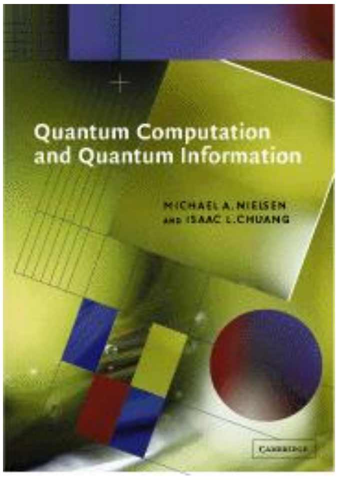
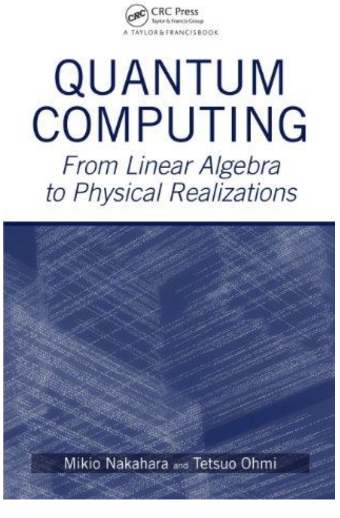
Quantum Computing
1 - Introduction

Angelo Bassi

Suggested textbooks



Quantum Computation and Quantum Information, by Michael Nielsen and Isaac Chuang.



Quantum Computing – From Linear algebra to Physical Realization, by Mikio Nakahara and Tetsuo Ohimi

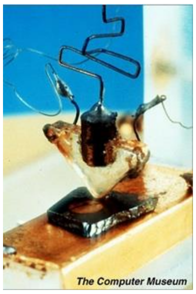
Lecture notes on Quantum Computing by Stefano Olivares:
https://sites.unimi.it/olivares/quantum-computing/

(especially for physical realization of quantum computers)

Introduction to Python:
https://github.com/mainaezio/TIF_2020_Introduction_to_Python

Snapshot of modern classical computers

1936: "On computable numbers, with an application to the Entscheidungsproblem", Alan Turing



1947: First transistor (Bell Labs)

1958: First integrated circuit

1975: Altair 8800, one of the first micro computers

1981: Osborne 1, first true mobile computer

1989: first Macintosh

Brief history of quantum computing

1980s: Richard Feynman

- Classical computers are very inefficient in simulating quantum systems (e^N)
- Computers are physical objects
- Why not creating computers following quantum laws?
- They will efficiently simulate at least themselves, maybe more, thus will be faster than any classical computer

Richard Feynman

On quantum physics and computer simulation

. . . there is plenty of room to make [computers] smaller. . . . nothing that I can see in the physical laws . . . says the computer elements cannot be made enormously smaller than they are now. In fact, there may be certain advantages.
—1959

Might I say immediately . . . we always have had a great deal of difficulty in understanding the world view that quantum mechanics represents. . . . I cannot define the real problem, therefore I suspect there's not a real problem, but I'm not sure there's no real problem.

I mentioned . . . the possibility . . . of things being affected not just by the past, but also by the future, and therefore that our probabilities are in some sense "illusory." We only have the information from the past and we try to predict the next step, but in reality it depends upon the near future . . .I'm trying to get . . . you people who think about computer-simulation possibilities to . . . digest . . . the real answers of quantum mechanics and see if you can't invent a different point of view than the physicists . . .
. . . the discovery of computers and the thinking about computers has turned out to be extremely useful in many branches of human reasoning. For instance, we never really understood how lousy our understanding of languages was, the theory of grammar and all that stuff, until we tried to make a computer which would be able to understand language . . . I . . . was hoping that the computer-type thinking would give us some new ideas . . .
. . . trying to find a computer simulation of physics seems to me to be an excellent program to follow out. . . . the real use of it would be with quantum mechanics. . . . Nature isn't classical . . . and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.
—1981

Feynman, R. 1959. There's Plenty of Room at the Bottom. Talk given at the annual meeting of the American Physical Society at Caltech. (Excerpt reprinted with permission from Caltech's *Engineering and Science*.)
———. 1981. Simulating Physics with Computers. Keynote address delivered at the MIT Physics of Computation Conference. Published in *Int. J. Theor. Phys.* **21** (6/7), 1982. (Excerpts reprinted with permission from the *International Journal of Theoretical Physics*.)

Brief history of quantum computing

1980: Paul Benioff describes the first QM model of computation

1985: David Deutsch describes first universal QC

1992: Deutsch-Jozsa algorithm

1993: Simon's algorithm

1994: Shor's algorithm

1995: Monroe & Wineland realize the first quantum gate (CNOT) with trapped ions

1996: Grover's algorithm

1998: First realization of a quantum algorithm (Deutsch-Jozsa), with NMR

1999: Nakamura and Tsai demonstrate superconducting qubits

2000: Fahri et al. propose Adiabatic Quantum Computation

2000: Raussendorf et al: One way (measurement based) quantum computing

2001: Shor's algorithm implemented to factorize 15

2014: Fahri et al. QAOA (Quantum Approximate Optimization Algorithm)

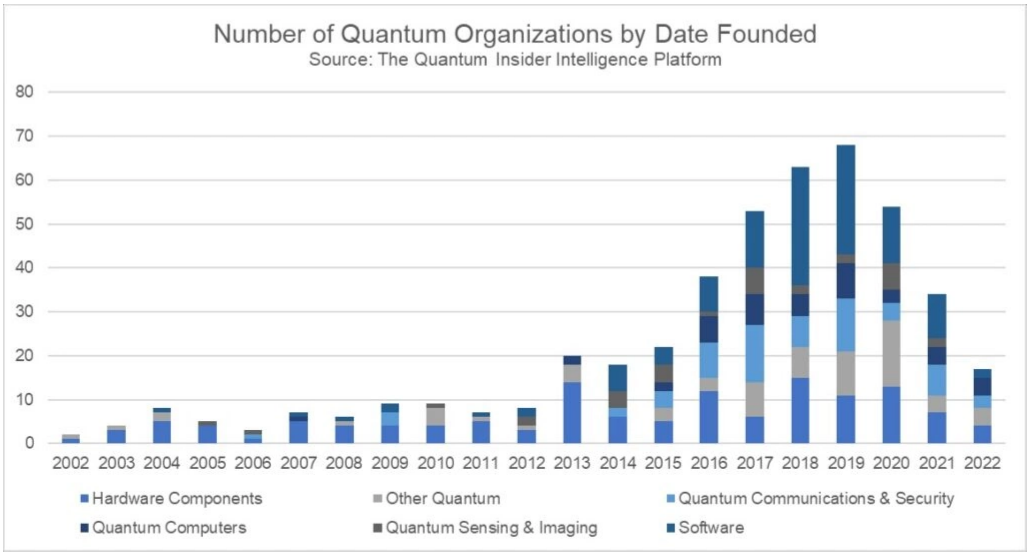
2016: IBM Quantum Experience

2019: Quantum supremacy by Google (?)

