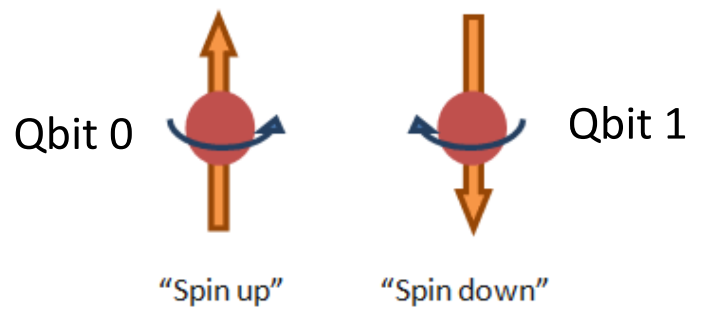
credits: Arvin Ash

Quantum Computing (algebraic approach to Qbit representation)

1) how to represent vectors of classical computer bit as Qbits? $|Dirac\ Vector\ Notation\rangle$

vector emulating Qbit 0 $\stackrel{\text{def}}{=} \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$ (Spin Up state of quantum particle)

vector emulating Qbit 1 $\stackrel{\text{def}}{=} \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$ (Spin Down state of quantum particle)



Review of algebraic rules of multiplication between matrices and vectors

$$\begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}ax+by\\cx+dy\end{pmatrix}$$

$$\begin{pmatrix}a & b & c\\d & e & f\\g & h & i\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}ax+by+cz\\dx+ey+fz\\gx+hy+iz\end{pmatrix}$$

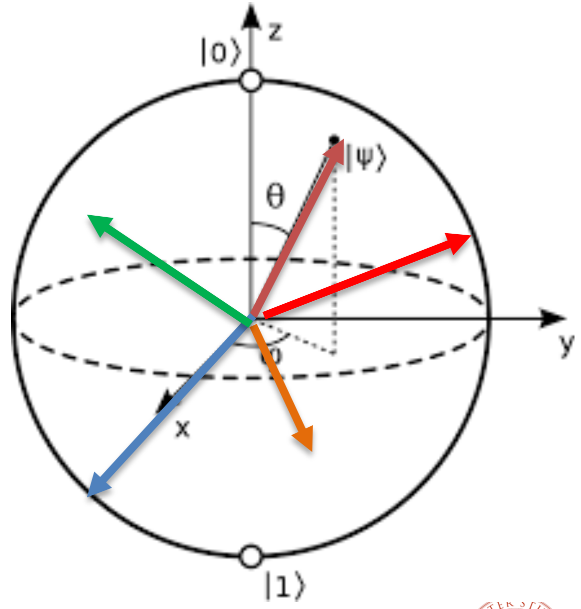
$$\begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}w & x\\y & z\end{pmatrix} = \begin{pmatrix}aw+by & ax+bz\\cw+dy & cx+dz\end{pmatrix}$$

Identity Matrix I

$$\boxed{\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}}\begin{pmatrix}a\\b\\c\\d\end{pmatrix} = \begin{pmatrix}a\\b\\c\\d\end{pmatrix}$$

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

swapped central rows of I



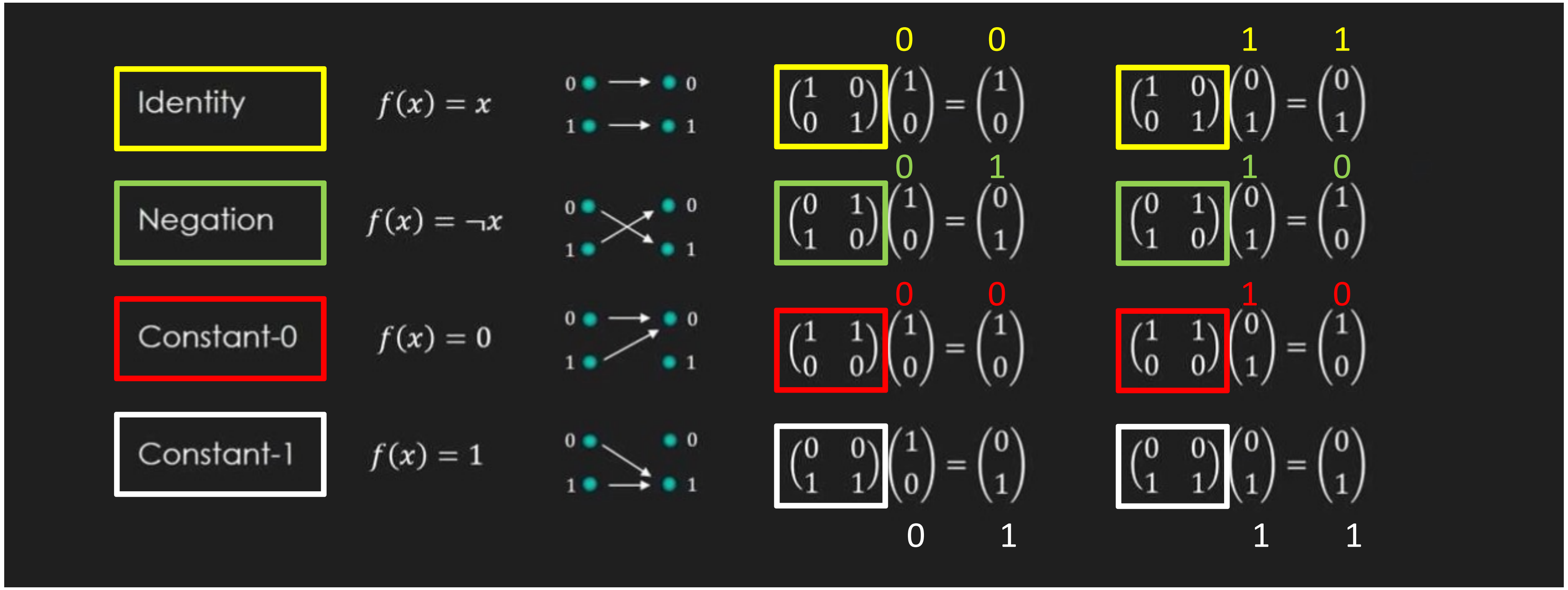
Bloch Sphere
(possible infinite Spin states)

Quantum Computing (operators on a single Cbit)

2) which are the four possible operations which we can execute on a SINGLE canonical bit (Cbit)? remind that

vector Qbit 0 $\overset{\text{def}}{=} \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$

vector Qbit 1 $\overset{\text{def}}{=} \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$



Identity	$f(x) = x$	$0 \to 0$, $1 \to 1$	$\begin{pmatrix}1&0\\0&1\end{pmatrix}\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}=\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}$	$\begin{pmatrix}1&0\\0&1\end{pmatrix}\overset{1}{\begin{pmatrix}0\\1\end{pmatrix}}=\overset{1}{\begin{pmatrix}0\\1\end{pmatrix}}$
Negation	$f(x) = \neg x$	$0 \to 1$, $1 \to 0$	$\begin{pmatrix}0&1\\1&0\end{pmatrix}\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}=\overset{1}{\begin{pmatrix}0\\1\end{pmatrix}}$	$\begin{pmatrix}0&1\\1&0\end{pmatrix}\overset{1}{\begin{pmatrix}0\\1\end{pmatrix}}=\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}$
Constant-0	$f(x) = 0$	$0 \to 0$, $1 \to 0$	$\begin{pmatrix}1&1\\0&0\end{pmatrix}\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}=\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}$	$\begin{pmatrix}1&1\\0&0\end{pmatrix}\overset{1}{\begin{pmatrix}0\\1\end{pmatrix}}=\overset{0}{\begin{pmatrix}1\\0\end{pmatrix}}$
Constant-1	$f(x) = 1$	$0 \to 1$, $1 \to 1$	$\begin{pmatrix}0&0\\1&1\end{pmatrix}\underset{0}{\begin{pmatrix}1\\0\end{pmatrix}}=\underset{1}{\begin{pmatrix}0\\1\end{pmatrix}}$	$\begin{pmatrix}0&0\\1&1\end{pmatrix}\underset{1}{\begin{pmatrix}0\\1\end{pmatrix}}=\underset{1}{\begin{pmatrix}0\\1\end{pmatrix}}$

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (reversibility of computation)

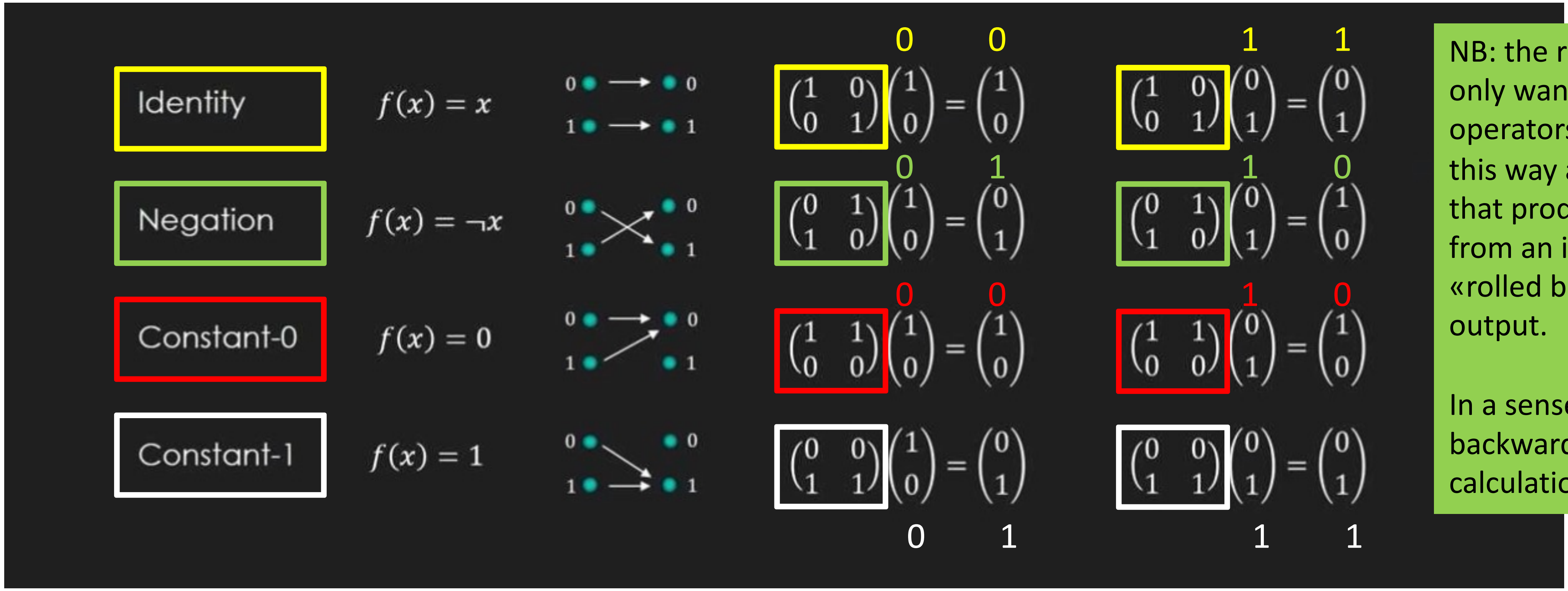
3) which of these are reversible operators? given the output and operator you can determine the input Qbit value

reversible: identity and negation (they just maintain or reverse the input bit on the output)

non reversible: Const-0 e Const-1 (delete input value and always overwrite with constant 0 or 1)

NB: in the figure we are using Cbit vectors which algebraically emulate a Qbit

RULE: Quantum Computers only adopt and implement reversible computation operators on Qbits



Identity
$f(x) = x$
Negation
$f(x) = \neg x$
Constant-0
$f(x) = 0$
Constant-1
$f(x) = 1$
$\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$ 0 0
$\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$ 1 1
$\begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$ 0 1
$\begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$ 1 0
$\begin{pmatrix}1&1\\0&0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$ 0 0
$\begin{pmatrix}1&1\\0&0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$ 1 0
$\begin{pmatrix}0&0\\1&1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$ 0 1
$\begin{pmatrix}0&0\\1&1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$ 1 1

NB: the reason why we only want reversible operators is because in this way all the calculation that produces the solution from an input can be «rolled back» from the output.

In a sense we can "go backwards in time" with calculation....

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (tensor product of vectors)

4) a tensor product of two vectors has the properties summarized below.

By making the tensor product of N vectors (which emulate N Qbits) we get a «state representation» vector of all the resulting possible binary values expressed by the combined associated Qbits.

So, what is the practical purpose of tensor product? to represent a binary sequence of many Qbits (and their possible states representations).