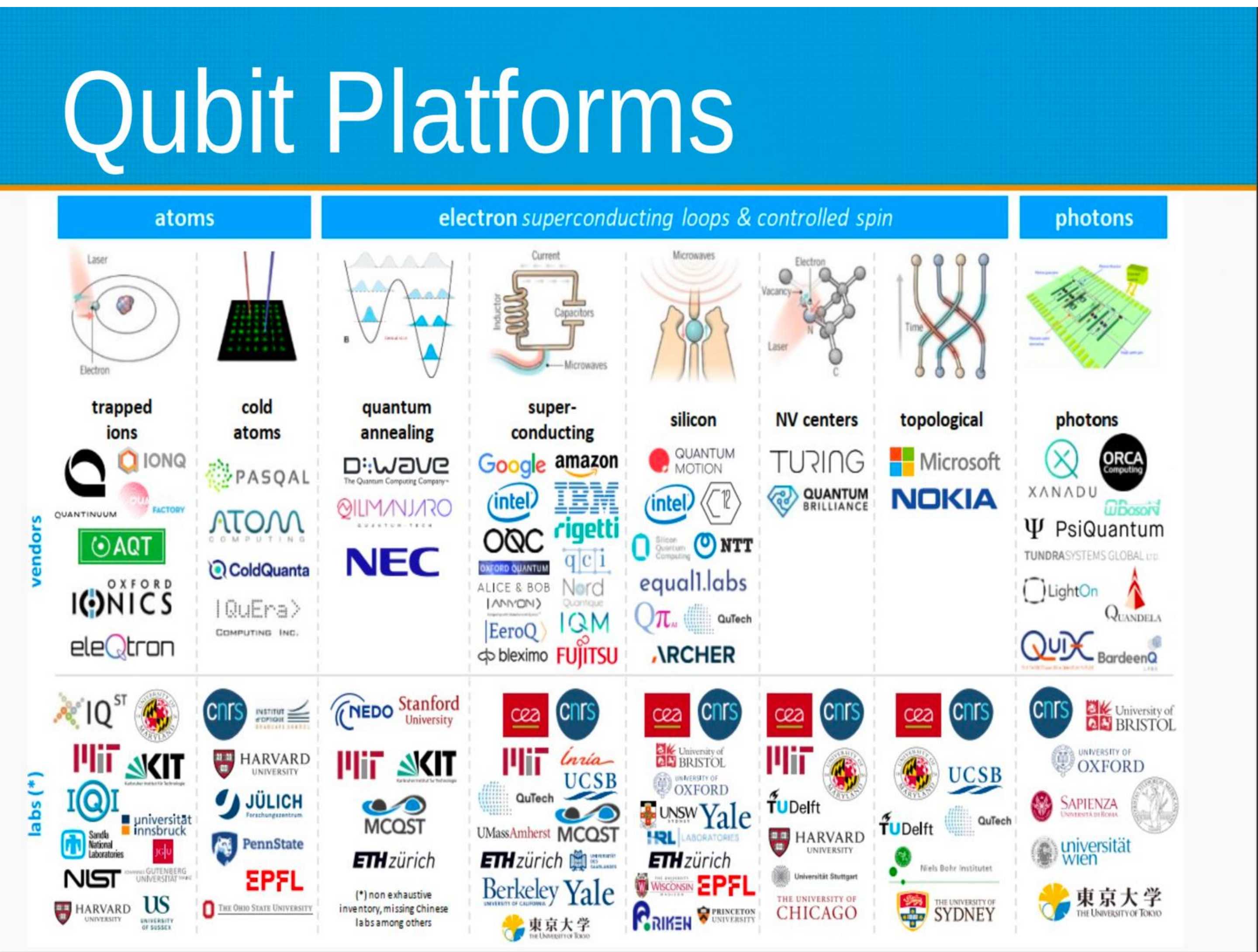
2023: First tests of error correcting schemes (Google)

(from "Timeline of Quantum Computing", Wikipedia)