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \\ x_1 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{pmatrix}$$

NB: the tensor product produces all the binary enumerated joint states of the two Qbits

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \otimes \begin{pmatrix} z_0 \\ z_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 z_0 \\ x_0 y_0 z_1 \\ x_0 y_1 z_0 \\ x_0 y_1 z_1 \\ x_1 y_0 z_0 \\ x_1 y_0 z_1 \\ x_1 y_1 z_0 \\ x_1 y_1 z_1 \end{pmatrix}$$

NB: the tensor product produces all the binary enumerated joint states of the 3 bits

$$\begin{array}{ccccc} 1 & & 1 & & 0 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \otimes & \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \otimes & \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{array} = \begin{array}{cc} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}$$

NB: (x1,y1,z1) is a binary value on 3 bits

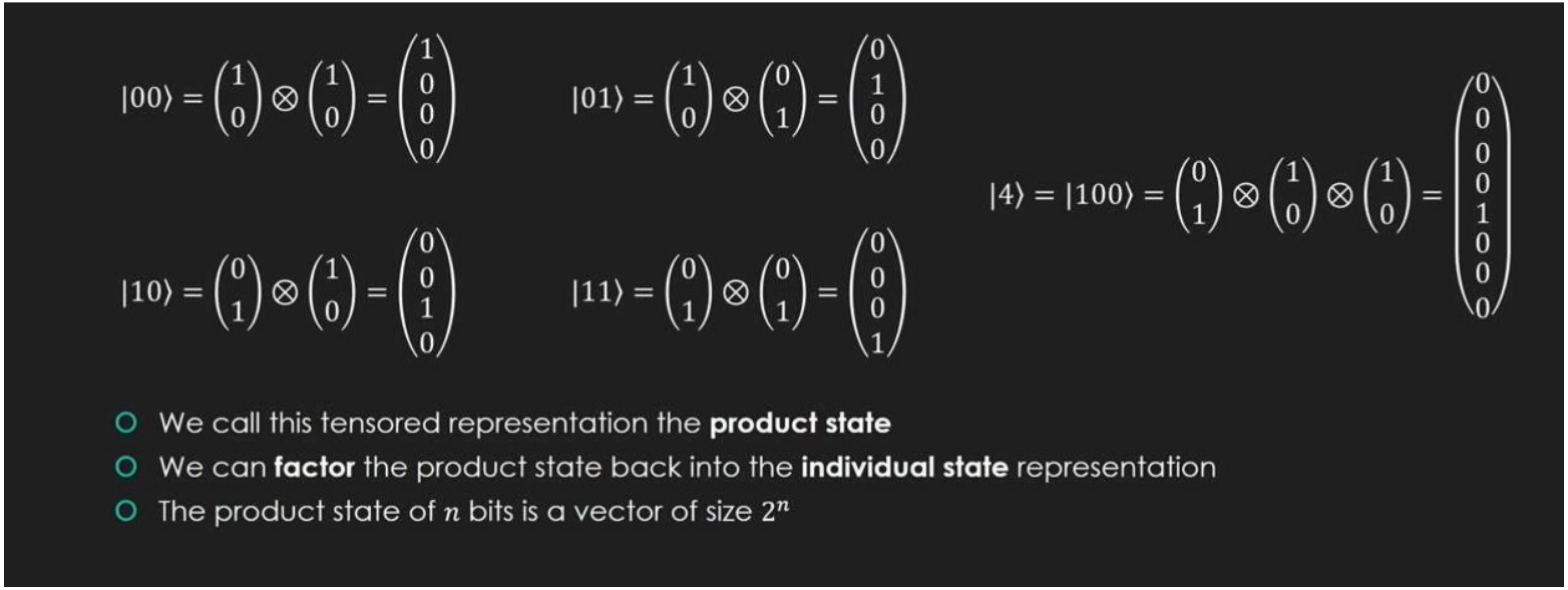
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (tensor product of multiple Qbit in a binary sequence)

5) the tensor product of multiple vectors (Qbit) is used to compute the «product state» of multiple consecutive Qbits.

see the example with all the 4 possible combinations of product tensor (product state)
of 2 Qbit $|00\rangle$ $|01\rangle$ $|10\rangle$ $|11\rangle$ or with 3 Qbit $|000\rangle$ $|001\rangle$ $|010\rangle$ $|011\rangle$ $|100\rangle$ $|101\rangle$ $|110\rangle$ $|111\rangle$

We are producing all the possible results for the state of 2 or 3 Qbit in sequence (binary sequence).



$$|00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad |01\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

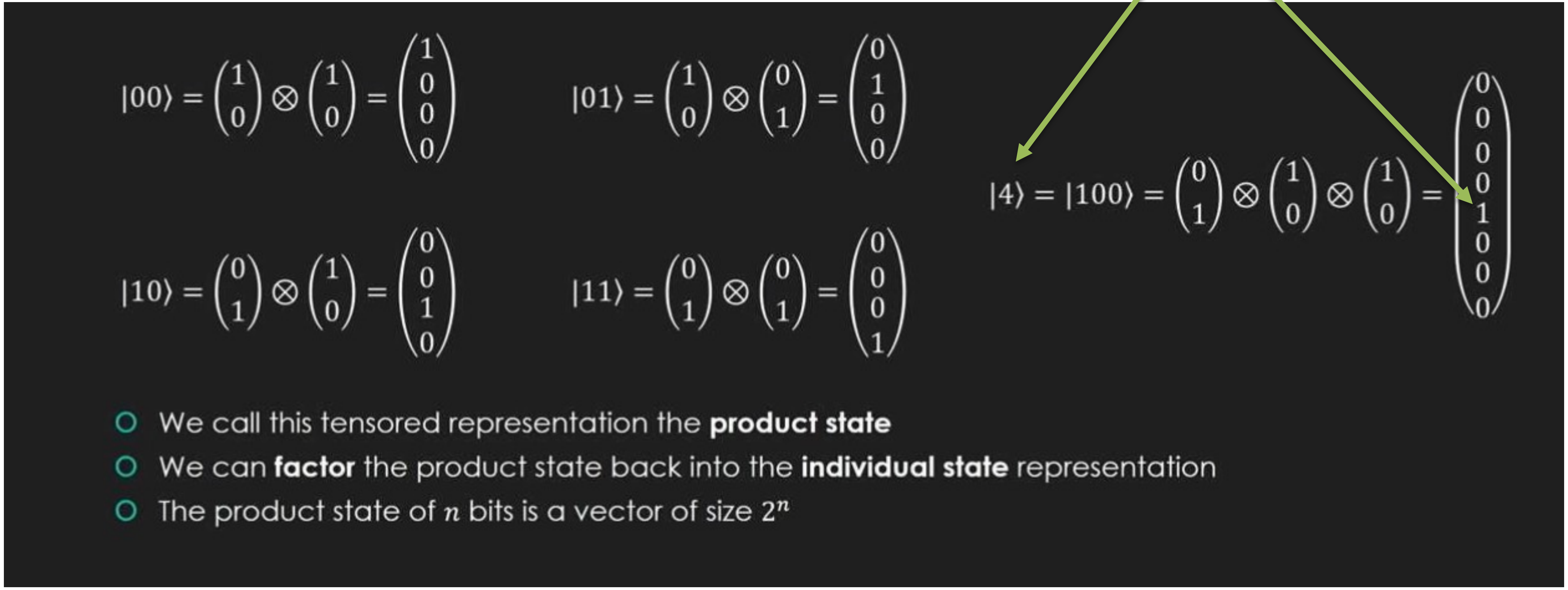
$$|4\rangle = |100\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$$

- We call this tensored representation the **product state**
- We can **factor** the product state back into the **individual state** representation
- The product state of n bits is a vector of size 2^n

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (tensor product of multiple Qbit in a binary sequence)

Let us now imagine that the state vector which lists all the possible states (combinations of binary values) of the N Qbits contains in each row the **probability** that the binary combination of the values of the N Qbits is precisely the one identified by the row. For example, HERE, the probability to have a binary result 4 is 100% and all the other results have probability 0%.



$$|00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad |01\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|4\rangle = |100\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

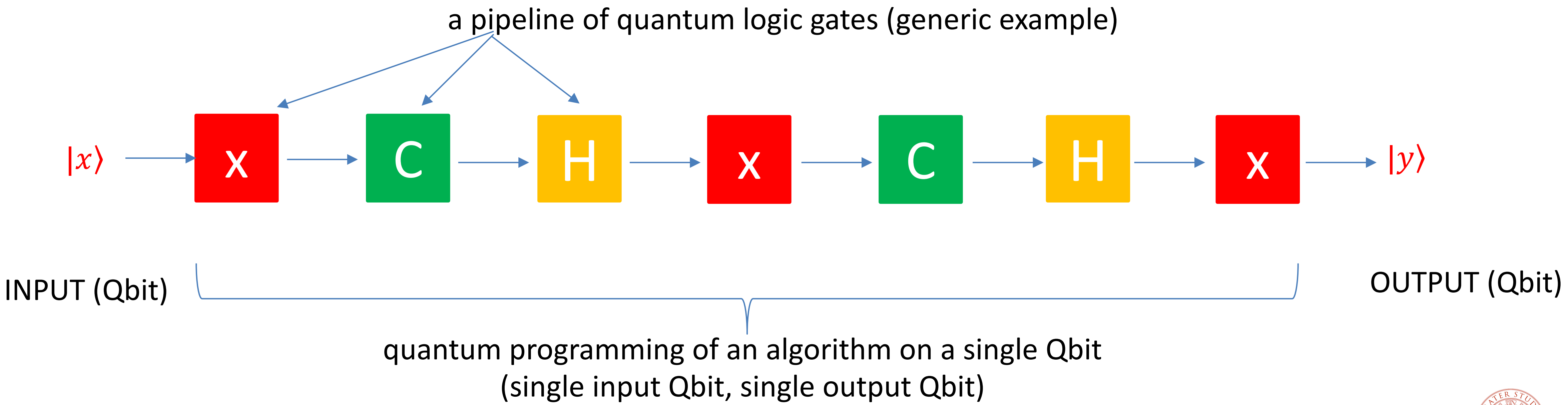
- We call this tensored representation the **product state**
- We can **factor** the product state back into the **individual state** representation
- The product state of n bits is a vector of size 2^n

credits: Quantum Computing for Computer Scientists, Microsoft

How to manipulate Qbit values? We need quantum logic gates....

At this point, are we able to build quantum logic gates that can manipulate the physical characteristics of the particles associated with the Qbit state?

If we were able to do this, we would have "transformation" operations of input into output on Qbit values, and we could use them to write quantum programs.

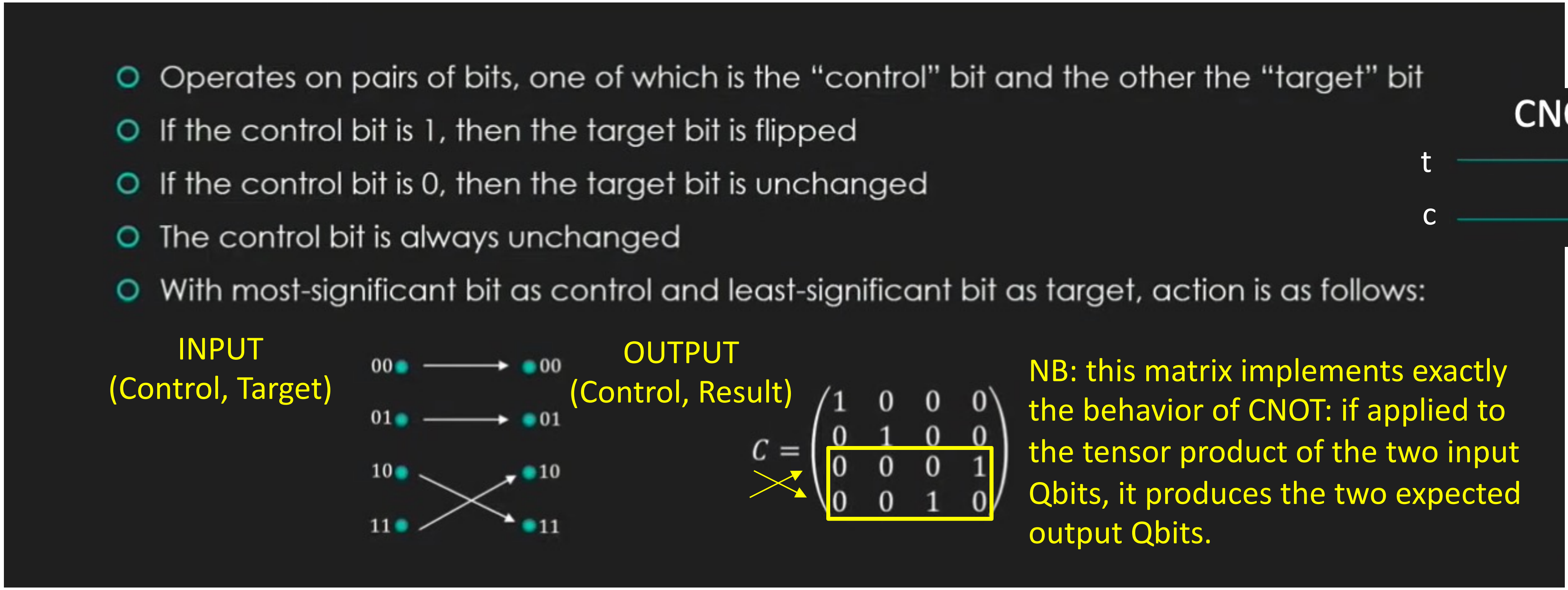

a pipeline of quantum logic gates (generic example)
|x⟩
x
C
H
x
C
H
x
|y⟩
INPUT (Qbit)
OUTPUT (Qbit)
quantum programming of an algorithm on a single Qbit
(single input Qbit, single output Qbit)

Quantum Computing (our first quantum logic gate: CNOT)

C

6) finally we have a way to create the first logic gate of the Quantum Computer (similar to the initial NAND-like building block of the classical computer): **the CNOT gate.**

The CNOT gate looks like a conditional choice and performs a quantum computation (operation) on 2 Qbits (c, t) which we can think of as follows: given INPUT (c,t), if c=0 then Output remains equal to (c, t), otherwise only t is inverted and Output = (c, not t).



- Operates on pairs of bits, one of which is the "control" bit and the other the "target" bit
- If the control bit is 1, then the target bit is flipped
- If the control bit is 0, then the target bit is unchanged
- The control bit is always unchanged
- With most-significant bit as control and least-significant bit as target, action is as follows:

INPUT
(Control, Target)

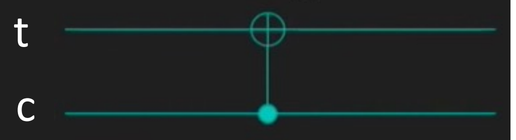
OUTPUT
(Control, Result)

$$\begin{tikzcd}
00 \arrow[r] & 00 \\
01 \arrow[r] & 01 \\
10 \arrow[dr] & 10 \\
11 \arrow[ur] & 11
\end{tikzcd}$$

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

NB: this matrix implements exactly the behavior of CNOT: if applied to the tensor product of the two input Qbits, it produces the two expected output Qbits.

CNOT gate



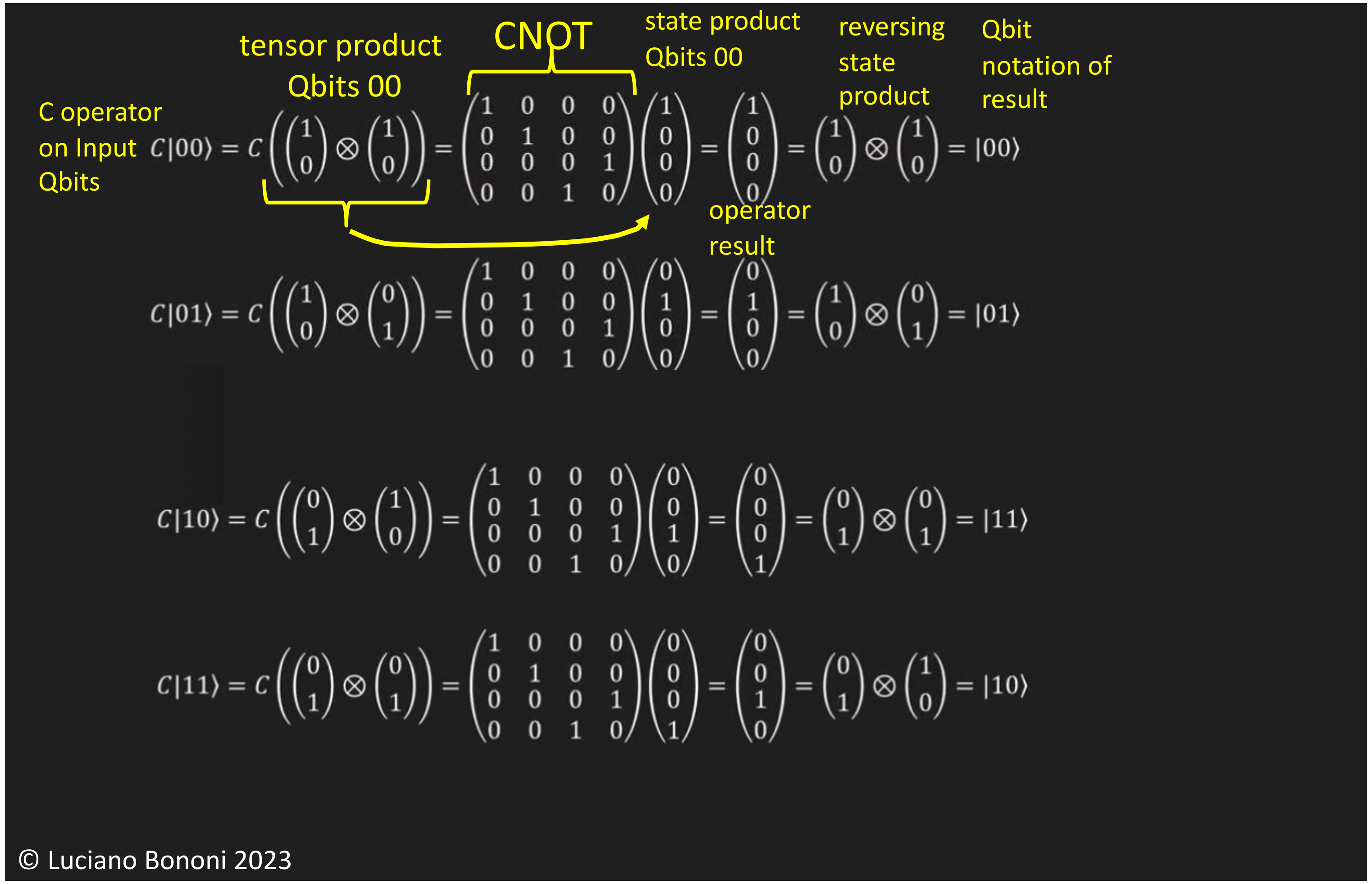
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (our first quantum logic gate: CNOT)

C

7) let's see a couple of applied cases of what happens by applying CNOT (C) to two pairs of Qbits:

NB: the final product state foresees all possible combinations of the result, but only one is true (probability 100%), and by reversing the state product we obtain the corresponding output result in Qbits.



$$C|00\rangle = C\left(\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix} = |00\rangle$$

$$C|01\rangle = C\left(\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix} = |01\rangle$$

$$C|10\rangle = C\left(\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix} = |11\rangle$$

$$C|11\rangle = C\left(\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix} = |10\rangle$$

We apply CNOT to $|00\rangle$ (control = 0 and target = 0) so output must be $|00\rangle$

We apply CNOT to $|01\rangle$ (control = 0 and target = 0) so output must be $|01\rangle$

We apply CNOT to $|10\rangle$ (control = 0 and target = 0) so output must be $|11\rangle$

We apply CNOT to $|11\rangle$ (control = 0 and target = 0) so output must be $|10\rangle$

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (quantum superposition)

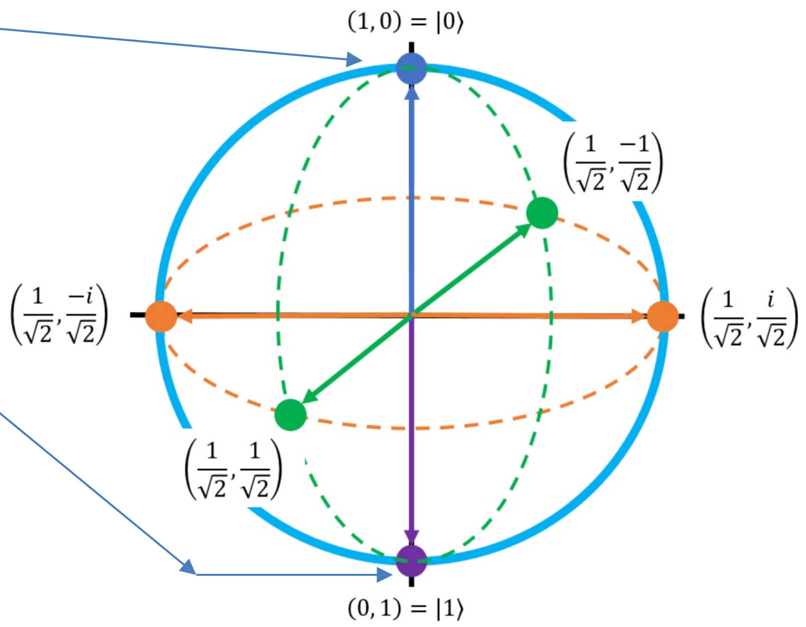
8. We can indeed use Qbits (because the vectors used up to now are in fact discrete emulations valid in the extreme values 0 and 1 of the real Qbits), and also Qbits in superpositions, as follows.

The real Qbit can be represented as $\binom{a}{b}$ where a,b are complex coefficients s.t. $||a||^2 + ||b||^2 = 1$

The extreme vector values used so far $\binom{1}{0}$ and $\binom{0}{1}$ are in fact compliant with the definition of Qbit above.

Other examples of interesting Qbits (you can verify that $||a||^2 + ||b||^2 = 1$)

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} \quad \begin{pmatrix}\frac{1}{2}\\ \frac{\sqrt{3}}{2}\end{pmatrix} \quad \begin{pmatrix}-1\\ 0\end{pmatrix} \quad \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}$$



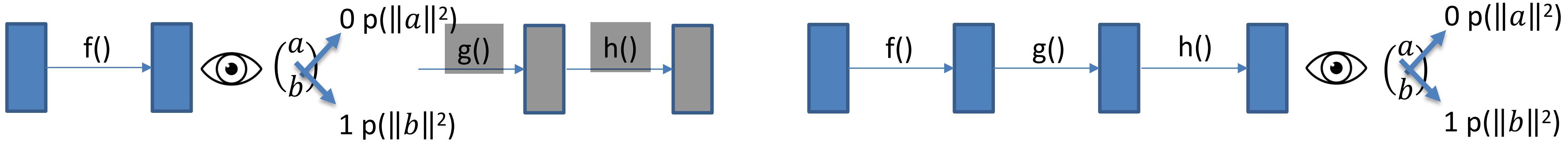
$(1,0) = |0\rangle$
$\left(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}},\frac{-i}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}},\frac{i}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)$
$(0,1) = |1\rangle$

Note that we can use irrational and negative coefficients (and complex coefficients as well but not here for simplicity). Every vector corresponds to a possible Spin direction in the Bloch Sphere.

Quantum Computing (quantum superposition)

8. we can put two or more Qbits in superposition. In this condition the value of the Qbit is no longer necessarily just 0 or 1, but it is somewhat Zero and somewhat One at the same time.

Fact: we don't know how much a Qbit is zero or one during quantum computation until we MEASURE the Qbit, but when we measure it we COLLAPSE its real value to one of two ideal values Zero or One with a certain probability, and from this moment the Qbit it will be consumed forever. For this reason, sometimes we measure the Qbits only at the end of the calculation process (not during), because by measuring them they collapse and be CONSUMED.



The blue box contains the Qbit, which is processed by the algorithm, but we don't know the value. Only when we look at the value (open the box) do we find the value, but we destroy the Qbit.

This is similar to the paradox of Schrödinger's cat.

Qbit measurement (observation): If a Qbit is measured while it is in the state $\binom{a}{b}$ then it will collapse to result value 0 with probability $||a||^2$ and result value 1 with probability $||b||^2$

Quantum Computing (quantum superposition)

For example, the Qbit in the state $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ it will have $\left\|\frac{1}{\sqrt{2}}\right\|^2$ = ½ probability to collapse in value 0 or 1.

Again, if we measure Qbit $\binom{0}{1}$ and $\binom{1}{0}$ which computational value will them produce and with which probability?

$\binom{0}{1}$ collapses to 1 with probability $\|b\|^2 = 1^2$ = 100% (that's why we used this as Qbit $|1\rangle$)

$\binom{1}{0}$ collapses to 0 with probability $\|a\|^2 = 1^2$ = 100% (that's why we used this as Qbit $|0\rangle$)

So basically we cannot do intermediate checks on the calculated values anymore? FALSE

If we carry out checks we consume Qbits that we will no longer be able to use having caused them to collapse. But in reality we no longer have to choose just one path from the tree of choices, but we go down all the paths at the same time (therefore less checks are needed)!

Great Computational Advantage (if the problem is suitable) because we will obtain in the Qbits all the possible solutions to the problem (and not just one) in a max number of computation steps which is congruous to the height of the computational tree (that is, logaritmic with respect to the number of possible computational states).

Quantum Computing (quantum superposition)

Similarly, multiple Qbits are similarly solved as a product of tensors ($\otimes$).

For example: if we wanted to toss two coins together (heads or tails) and we wanted to calculate all possible outcomes, just program two Qbits that produce 50% heads or tails and make them the product of the tensors:

$$\begin{array}{ccc} A & \otimes & B \end{array}$$

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

(note that $\|1/2\|^2 = ¼$, and $¼ + ¼ + ¼ + ¼ = 1$.

So measuring the two Qbits A and B after the execution of the tosses we will see both of them collapsed in a state vector showing <u>only one</u> of the results $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with ¼ probability each.

This calculation is consistent with all the results we could get from infinite tosses of two real coins.

Quantum Computing (further manipulate Qbits: bit-flip Gate)

At this point we can manipulate the calculation (implementing the program or quantum algorithm) on the Qbits (data structure of the algorithm) in the same way seen for the emulation vectors, i.e. using operators (in the form of a matrix).

We have operators eg. CNOT, etc. and many of these only make sense in the quantum context.

The operators in matrix form model the effects of physical devices which in quantum reality manipulate the Spin/Polarization of the super-fluid particles, but without measuring/observing their value and therefore without collapsing and consuming them.