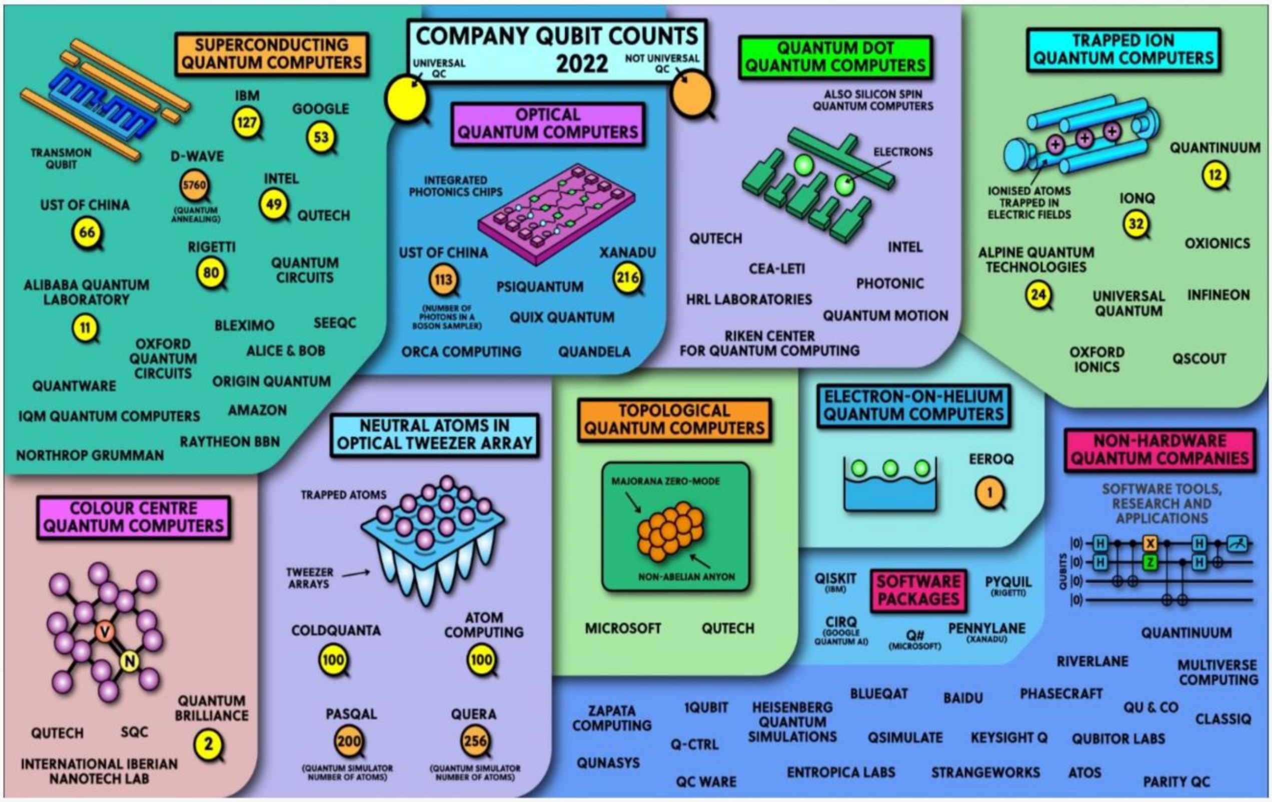
The Rise of Quantum Computing Companies



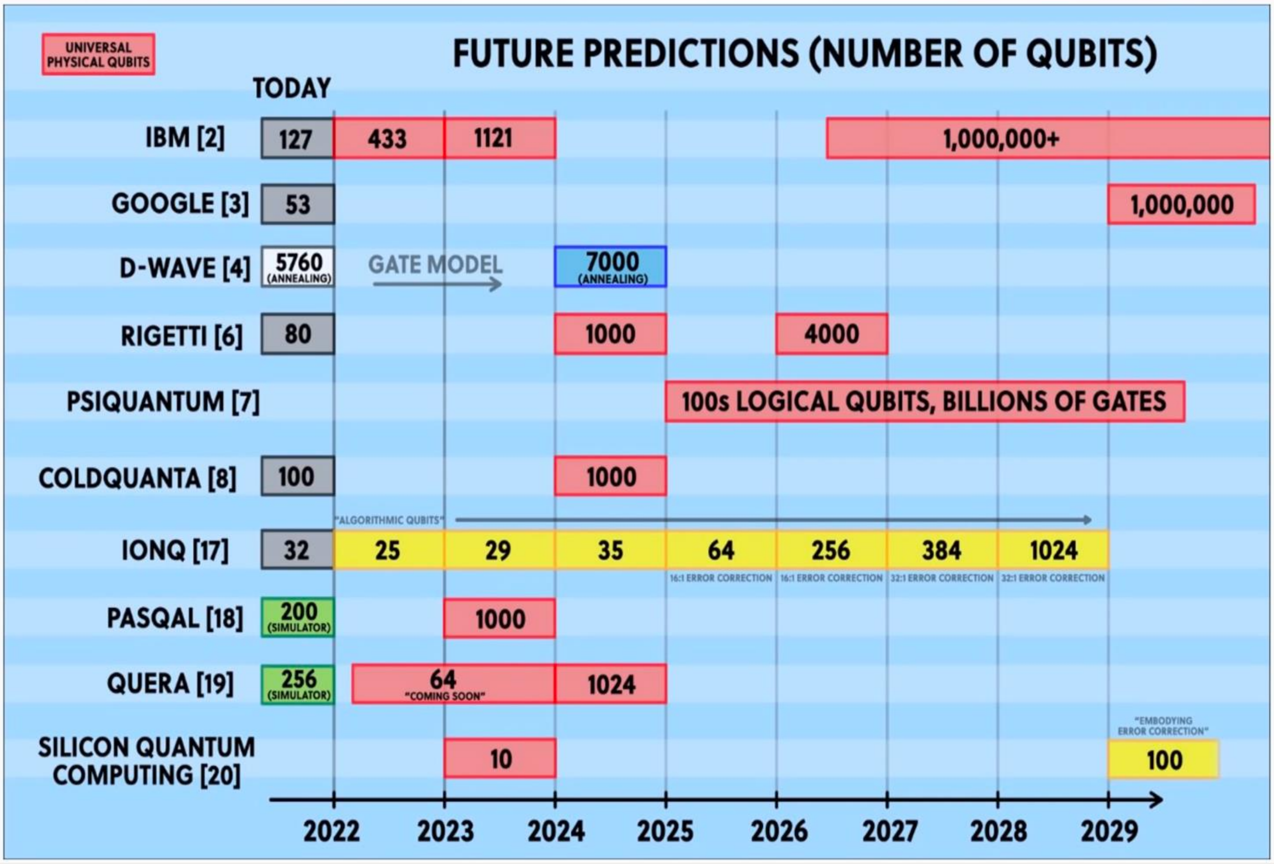
Source: The Quantum Insider Intelligence Platform



Qubit Counts (2021)



Qubit Count Predictions (2030)



Physical realization of Quantum Computers

Superconducting qubits: In superconductors, the basic charge carriers are pairs of electrons (known as Cooper pairs), rather than single fermions as found in typical conductors. These implement superconducting electronic circuits using superconducting qubits as artificial atoms; the two logic states are the ground state and the excited state. Superconducting quantum computing devices are typically designed in the radio-frequency spectrum, cooled in dilution refrigerators below 15 mK (millikelvins) and addressed with conventional electronic instruments, e.g. frequency synthesizers and spectrum analyzers.

The **largest number of qubits** is about **433** (IBM)

Companies:

1. IBM
2. Google
3. Intel
4. Rigetti

Trapped ions: the ions are suspended in free space using electromagnetic fields. Qubits are stored in stable electronic states of each ion, and quantum information can be transferred through the collective quantized motion of the ions in a shared trap (interacting through the Coulomb force). Lasers are applied to induce coupling between the qubit states (for single qubit operations) or coupling between the internal qubit states and the external motional states (for entanglement between qubits).

The largest number of particles to be controllably entangled is about **20-30 trapped ions**.

Companies:

1. Quantinuum (2021 – Cambridge UK)
2. IonQ (2015 – Maryland USA)
3. Quantum Factory (2018 – Munich DE)
4. Alpine Quantum Technologies (2018 – Austria AT)
5. Oxford Ionics (2019 – Begbroke UK)
6. EleQtron (2020 – Siegen DE)

Neutral atoms: the atoms are trapped in optical lattices, and manipulated with lasers. Qubits are encoded in the internal states. To turn on interactions between qubits, researchers target a pair of adjacent atoms with a laser pulse that excites one of them to a high-energy state called a Rydberg state, in which a valence electron orbits far from the nucleus. The Rydberg atom's strong electric dipole interactions prevent the laser from also exciting its neighbour, an effect known as a **Rydberg blockade**, but it's impossible to know which of the atoms was excited. The result is a single excitation shared between two qubits that can't be described separately—the characteristic feature of entanglement.

The largest number of particles to be controllably entangled is about **10 neutral atoms.** Overall they can control hundred of atoms.

Companies:

1. Pasqual (2019 – Paris Region FR)
2. Atom Computing (2018 – Berkeley USA)
3. ColdQuanta (2007 – Boulder USA)
4. Quera Computing (2018 – Boston USA)

Photons: It is a type of quantum computing that uses photons as a representation of qubits. The main advantages are simple components, the ability to run a variety of quantum operations, and most importantly, photonic quantum computers can perform at room temperature, which reduces the size of the extreme cooling systems.

Companies:

1. Xanandu Quantum Technologies (2016 – Canada)
2. ORCA Computing (2019 – London UK)
3. PsiQuantum (2015 – Silicon Valley USA)
4. TundraSystem Global (2014 – Cardiff UK)
5. Quandela (2017 – Paris FR)
6. QuiX Quantum (2019 – Enschede NL)

Cloud-based Quantum Computing

IBM Q Experience (superconducting qubits)

Xanadu (photonic quantum computer)

Forest by Rigetti Comuting (superconducting qubits)

Several simulators of quantum computers

Classical computation

Several models studied for the theory of classical computation

- Turing machines
- High-level programmable languages
- Boolean circuits

So far, the **Boolean circuit model** is by far the easiest model to generalize to quantum computation, being the closest to physical implementation. We will review it very briefly.

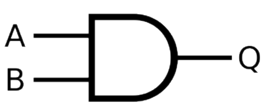
Boolean circuit model

Proposition: Any Boolean function

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

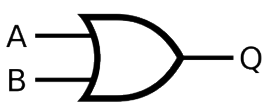
is computable by a Boolean circuit C using just AND, OR and NOT **gates** (in other words, AND, OR, NOT are **universal** for classical computation)

AND A∧B



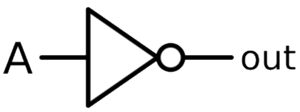
INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR A∨B



INPUT A	B	OUTPUT A + B
0	0	0
0	1	1
1	0	1
1	1	1

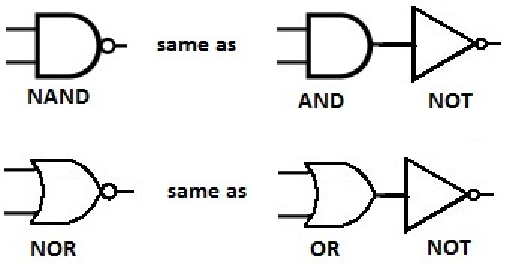
NOT ¬A



INPUT	OUTPUT
A	NOT A
0	1
1	0

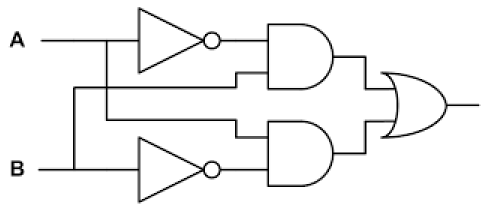
Example 1: NAND, NOR, XOR

Pronunciation: ex-or

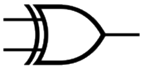


INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0



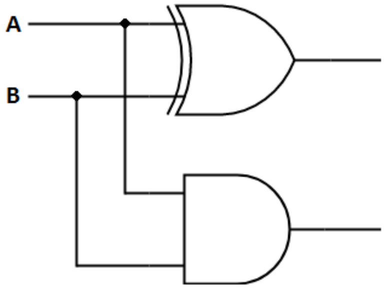
=



XOR $A \oplus B$

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example 2: Half adder



Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Note the **elements of a circuit**:

- Wires
- Gates
- Input on the left
- Output on the right

Size of a circuit = number of gates

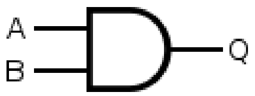
DUPE gate: duplicates bits

NAND is universal

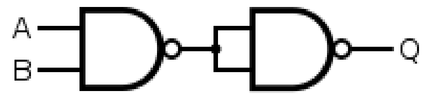
The number of fundamental gates can be reduced

Proposition: The NAND and DUPE gates are universal for computation

Desired AND Gate

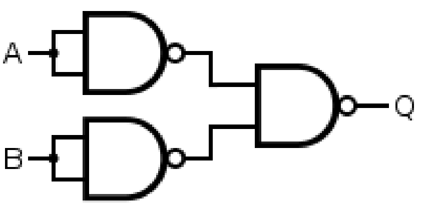


NAND Construction

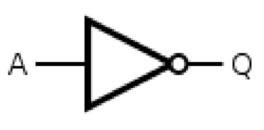


Desired OR Gate

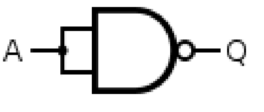
NAND Construction



Desired NOT Gate



NAND Construction



Reversible Computation

Logical gates are not always reversible:

- NOT is reversible
- AND is irreversible

INPUT	OUTPUT
A	NOT A
0	1
1	0

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

The laws of Physics are reversible, therefore if computation is implemented physically, it should be written in terms of reversible gates ➔ **Universal reversible computation** should be possible, there should exists a **universal set of reversible gates.**

This problem was studied in the '60s and '70s by Landauer e Bennett in connection with **thermodynamics**.