For example: this is the operator programmed to bit-flip a single Qbit: X $\binom{a}{b} = \binom{b}{a}$

$\boxed{X}$

$\boxed{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

example : $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}$

The input (Red) Qbit was value ONE with 3/4 and ZERO with ¼ probability, but after the bit-flip it became the (Blue) output Qbit with value ONE with 1/4 and ZERO with 3/4 probability.

Intuitively, if at some point in the computation the Qbit had a value of zero or one with certain probabilities, without observing or consuming it, the bit flip $\boxed{X}$ transforms the Qbit (changes its state) with the same probabilities of assuming the values one and zero, but in reverse order (flip).

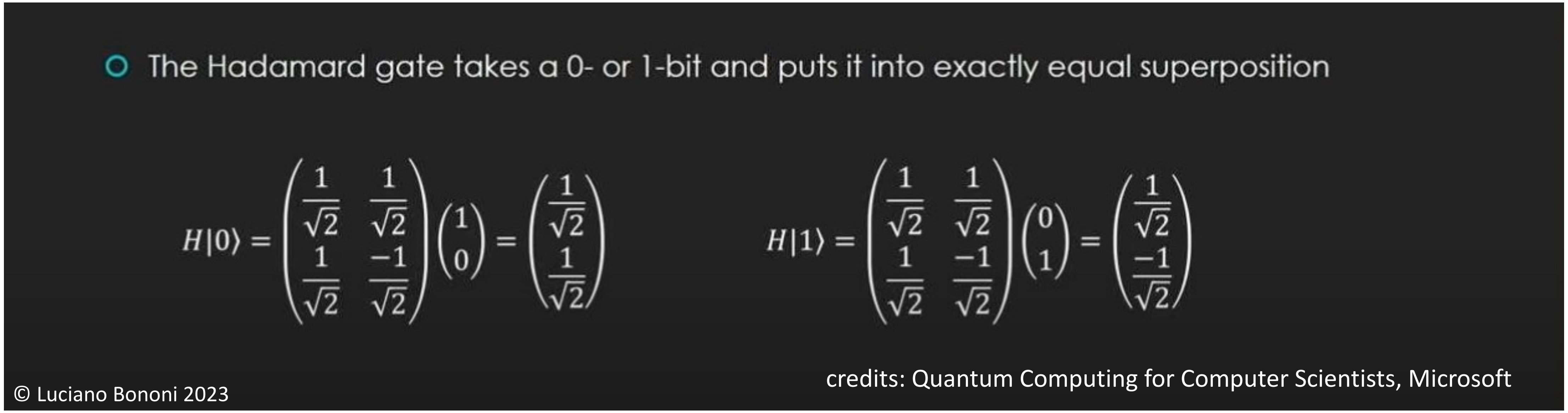
Quantum Computing (Hadamard Gate)

A fundamental operator for the programming of quantum computing is the **Hadamard Gate**.

This operator H takes as input a Qbit (with any value, even 0 or 1) and places it in the state of exact superposition (i.e. in statistical equilibrium between the two states 0 and 1, with probability ½ of collapsing on the value 0 and ½ to collapse to the value 1). It is equivalent to a sort of reset of the state of a Qbit which will be able to advance in the calculation being a bit 0 and a bit 1, but the important thing is that in reality the Qbit knows how to go back (the **operation is reversible**).

Hadamard gate H = $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$ (Note the negative value. In your opinion, why you need the negative?)

Because this makes H reversible! Try it with H H $|0\rangle = |0\rangle$ e H H $|1\rangle = |1\rangle$. Noteworthy!



- The Hadamard gate takes a 0- or 1-bit and puts it into exactly equal superposition

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

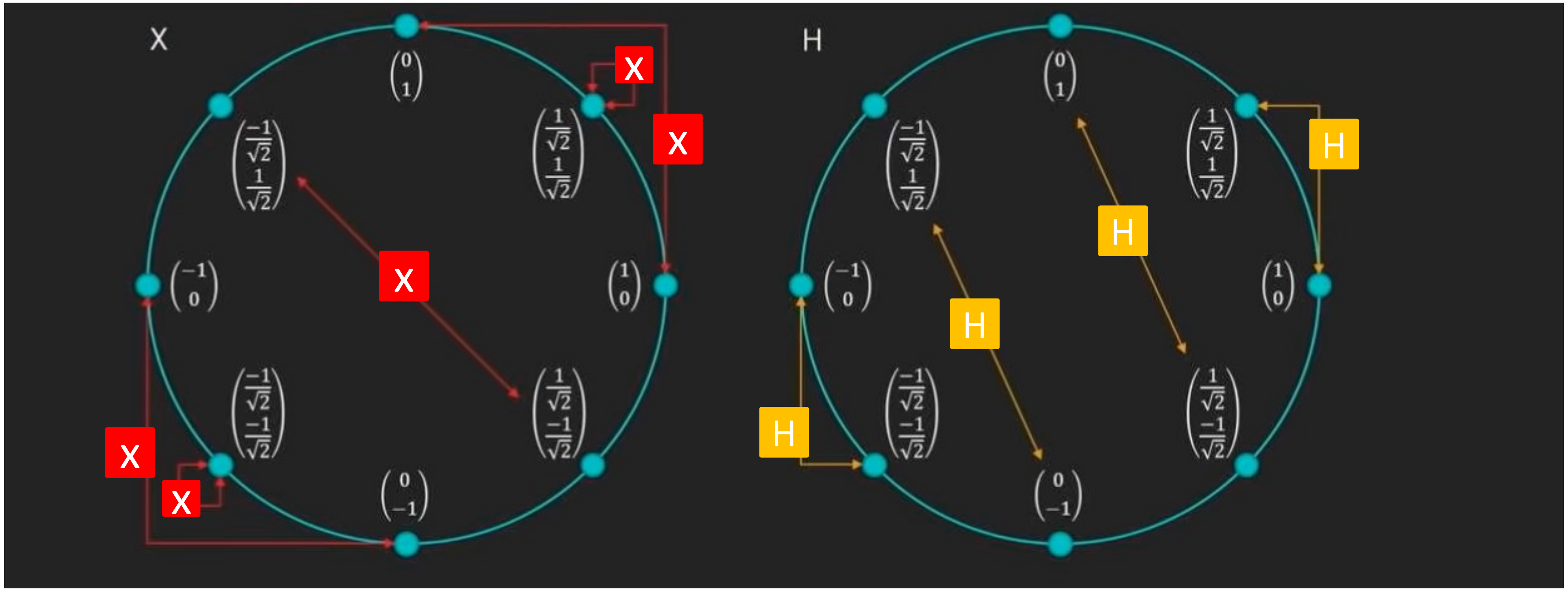
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Unit Circle State Machine)

We can now summarize the interesting states and transitions that we are able to implement for programming a Qbit. The map of its possible Spin states represented on the Unit Circle (a 2D projection of the Bloch sphere) for computer scientists is equivalent to looking at the states and transitions of a programmable finite state machine.

Bit Flip Operator (X)

Hademard Operator (H)



X
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{-1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}0\\-1\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\\frac{-1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}-1\\0\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix}$
X
X
X
X
X
H
H
H
H
H

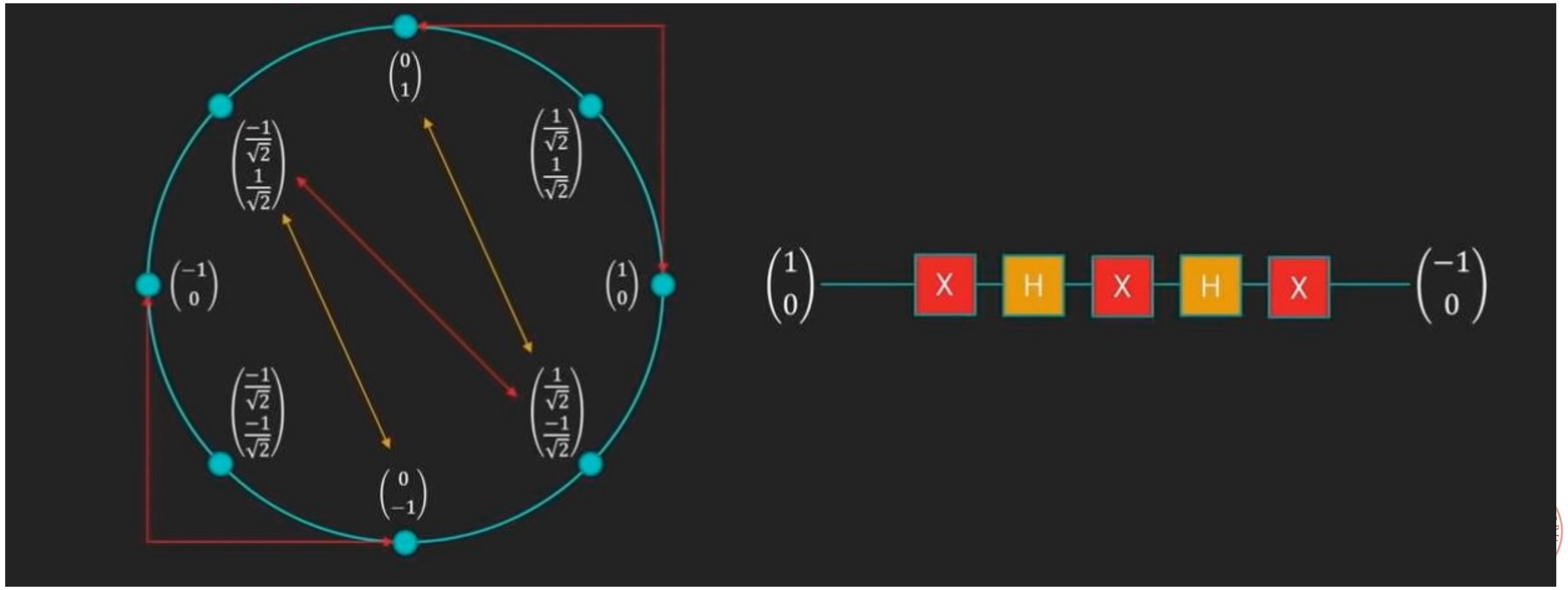
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Quantum Circuit Notation)

How can we imagine a quantum computation in an algorithm implemented by sequence operators starting from an initial Qbit of input and to produce a final Qbit as output?

Through a navigation of the state machine that follows the transformation steps on the Qbit caused by operators. Note that the calculation process is reversible (the calculation occurs in both directions).

From $\binom{1}{0}$ we make a BitFlip (X) and reach $\binom{0}{1}$ then we make Hademard H which reset in superposition state $\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}$ then again we make X, leading to a $\begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$, then again H leading to a $\binom{0}{-1}$ and finally X leading to final value $\binom{-1}{0}$



$\binom{0}{1}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$
$\binom{1}{0}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}$
$\binom{0}{-1}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}$
$\binom{-1}{0}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$
$\binom{1}{0}$
X
H
X
H
X
$\binom{-1}{0}$

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Summary)

Summarizing everything we've seen so far:

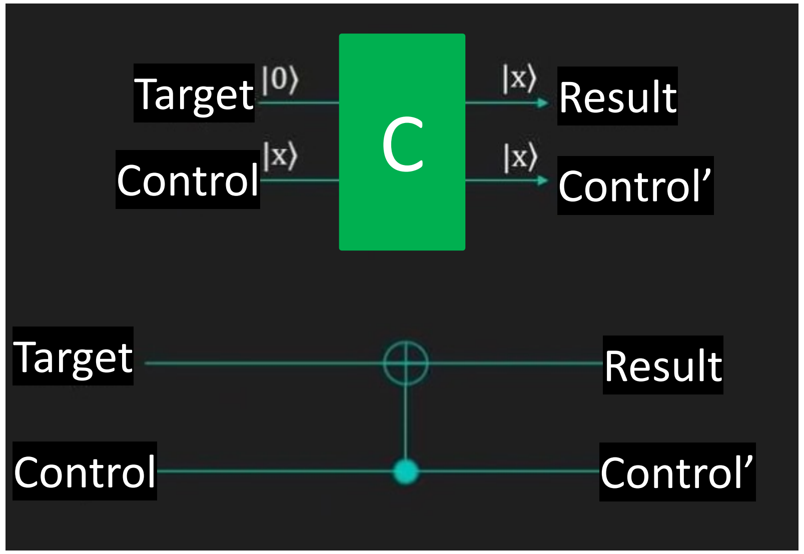
- we modeled the Qbits as binary vectors of complex values
- the classic bits (that assume only the canonical values 0 and 1) we call them Cbits.
- Cbits are special cases of Qbits and can be modeled as vectors of 2 complex numbers
- Qbits can be superimposed, and if measured they collapse probabilistically into Cbits, with certain probabilities depending on the quantum state reached by the Qbit.
- Systems composed of multiple Qbits are tensor products of single Qbit systems (as sequenced Cbits)
- We can model the operators (expressed as multiplicative matrices) on Qbits (expressed as vectors) in the way shown with Cbits.
- The Hadamard gate brings Qbits in overlapping (superposition) between the states, and brings them back to the original value (being reversible)
- Qbits and their operations can be interpreted as algorithms running on state machine (on unit circle, if we use real values, and on Bloch sphere if we also use complex values).