They were considering whether it is possible to have circuits made only of reversible gates, thus dissipating no energy. This was thought to be an important issue at that time. In fact now supercomputers needs **heavy cooling systems**. Yet it is not the most pressing one.

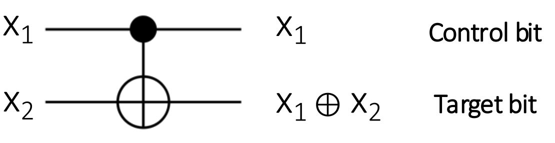
Reversible computation is important in the context of **quantum computation**, because – as we will see – quantum circuits need to be reversible in order to work properly.

Reversible gates - CNOT gate

Definition: A Boolean gate G is said to be reversible if it has the same number of inputs and outputs, and its mapping is bijective.

Some important new reversible gates

CNOT =

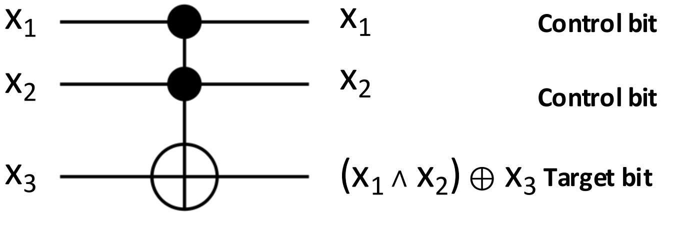


INPUT		OUTPUT	
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

If the control bit is 0, the target bit is left unchanged, otherwise it is flipped

CCNOT gate

CCNOT =



INPUT			OUTPUT		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

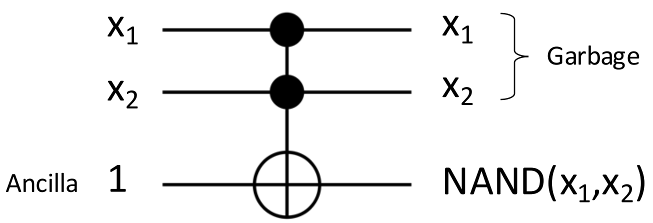
A NOT gate is applied to the target bit only if both control bits are 1, otherwise it is left unchanged. This is also called **Toffoli gate.**

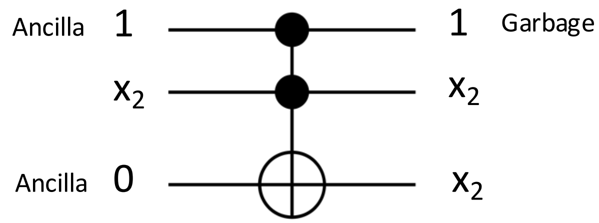
Comments:

- With the same logic, one can build the CCCNOT = C^3NOT gate and in general the **C^nNOT gate**.
- The CNOT and CCNOT are their own inverse. If applied twice, they give the **identity**. This is not always the case.

Universal reversible gates

CCNOT can be used to simulate NAND and DUPE

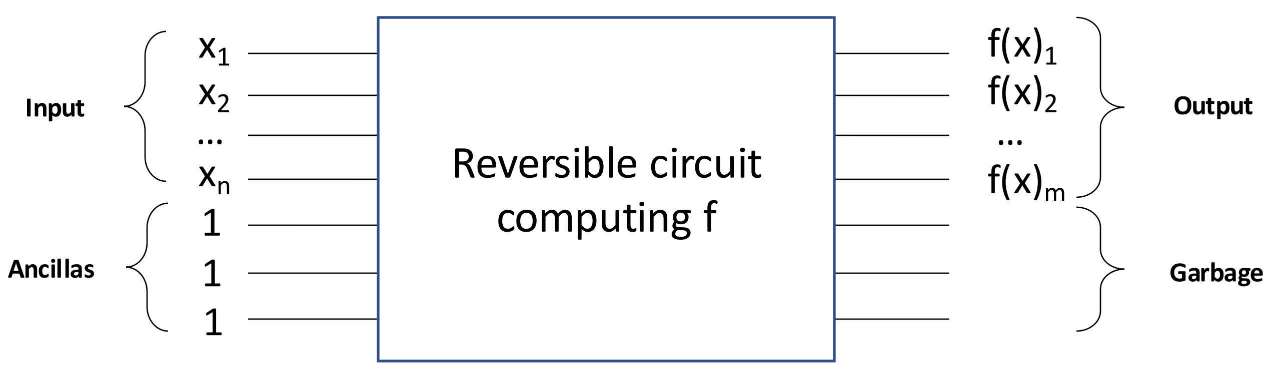




Theorem: The **CCNOT gate is universal**, assuming that ancilla inputs and garbage outputs are allowed. Any standard Boolen circuit can be **efficiently** transformed into a reversible circuit.

Reversible circuit

So far ancillas were sometimes 0 sometimes 1. They can be initialized to the same value, let's say 1, by means of a NOT gate. A reversible circuit computing f: $\{0,1\}^n \rightarrow \{0,1\}^m$ will then look as follows



The number of inputs and outputs is the same; the number of wires never changes. In fact, we can stop thinking about wires and think about each bit being carried in its own register, keeping its identity throughout the computation.

Probabilistic (randomized) computation

For finite probabilistic models, see

Van Kampen "Stochastic process in Physics and Chemistry", chapter V.2

https://arxiv.org/pdf/2209.14902 (section 4.3)

https://arxiv.org/pdf/1405.0303 (section 2)

We can open to the possibility that the value of a bit is not known with certainty

$$\text{0 or 1} \longrightarrow \begin{cases} 0 \text{ with probability } p_1 \\ 1 \text{ with probability } p_2 \end{cases}$$

deterministic bit   random bit

Note: the physics has not changed, we simply do not know the value of the bit.

The mathematical model changes, though. There are some computational tasks which we know how to provably solve efficiently using randomized computation (like generating prime numbers) but which we do not know how to provably solve efficiently using deterministic computation.

However there **should not** be any fundamental difference between the two models of computation, since they are based on the same physics.

We will introduce a **new notation** to deal with probabilistic computation, which will bring us a bit closer to quantum computation.

Basic notation	Vector notation	Abstract (Dirac) notation
0	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\vert 0>$
1	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\vert 1>$
0 with probability p_1 1 with probability p_2	$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$	$p_1 \vert 0> + p_2 \vert 1>$

Gates in the new notation: the NOT gate

In the new notation, **gates** are represented by **matrices**

$$\textbf{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Then

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

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

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} p_2 \\ p_1 \end{pmatrix}$$

For all other gates, we need to understand how to represent two and more bits.

Two (and more) random bits

With two bits, we have four possible states

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

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

Tensor product (we'll come back on this soon)

Two-bit gates: the AND gate

$$\textbf{AND} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note: it is not a square matrix, because the gate is not reversible

$$00 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0 \qquad 01 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$$

$$10 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0 \qquad 11 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

Two-bit gates: the CNOT gate

$$\textbf{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

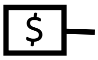
Note: it is a square matrix, because the gate is reversible

$$00 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 00 \qquad 01 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 01$$

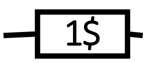
$$10 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = 11 \qquad 11 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 10$$

A truly probabilistic gate

We introduce two new gates

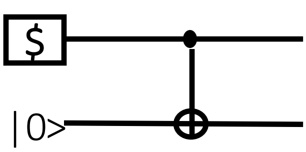
COIN =  $\rightarrow \begin{pmatrix} ½ \\ ½ \end{pmatrix}$

It has no input and a single bit output. It generates randomly either a 0 or a 1, with probability ½ each. It is like **fair coin tossing.**

1COIN =  $= \begin{pmatrix} 1 & ½ \\ 0 & ½ \end{pmatrix}$

If the input bit is 0, it is left unchanged. If it is 1, it is replaced by a COIN.

Example 1

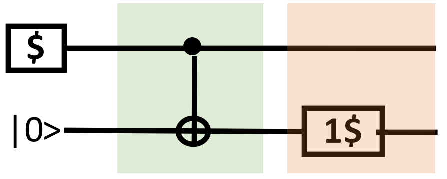


With probability ½ the input bit 00 and with probability ½ it is 10. In the first case the CNOT will leave in unchanged, in the second case it will changed into 11.

In mathematical terms

$$\begin{pmatrix} ½ \\ ½ \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ 0 \\ ½ \end{pmatrix} = ½ \underset{00}{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}} + ½ \underset{11}{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}$$

Example 2



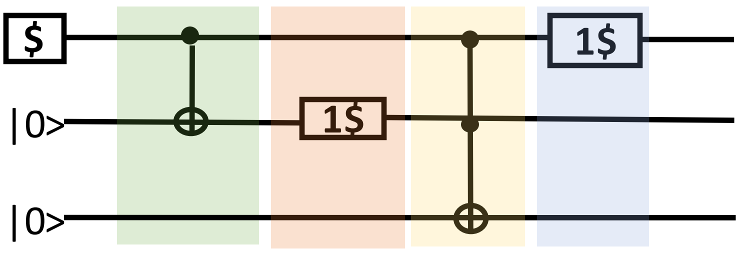
$$\begin{pmatrix}\frac{1}{2}\\ \frac{1}{2}\end{pmatrix} \otimes \begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{2}\\ 0\\ \frac{1}{2}\\ 0\end{pmatrix} \rightarrow \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{2}\\ 0\\ \frac{1}{2}\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{2}\\ 0\\ 0\\ \frac{1}{2}\end{pmatrix} \rightarrow \underbrace{\begin{pmatrix}1 & \frac{1}{2} & 0 & 0\\ 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & 1 & \frac{1}{2}\\ 0 & 0 & 0 & \frac{1}{2}\end{pmatrix}}_{\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \otimes \begin{pmatrix}1 & \frac{1}{2}\\ 0 & \frac{1}{2}\end{pmatrix}}\begin{pmatrix}\frac{1}{2}\\ 0\\ 0\\ \frac{1}{2}\end{pmatrix} = \begin{pmatrix}\frac{1}{2}\\ 0\\ \frac{1}{4}\\ \frac{1}{4}\end{pmatrix}$$

Using the Dirac notation ($|0\rangle \otimes |0\rangle = |00\rangle$, and same for others)

$$\frac{1}{2}\,|00\rangle + \frac{1}{2}\,|10\rangle \rightarrow \frac{1}{2}\,|00\rangle + \frac{1}{2}\,|11\rangle \rightarrow \frac{1}{2}\,|00\rangle + \frac{1}{2}\left(\frac{1}{2}\,|10\rangle + \frac{1}{2}\,|11\rangle\right)$$

$$= \frac{1}{2}\,|00\rangle + \frac{1}{4}\,|10\rangle + \frac{1}{4}\,|11\rangle = \begin{pmatrix}\frac{1}{2}\\ 0\\ \frac{1}{4}\\ \frac{1}{4}\end{pmatrix}$$

Example 3



$\frac{1}{2}\,|000\rangle + \frac{1}{2}\,|100\rangle \rightarrow \frac{1}{2}\,|000\rangle + \frac{1}{2}\,|110\rangle$

$\rightarrow \frac{1}{2}\,|000\rangle + \frac{1}{2}\,(\frac{1}{2}\,|100\rangle + \frac{1}{2}\,|110\rangle) = \frac{1}{2}\,|000\rangle + \frac{1}{4}\,|100\rangle + \frac{1}{4}\,|110\rangle$

$\rightarrow \frac{1}{2}\,|000\rangle + \frac{1}{4}\,|100\rangle + \frac{1}{4}\,|111\rangle$

$\rightarrow \frac{1}{2}\,|000\rangle + \frac{1}{4}\,(\frac{1}{2}\,|000\rangle + \frac{1}{2}\,|100\rangle) + \frac{1}{4}\,(\frac{1}{2}\,|011\rangle + \frac{1}{2}\,|111\rangle)$

$= \frac{5}{8}\,|000\rangle + \frac{1}{8}\,|100\rangle + \frac{1}{8}\,|011\rangle + \frac{1}{8}\,|111\rangle$

Comment 1

We used the formalism of **linear algebra** for probabilistic computation because “**ignorance propagates linearly**”.

If a physical system is either in state x with probability p or in state y with probability q, and x evolves into X and y into Y, then at the end the system will be in state X with probability p or in state Y with probability q. In Dirac notation:

$p|x\rangle + q|y\rangle \rightarrow p|X\rangle + q|Y\rangle = p\,T[|x\rangle] + q\,T[|y\rangle] = T[p|x\rangle + q|y\rangle]$

The evolution **operator T is linear**, and can be represented by a matrix.