«Algorithmic» emulation of (CNOT) C

CNOT gate: Control	Target	Control'	Result
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

NB: here in (Control == 1) we observe the Control Qbit! However, we created a copy before to let the value to go forward...

```
Control' = Identity (Control)
If (Control == 1) then
            result = Negation (Target)
              else
            result = Identity(Target)
```



credits: Quantum Computing for Computer Scientists, Microsoft

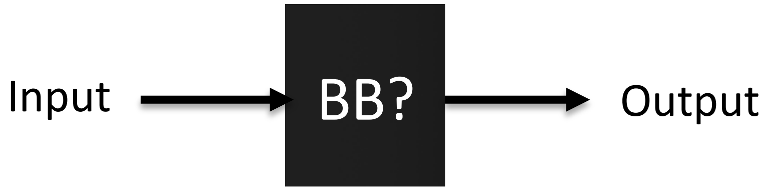
This slide is shown just to make it clear that the quantum gate CNOT it is actually usable according to a classic programming principle whose semantic is more "recognizable" to classic Computer Scientists.

limited example of Quantum Supremacy?

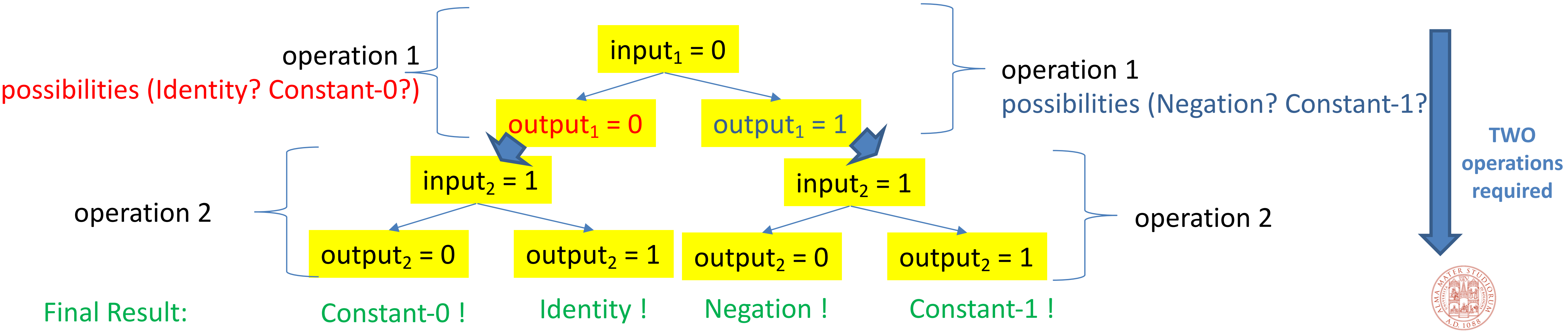
Let’s now define a suitable problem to test the (limited example of) quantum supremacy

The Deutsch Oracle problem (or the problem of the «misterious» function)

Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function. We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1, but which is the one in BB?

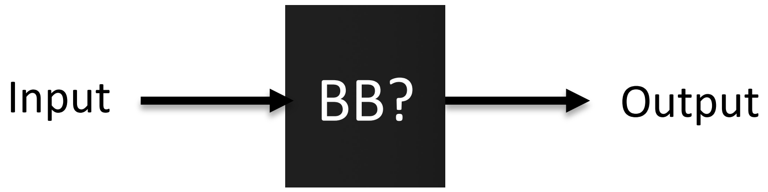


Problem: how many operations must be done to know the answer on a **Classical Computer?**
We can only try to supply input values and observe Output to figure out what function it is.



The Deutsch Oracle problem

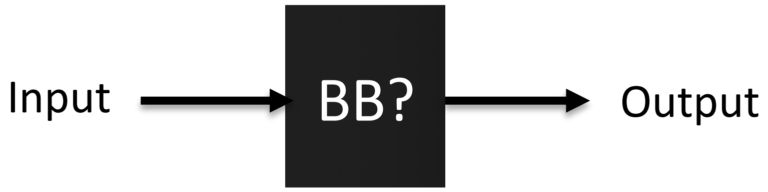
Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function. We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1, but which is the one in BB?



Problem: how many operations must be done to know the answer on a **Quantum Computer?**
We can only try to supply input values and observe Output to figure out what function it is.

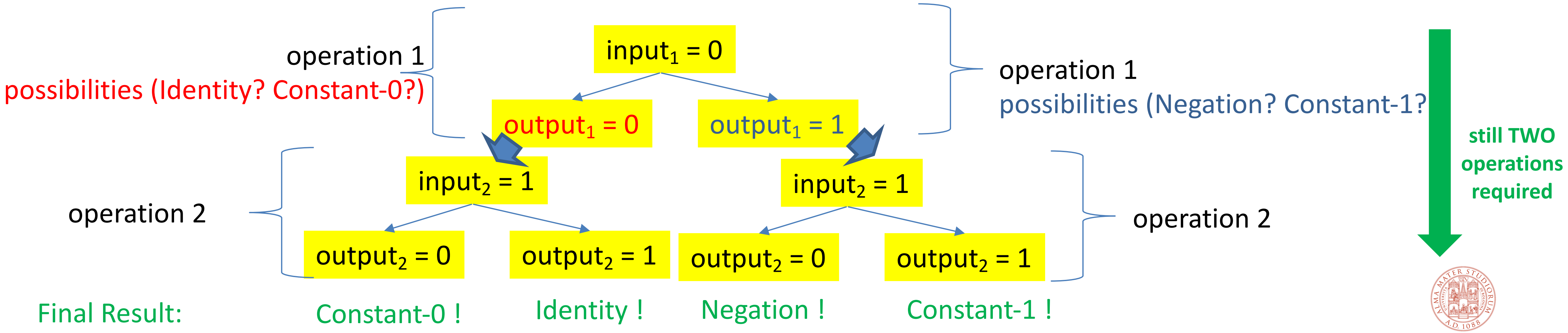
The Deutsch Oracle problem

Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function. We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1, but which is the one in BB?



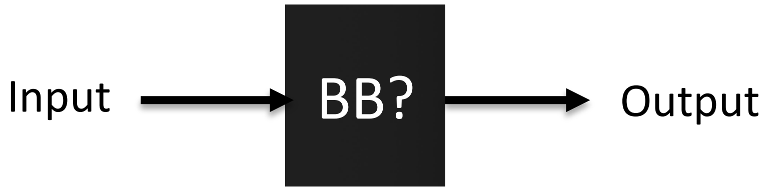
Problem: how many operations must be done to know the answer on a **Quantum Computer?**

If we use only one input Qbit, there are still TWO operations required!



The Deutsch Oracle problem

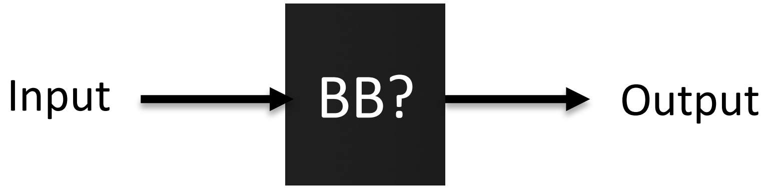
Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function. We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1, **but the new question is: is the one in BB a constant or a variable function?**



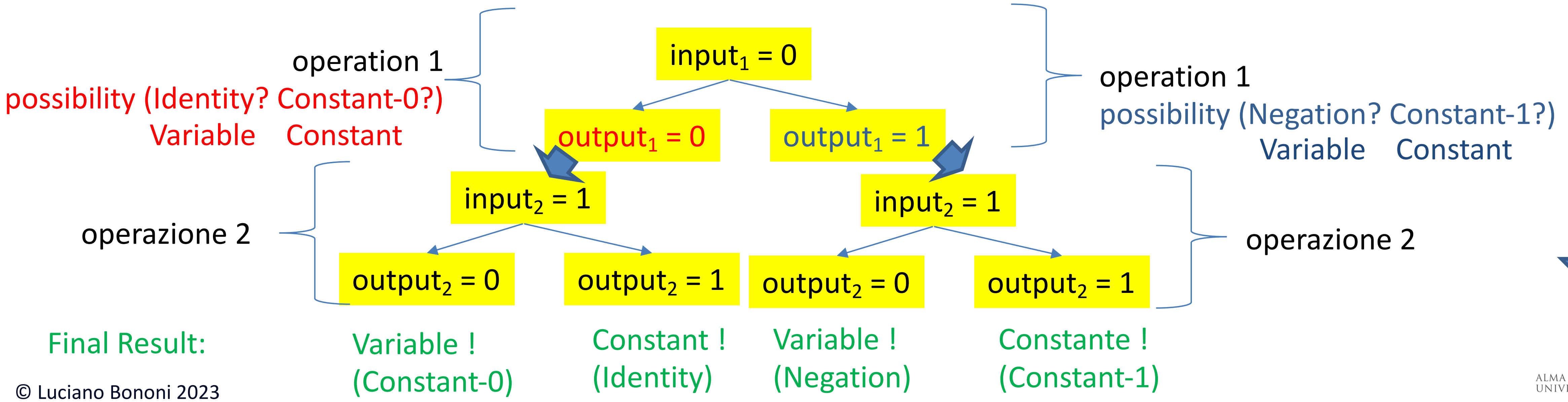
Problem: how many operations must be done to know the answer on a **Classical Computer?**
We can only try to supply input values and observe Output to figure out what function it is.

The Deutsch Oracle problem

Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function.
We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1,
but the new question is: is the one in BB a constant or a variable function?



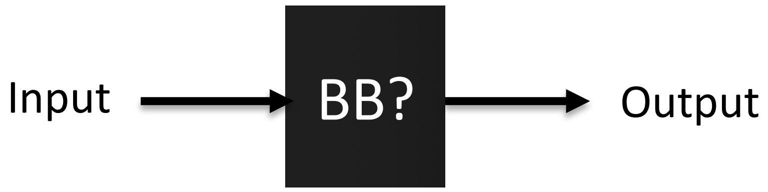
Problem: how many operations must be done to know the answer on a **Classical Computer?**
We can only try to supply input values and observe Output to figure out what function it is.



TWO Operazions required

The Deutsch Oracle problem

Let's imagine that we are given a black box (Black Box = BB) that implements a one-bit function.
We know that the function will be one of 4 possible: Identity, Negation, Constant-0, Constant-1,
but the new question is: is the one in BB a constant or a variable function?

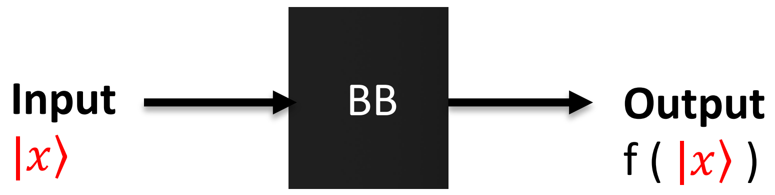


Problem: how many operations must be done to know the answer on a **Quantum Computer?**
Note that the answer is **between two possibilities:** one Qbit is enough to distinguish them!
So just do programming operations using **a single superimposed Qbit** as input!
Let’s also cut computing time in half thanks to the power of Quantum Computing....

Q: but how do you program the question?
A: first we have to define how each of the 4 possible functions in BB can be realized on a Quantum Computer...

The Deutsch Oracle problem

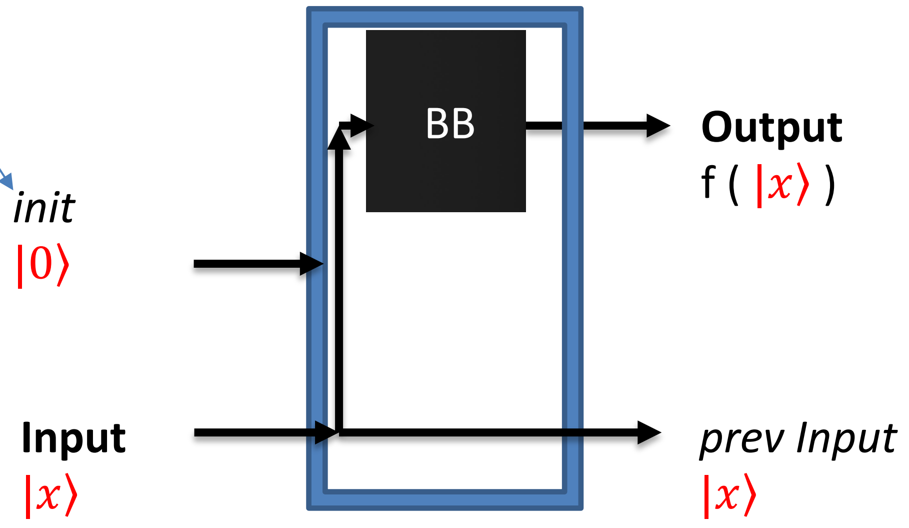
Problem: Constant-0 and Constant-1 functions are NOT REVERSIBLE! (in fact they overwrite the history)



Trick: We add an extra Qbit of output just to remember «where we came from».

Then we need to "rewire" or reprogram our Black Box (BB) quantum circuit as follows:

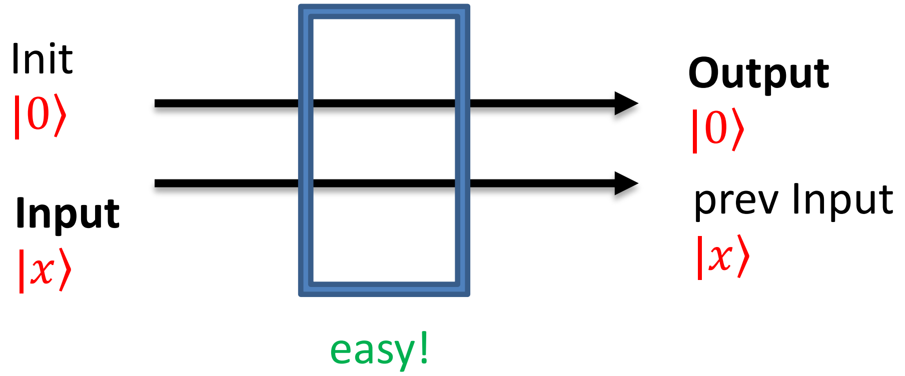
NB: this additional «input» we add (called Init) is always set by us to $|0\rangle$, so it is technically just a static constant input of the quantum circuit.
It is just like defining an initialised constant in a program, hence it is not a additional input of the problem.

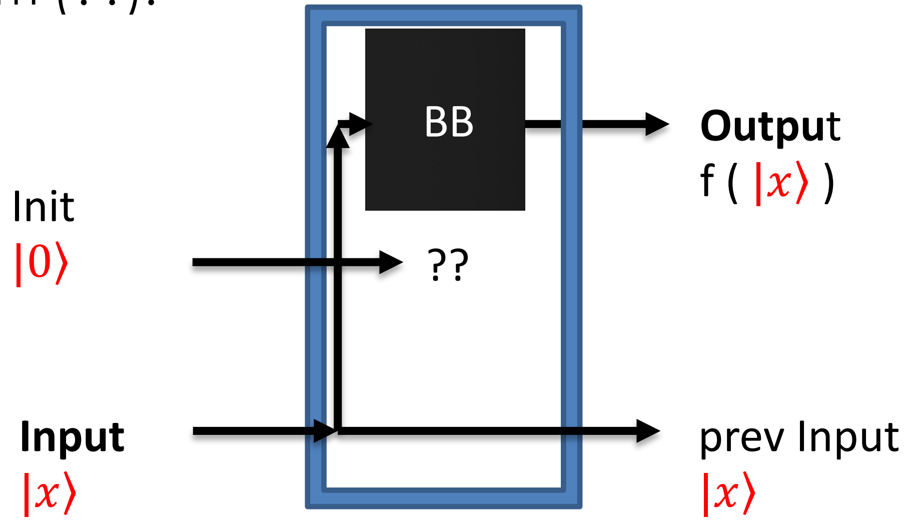


The Deutsch Oracle problem

Let's see how to rewire the two functions Constant-0 and Constant-1 so that they produce the general function model on the right (two Qbits) by solving the internal form (??):

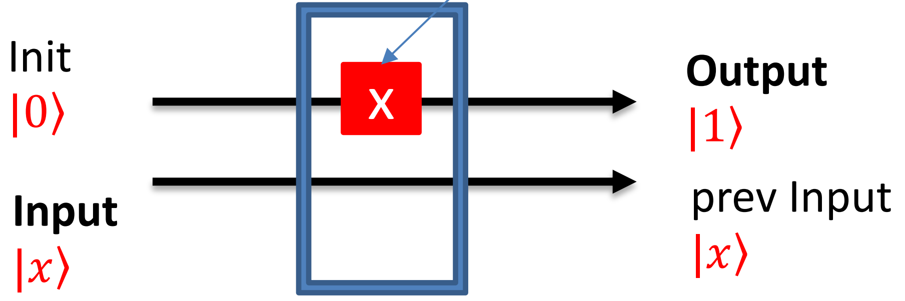
rewired circuit Constant-0





rewired circuit Constant-1

NB: we can use the BitFlip (x) gate of a Qbit

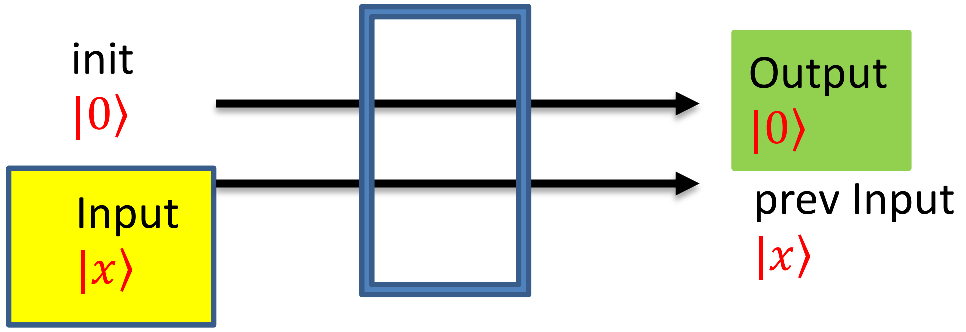


easy by using X internally as shown

The Deutsch Oracle problem

Now let's see if the two rewired functions Constant-0 and Constant-1 are reversible?

rewired circuit Constant-0



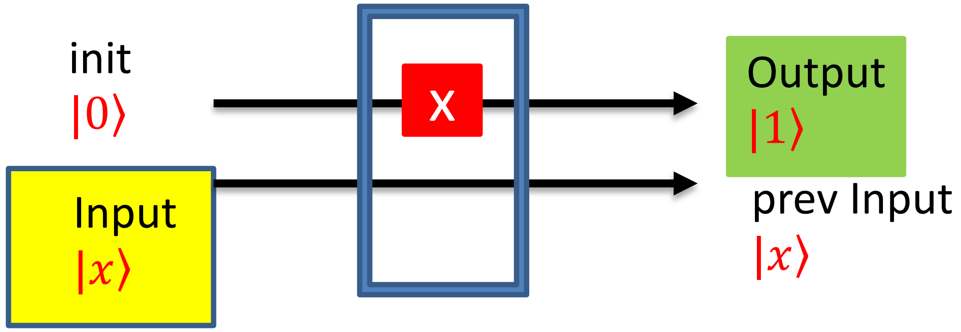
Let's see the two possible cases with Constant-0:

If Output = $|0\rangle$ and prev Input = $|0\rangle$
then input was Init $|0\rangle$ and Input $|0\rangle$

If Output = $|0\rangle$ and prev Input= $|1\rangle$
then input was Init $|0\rangle$ and and Input $|1\rangle$

OK!
REVERSIBLE
computation
despite constant
OUTPUT to $|0\rangle$

rewired circuit Constant-1



Let's see the two possible cases with Constant-0:

If Output = $|1\rangle$ and prev Input = $|0\rangle$
then input was Init $|0\rangle$ and Input $|0\rangle$

If Output = $|1\rangle$ and prev Input= $|1\rangle$
then input was Init $|0\rangle$ and and Input $|1\rangle$

OK!
REVERSIBLE
computation
despite constant
OUTPUT to $|1\rangle$

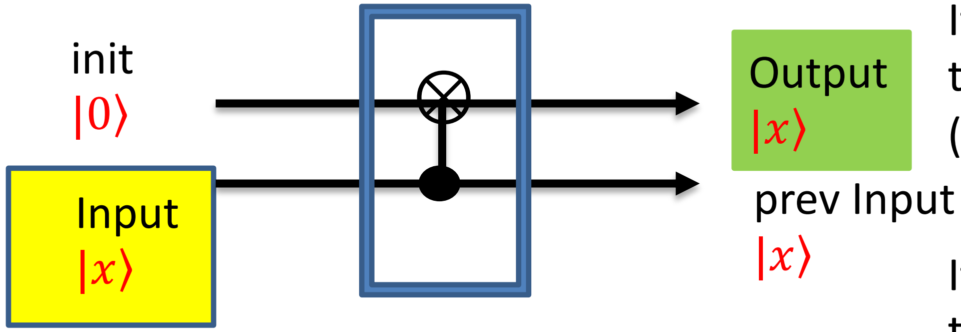
The Deutsch Oracle problem

Now let's rewire the Quantum Computing internal circuit for the function Identity e Negation as well (note these are already reversible)

rewired Identity

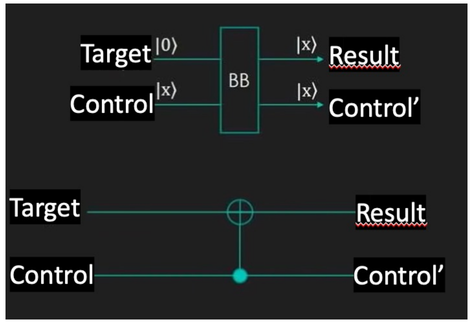
🟩 C

NB: this internal rewire for Identity is in reality a CNOT between the two inputs

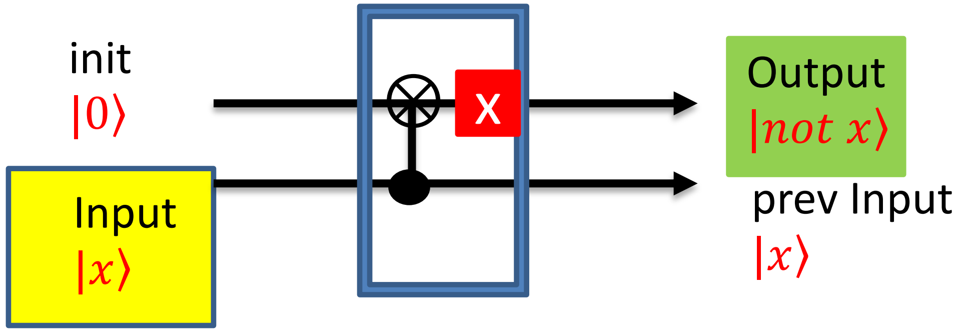


If Input (Control) is $|0\rangle$ (and init (Target) is $|0\rangle$) then Output (Result) is $|0\rangle$ and prev Input (Control') is $|0\rangle$

If Input (Control) is $|1\rangle$ (and init (Target) is $|0\rangle$) then Output (Result) is $|1\rangle$ and prev Input (Control') is $|1\rangle$



rewired Negation

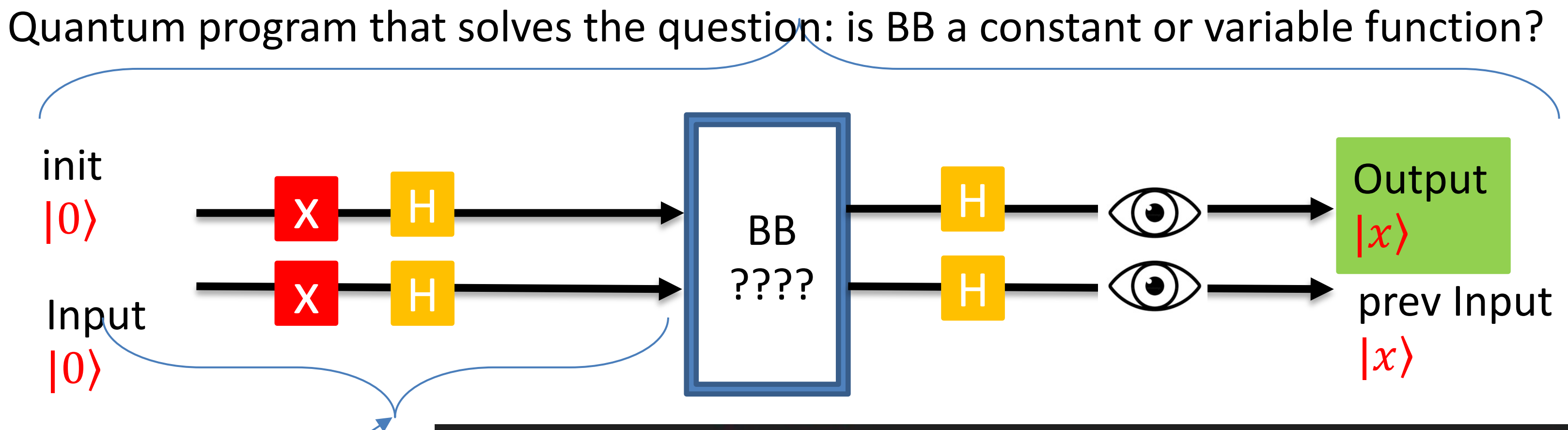


trivially Negation is the previous Identity circuit (CNOT) with output negated (by a flip-bit gate)