Comment 2

Measurements simply reveal the true state of the system, which was unknown to us before the measurement. **After the measurement**, the information about **the state of the system changes**, and with it the probability distribution. With reference to the previous example

1. We measure the three bits and find 000:

$^5/_8$ |000> + $^1/_8$ |100> + $^1/_8$ |011> + $^1/_8$ |111> ➜ |000>

This happens with probability $^5/_8$

2. We measure the first bit and find 0; this happens with probability $^5/_8$ + $^1/_8$ = $^3/_4$

$^5/_8$ |000> + $^1/_8$ |100> + $^1/_8$ |011> + $^1/_8$ |111> ➜ $\dfrac{^5/_8 |000> + ^1/_8\ |011>}{^3/_4}$

= $^5/_6$ |000> + $^1/_6$ |011>

We can call it "**collapse**" of the probability. It is not a real physical phenomenon. It is **Bayes rule**: P(A|B) = P(B|A) P(A) / P(B). In our case:

P(|000>|"0") = P("0"||000>) P(|000>) / P("0") = 1 × $^5/_8$ ÷ $^3/_4$ = $^5/_6$

Rules of probabilistic classical computation

1. The **state** of a single probabilistic bit is given by a vector in R^2, or in Dirac notation:

$|x\rangle = p|0\rangle + q|1\rangle$, with $p,q \in \mathbb{R}$, and $p+q=1$.

The **coefficients** give the **probabilities** for the bit to have that value.

States for **multiple bits** are constructed via **tensor product** of R^2

Two bits: $|xy\rangle = |x\rangle \otimes |y\rangle$

Three bits: $|xyz\rangle = |x\rangle \otimes |y\rangle \otimes |z\rangle$, and so on

Why tensor products, and not – for example – Cartesian product?

Take for example three bits. There are 8 possible configurations: 000, 001, 010, 011, 100, 101, 110, 111. The register can be in any of these 8 states, and the information propagates linearly (without interference among the states), therefore they behave like **linearly independent states**.

This means that one needs 8 basis states in the vector space, which is what is provided by the tensor product, not by the Cartesian product.

Rules of probabilistic classical computation

2. **Gates** are implemented by **linear operators**, i.e. matrices.

Gates can be either reversible (square invertible matrices) or irreversible (for example rectangular matrices).

As we saw that computation can always be made reversible, without loss of generality we can say that gates are implemented by linear invertible operators (**NxN invertible stochastic matrices**).

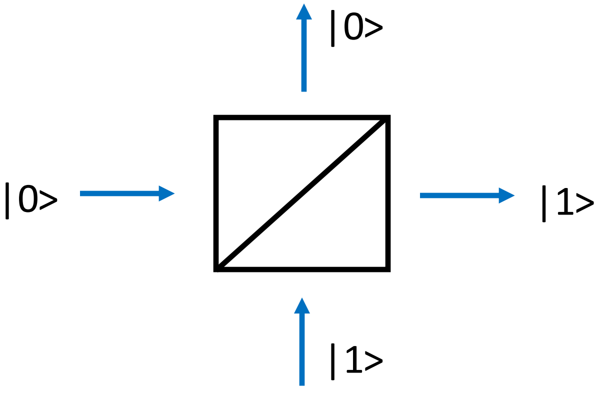
Of course, they have to preserve probabilities.

3. **Measurements** are updates of information. The states changes according to Bayes rule ("collapse" of the state)

As we will see, the rules of quantum computation are almost similar, but with fundamental differences.

Preview of Quantum Computation

Beam splitters (BS) are optical devices, which split the path of a photon in two: once a photon has entered, there is ½ probability that it goes one way, and ½ probability that it goes the other way. It is a **probabilistic gate.**



If we associate the value of the bit to the path of the photon (instead of the voltage as in standard computers), then we have

$$|0\rangle \rightarrow \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle$$

$$|1\rangle \rightarrow \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle$$

Then

$$-\boxed{\mathbf{BS}}- \;=\; -\boxed{\$}- \qquad |x\rangle \rightarrow \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle$$

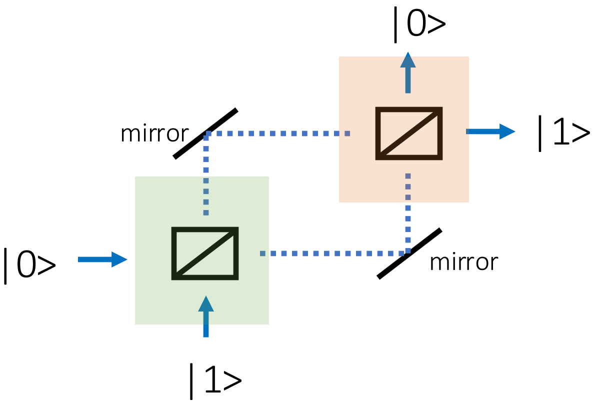
Whatever the input state, it generates an equal weighted distributions of 0 and 1. The matrix representation is:

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

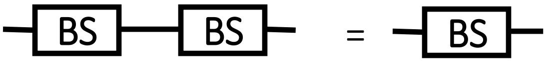
In fact: $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ since $p+q=1$

Preview of Quantum Computation

But now we can do the following optical construction:



This is equivalent to the following circuit

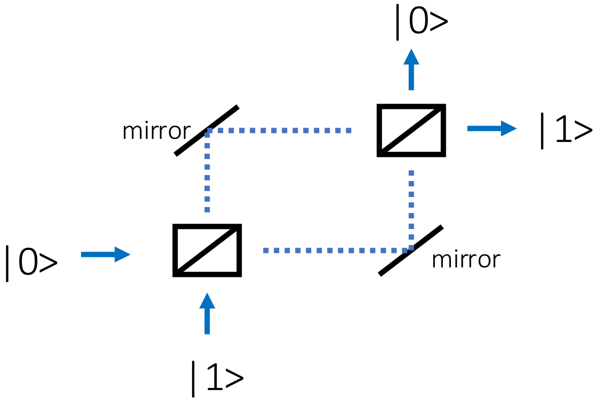


Since

$$\begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix} \begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix} = \begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix}$$

In a classical picture (coin tossing), this makes perfectly sense

But this is not what happens. What happens it:



$|0\rangle \rightarrow |0\rangle$

$|1\rangle \rightarrow |1\rangle$

How is this possible? The answer is that photons are quantum: they cannot be thought as particles which follow **one path or the other**. They are more like waves, which split in two, **interfere** and then recombine

Preview of Quantum Computation

We will see how this is described by quantum mechanics, but the essence is the following: how can we **destroy probabilities**?