🟩 C 🟥 X

The Deutsch Oracle problem

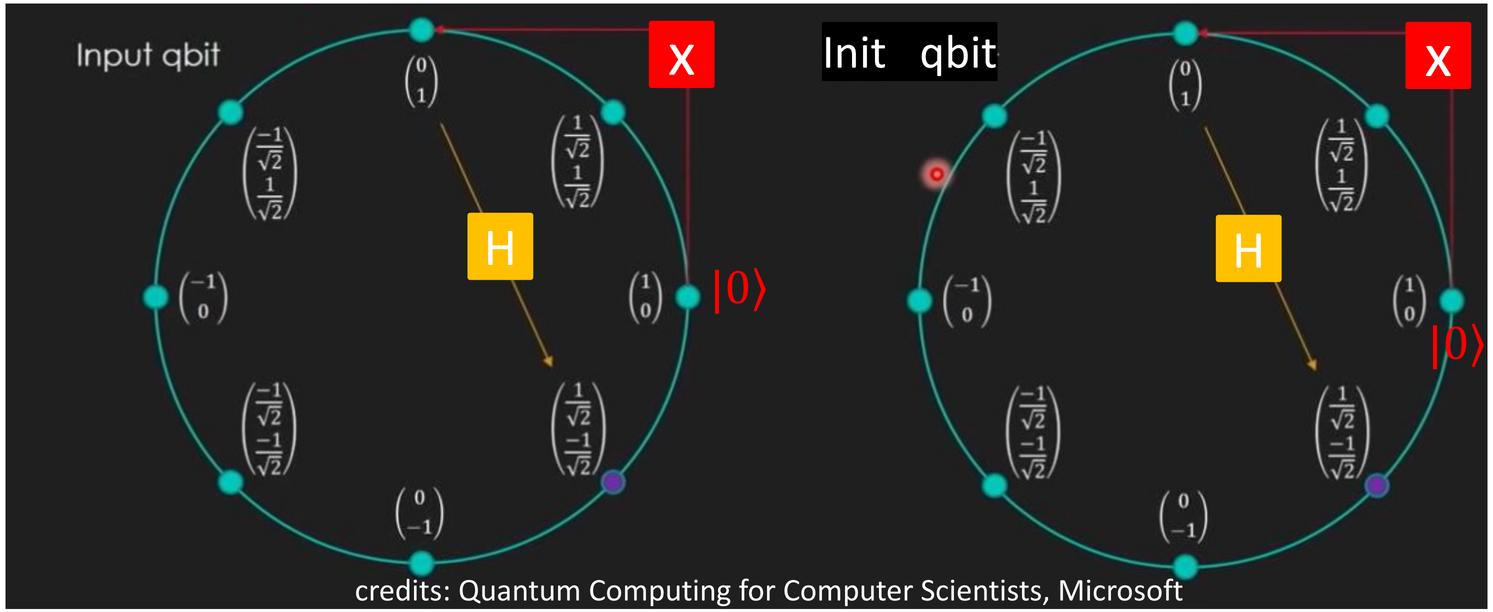
Now, if these are the 4 possible (all made reversible) functions that we could find implemented in the Black Box, how do we program the solution that tells us whether the function BB is constant or variable in one step on the Quantum Computer?



Let’s look the pre-processing.

This leads to state

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$


Input qbit
X
H
$|0\rangle$
Init qbit
X
H
$|0\rangle$
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt2}\\\frac{1}{\sqrt2}\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt2}\\\frac{-1}{\sqrt2}\end{pmatrix}$
$\begin{pmatrix}0\\-1\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt2}\\\frac{-1}{\sqrt2}\end{pmatrix}$
$\begin{pmatrix}-1\\0\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt2}\\\frac{1}{\sqrt2}\end{pmatrix}$

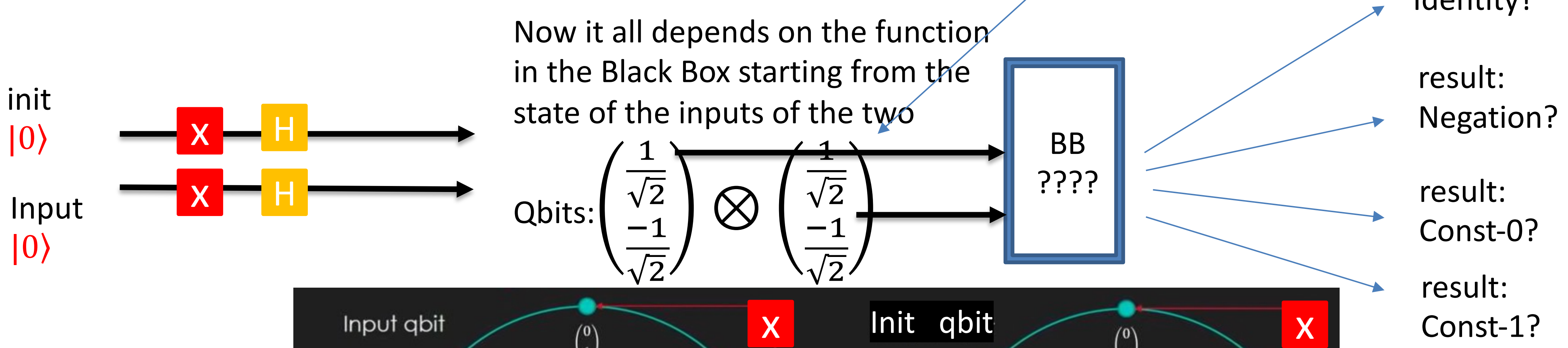
credits: Quantum Computing for Computer Scientists, Microsoft

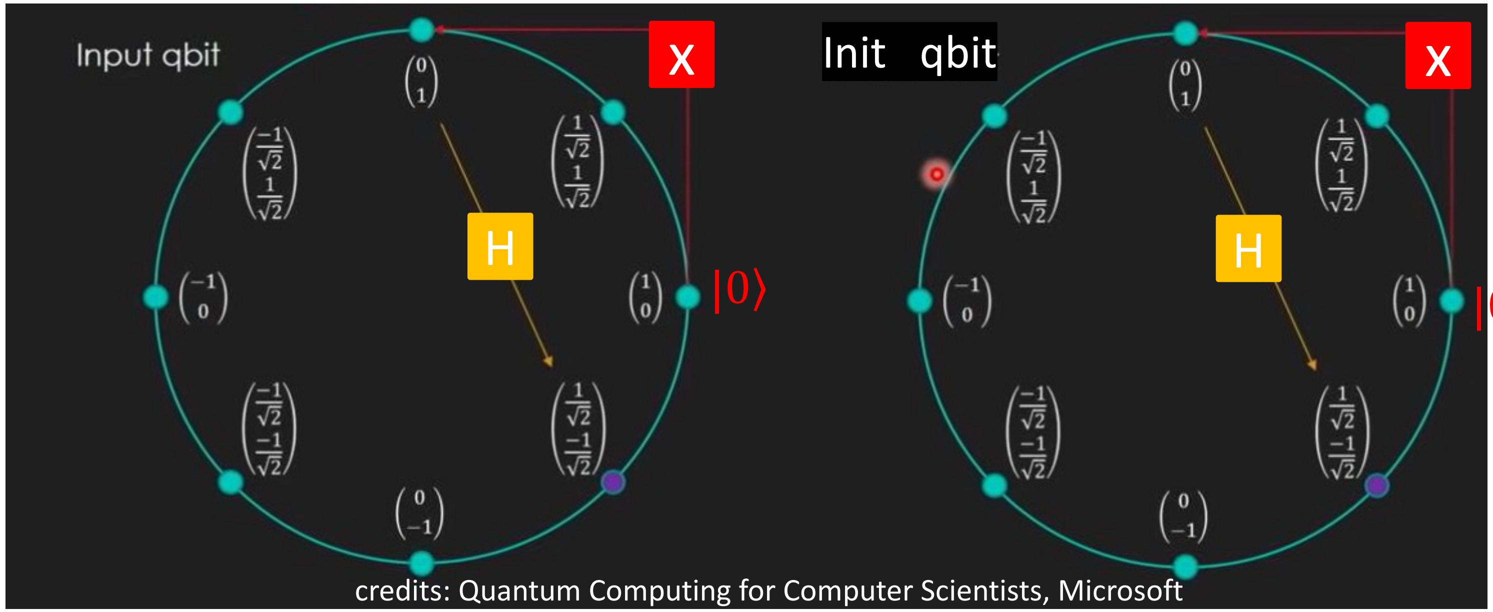
The Deutsch Oracle problem

Now, let's see how the function BB modifies in all 4 possible ways the states of the input Qbits transformed by the pre-processing.

Now it all depends on the function in the Black Box starting from the state of the inputs of the two

Qbits: $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$



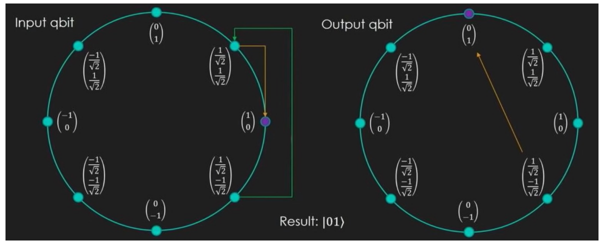

Input qbit
Init qbit
X
H
$|0\rangle$
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$
$\begin{pmatrix}-1\\0\end{pmatrix}$
$\begin{pmatrix}0\\-1\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}\frac{-1}{\sqrt{2}}\\\frac{-1}{\sqrt{2}}\end{pmatrix}$
$\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{-1}{\sqrt{2}}\end{pmatrix}$

credits: Quantum Computing for Computer Scientists, Microsoft

The Deutsch Oracle problem

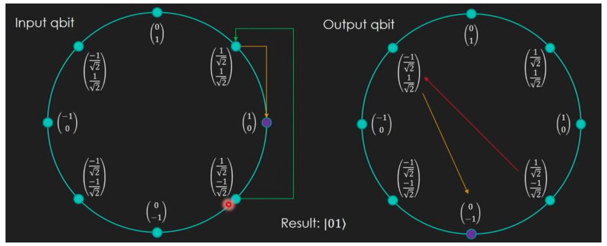
Let's see the results of the State Vector for the 4 possible functions starting from the input of the operation:

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} = \overset{\text{CNOT}}{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}} \begin{bmatrix}1/2\\-1/2\\-1/2\\1/2\end{bmatrix} = \begin{bmatrix}1/2\\-1/2\\1/2\\-1/2\end{bmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \quad \boxed{H} = |01\rangle$$



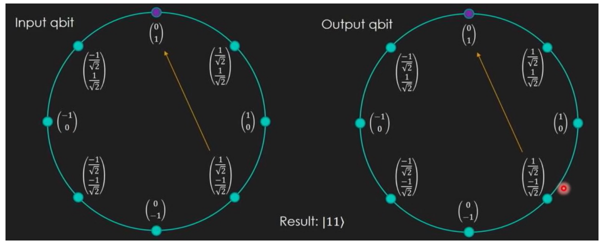
BB Identity (made with CNOT)

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \begin{bmatrix}1/2\\-1/2\\-1/2\\1/2\end{bmatrix} = \begin{bmatrix}1/2\\-1/2\\-1/2\\1/2\end{bmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \quad \boxed{H} = |01\rangle$$



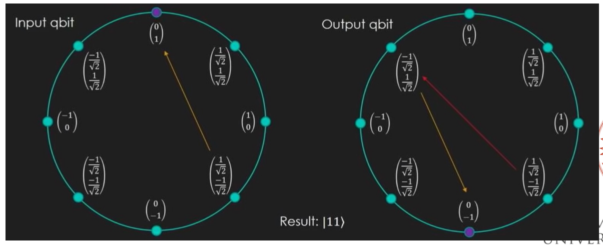
BB Negation

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} = \;.....\quad \boxed{H} = |11\rangle$$



BB Const-0

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} = \;....\quad \boxed{H} = |11\rangle$$



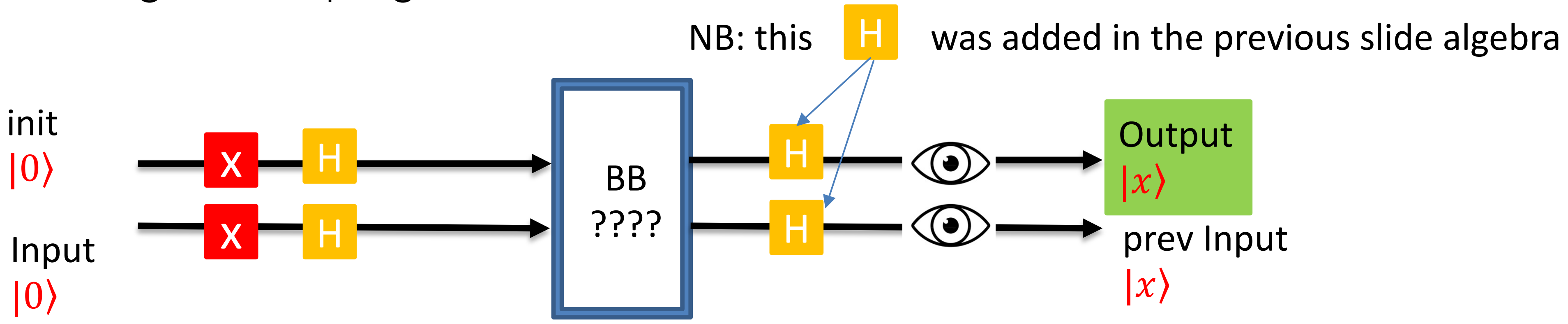
BB Const-1

note that there was also a bit flip here with respect to the line above (since negation is identity plus a final bitflip)

note that here in output we will have H (not shown algebraically), and then we read the Qbits indicated in red.

The Deutsch Oracle problem

Now, we observe that given this program on the central BlackBox:



If BB = constant function (Constant-0 or Constant-1) by providing Init and Input $|00\rangle$ we measure Output and Prev Input = $|11\rangle$

If BB = variable function (Identity or Negation) by providing Init and Input $|00\rangle$ we measure Output and Prev Input = $|01\rangle$

noteworthy result!

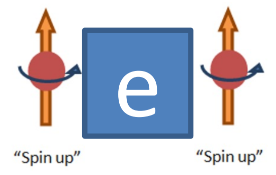
correct answer obtained by halving the execution steps compared to non-quantum computing

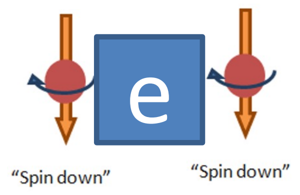
... and there is still much more...

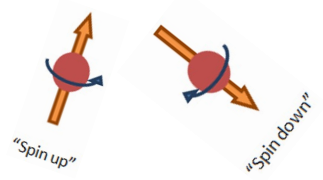
because you can exploit the **Qbit entanglement** $\boxed{e}$ as a new gate to do wonderful things.

I entangle 2 Qbits nearby here and now (without observing them)

The two Qbits will remain correlated



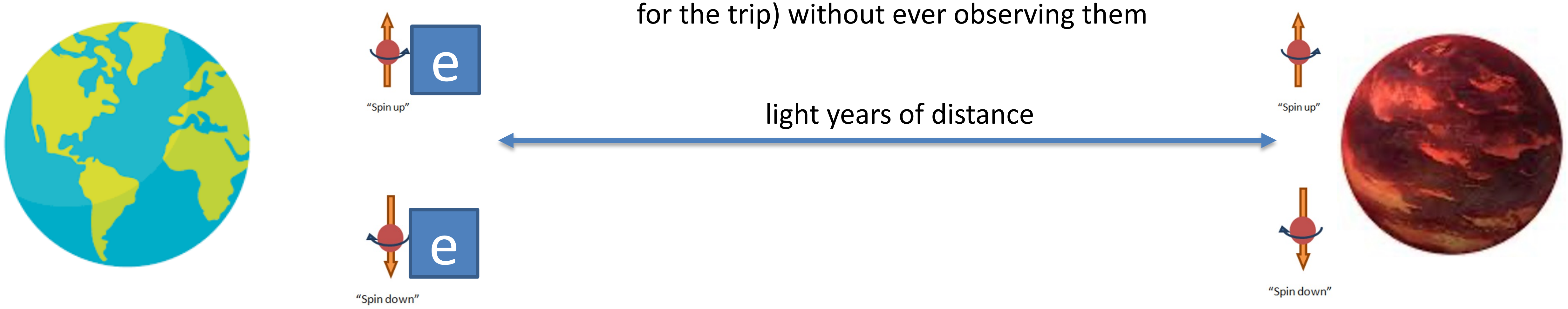




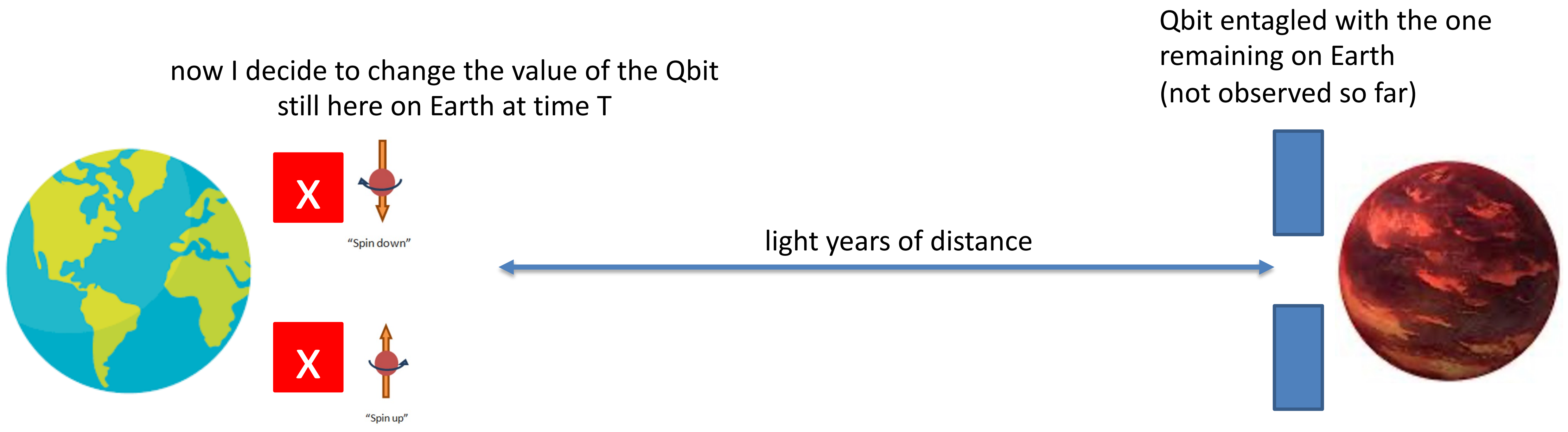
... and there is still much more...

because you can exploit the **Qbit entanglement**

transport one of the two Qbits over a huge distance (taking the time needed for the trip) without ever observing them

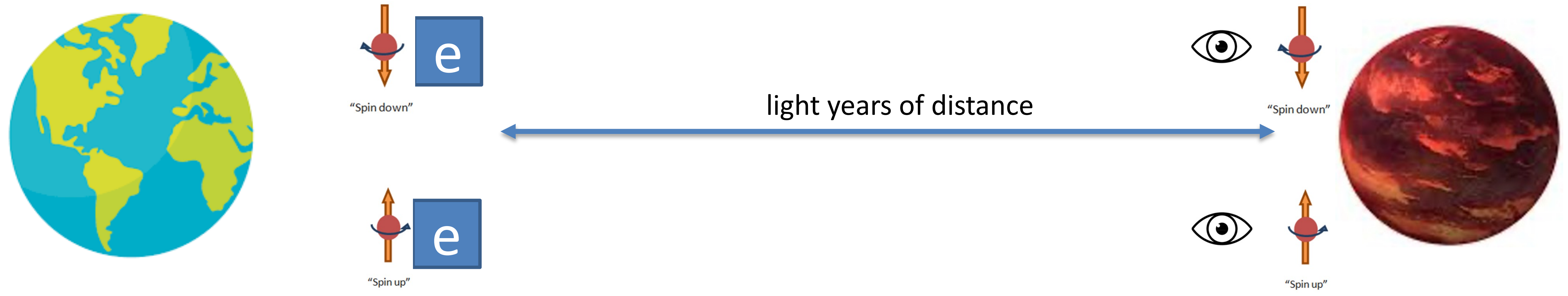


… and there is still much more…


now I decide to change the value of the Qbit
still here on Earth at time T
X
"Spin down"
X
"Spin up"
light years of distance
Qbit entagled with the one
remaining on Earth
(not observed so far)

... and there is still much more...

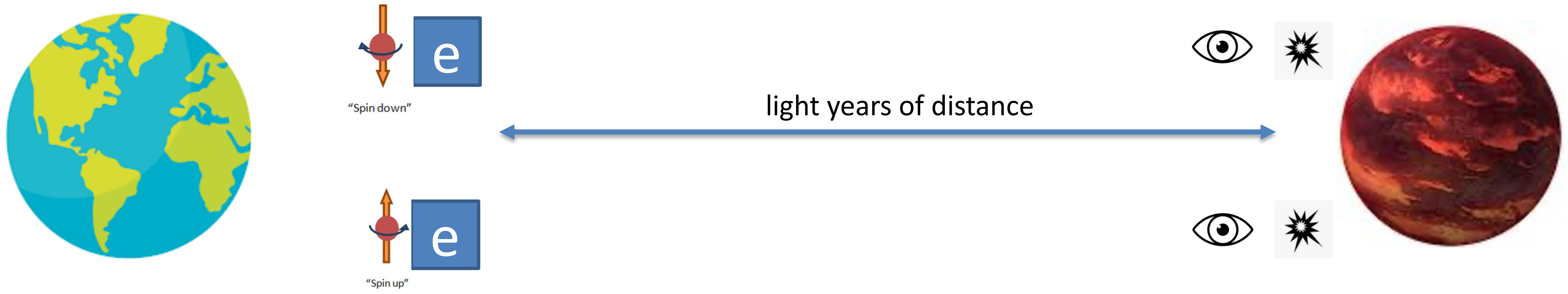
if from time T onwards I observe the distant Qbits, "instantly",
I will find the value correlated to the one of the modified Qbit on earth



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

... and there is still much more...

if from time T onwards I observe the distant Qbits, "instantly",
I will find the value correlated to the one of the modified Qbit on earth
but I will have consumed the Qbit!

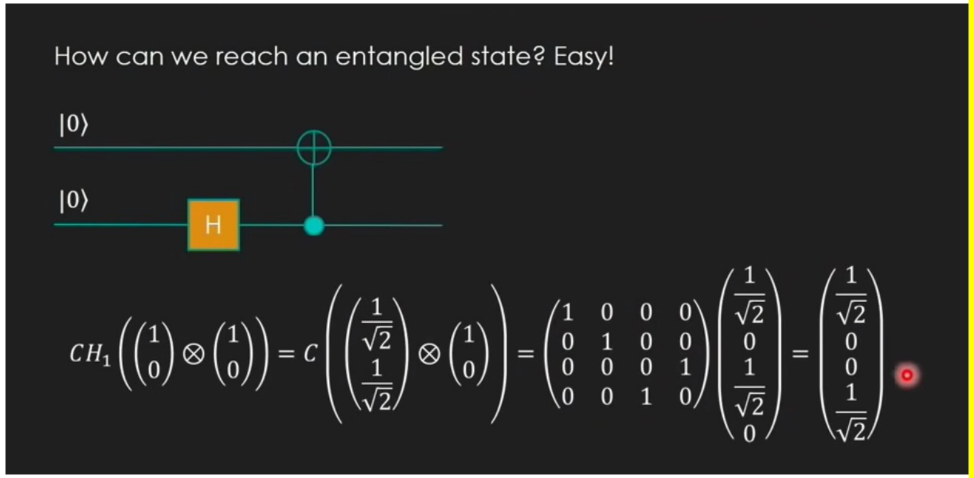


The entanglement power

Then can we also coordinate two systems at a huge distance in (almost) zero time?

Yes, thanks to entanglement!

But is it possible to make an entanglement port between Qbits? Yes, here it is! By using C and H



How can we reach an entangled state? Easy!

$$CH_1\left(\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\right) = C\left(\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\\frac{1}{\sqrt{2}}\\0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{pmatrix}$$

Entanglement circuit of two Qubits realized with one H gate and one CNOT. Note that by entangling two input Qbits $|0\rangle$ and $|0\rangle$ the H gate brings the first one s in superposition (s = 0 and 1 at 50%), leading to Qbits state $|s\ 0\rangle$ (the state vector contains two states at 50%: $|s\ 0\rangle$ or $|s\ 1\rangle$. This situation is given in input to a CNOT gate which generates the state vector in the figure (red dot). This shows only two possible cases: 50% $|0\ 0\rangle$ and 50% $|1\ 1\rangle$. This means that if I change the value of a Qbit, then the other entangled Qbit will change its value as well and it is not possible that two entangled Qbits will have different values... and this is almost instantaneously true on great distances with a speed which is much greater than the speed of light (observed sperimentally). Do not ask me why... ☺ (I would get a Nobel prize)

Will we have soon Quantum Co-Processors side by side to traditional ones? probable.

Will we have «quantum communication networks» able to synchronize systems in zero time? probable

Will we have secure «quantum communication networks» able to resist to attacks? sure

(NB: this does not mean we will have communication... think about how to realize digital transmission).

Conclusion

There are only $\binom{0}{1} \otimes \binom{1}{0}$ groups of Computer Scientists in the World

Which group would you think you belong to?

Conclusion

There are only $\binom{0}{1} \otimes \binom{1}{0}$ groups of Computer Scientists in the World

$\binom{0}{1} \otimes \binom{1}{0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ = state product shows 100% probability for value 2

1) those who do not understand Quantum Computing
2) those who erroneously think to have understood it.

Which group would you think you belong to?

Conclusion

There are only $\binom{0}{1} \otimes \binom{1}{0}$ groups of Computer Scientists in the World

$\binom{0}{1} \otimes \binom{1}{0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ = state product shows 100% probability for value 2

1) those who do not understand Quantum Computing

✅ 2) those who erroneously think to have understood it.

Which group would you think you belong to?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Conclusion

There are only $\binom{0}{1} \otimes \binom{1}{0}$ groups of Computer Scientists in the World

$\binom{0}{1} \otimes \binom{1}{0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ = robability for value 2

1) those wh um Computing

✅ 2) those w understood it.

Which group would you think you belong to?

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Luciano Bononi

Dipartimento di Informatica – Scienza e Ingegneria

luciano.bononi@unibo.it

https://www.unibo.it/sitoweb/luciano.bononi