We have to justify

$$|0\rangle \xrightarrow[\text{first BS}]{} \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle \xrightarrow[\text{second BS}]{} |0\rangle$$

Instead of

$$|0\rangle \xrightarrow[\text{first BS}]{} \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle \xrightarrow[\text{second BS}]{} \tfrac{1}{2}\,|0\rangle + \tfrac{1}{2}\,|1\rangle$$

We destroy probabilities with negative (in general, complex) numbers. But what does it mean to have negative probabilities? The solution of QM is:

Bit $\rightarrow$ $p|0\rangle + q|1\rangle$ with $\underbrace{p,q}_{\text{probabilities}} \in \mathbb{R}^+$ and $p+q=1$

changed into

Qubit $\rightarrow$ $a|0\rangle + b|1\rangle$ with $\underbrace{a,b}_{\text{amplitudes}} \in \mathbb{C}$ and $\underbrace{|a|^2 + |b|^2}_{\text{Probabilities (they remain always positive)}} = 1$

Preview of Quantum Computation

The BS is mathematically described by

$-\boxed{\text{BS}}- \;=\; -\boxed{\text{H}}-$ Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

!!!

Then

$-\boxed{\text{H}}-$

$$\left.\begin{aligned} |0\rangle &\rightarrow {}^{1}/_{\sqrt{2}}\,|0\rangle + {}^{1}/_{\sqrt{2}}\,|1\rangle \\ |1\rangle &\rightarrow {}^{1}/_{\sqrt{2}}\,|0\rangle - {}^{1}/_{\sqrt{2}}\,|1\rangle \end{aligned}\right\}$$

In both cases, probabilities are 50% of getting the value 0 or 1

But now

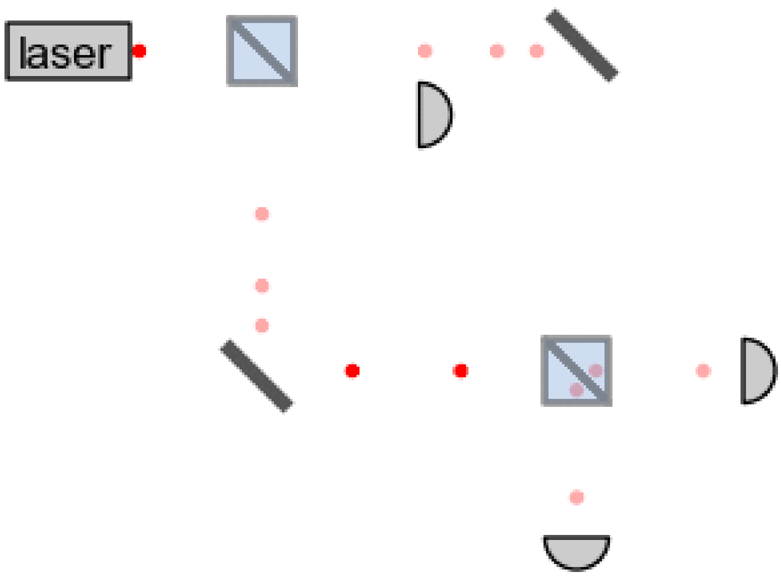
$-\boxed{\text{BS}}-\boxed{\text{BS}}- \;=\; \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

After the second BS, the bit takes the initial value

What happens physically is that the **photon** behaves like a **wave**. There can be **constructive interference**, which mathematically is expressed by amplitudes **adding**, and **destructive** interference, which mathematically is expressed by amplitudes **subtracting**. This is the role of negative numbers.

Preview of Quantum Computation

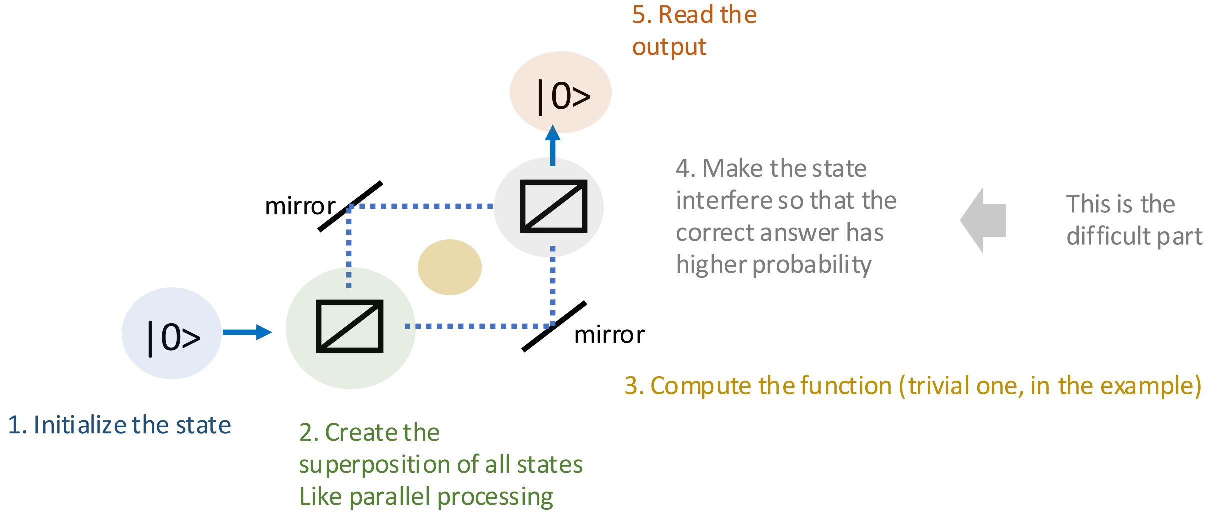
The surprising thing is that if we measure the photon right after the first BS and before it enters the second one, we will not find it half here and half there, as it would happen with classical waves. It will always be **either here or there**, and the wave behaviour is destroyed.



Understanding what this means brings into the **foundations of quantum mechanics**, which is beyond the scope of the present course.

Quantum Computation

The essence of a quantum computation is the following



The art of quantum computing is to make the different terms of the superposition interfere in such a way to maximize the correct answer, in a number of steps which is smaller than for any classical algorithm.