

Advanced Quantum Mechanics

Angelo Bassi

Academic Year 2022-23

Advanced Quantum Mechanics

Advanced Quantum Mechanics

9 CFU

New Frontiers in Quantum Mechanics
(Physics)

=

Introduction to Quantum Mechanics
and Quantum computing
(Data science)

6 CFU

Quantum Mechanics and
Special Relativity

3 CFU

Program

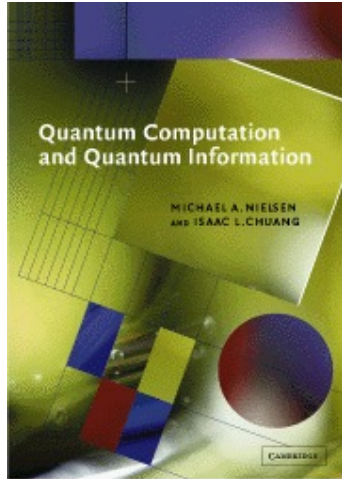
1 – **New Frontiers = Intro QM and QC (6 CFU)**

- Brief review of QM and its working rules.
- QC: The qubit and quantum gates
- QC: Some relevant quantum circuits
- Seminars by IBM
- Open quantum systems and decoherence

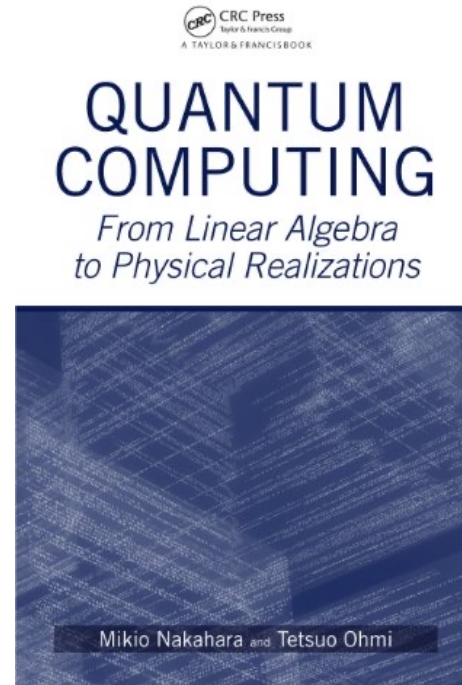
2 – **Quantum Mechanics and Relativity (3 CFU)**

- Review of special relativity with some exercises
- Quantum nonlocality, teleportation, no cloning, cryptography...

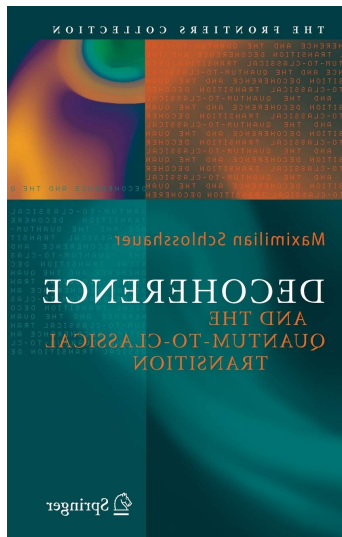
Textbook for the first part



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



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



Decoherence: And the Quantum-To-Classical Transition, by Maximilian A. Schlosshauer

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

Brief history of quantum computing

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

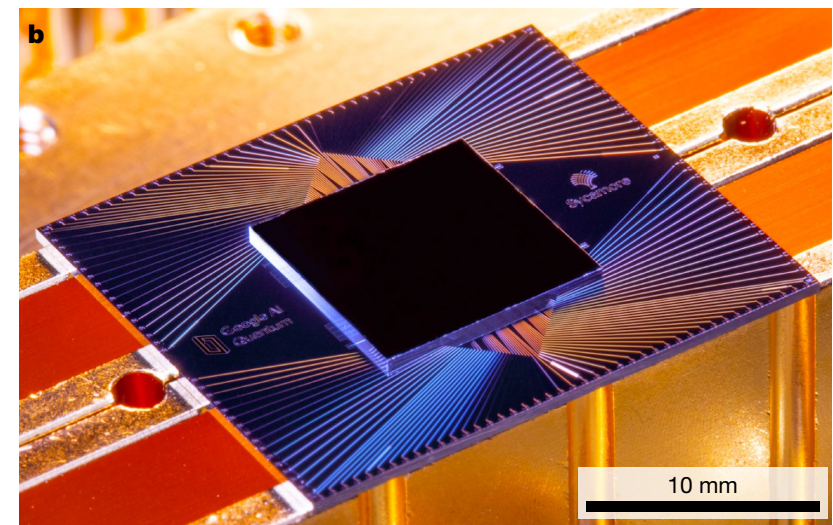
2014: Fahri et al. QAOA (Quantum Approximate Optimization Algorithm)

2016: IBM Quantum Experience

2019: Quantum supremacy by Google (?)

(from "Timeline of Quantum Computing",
Wikipedia)

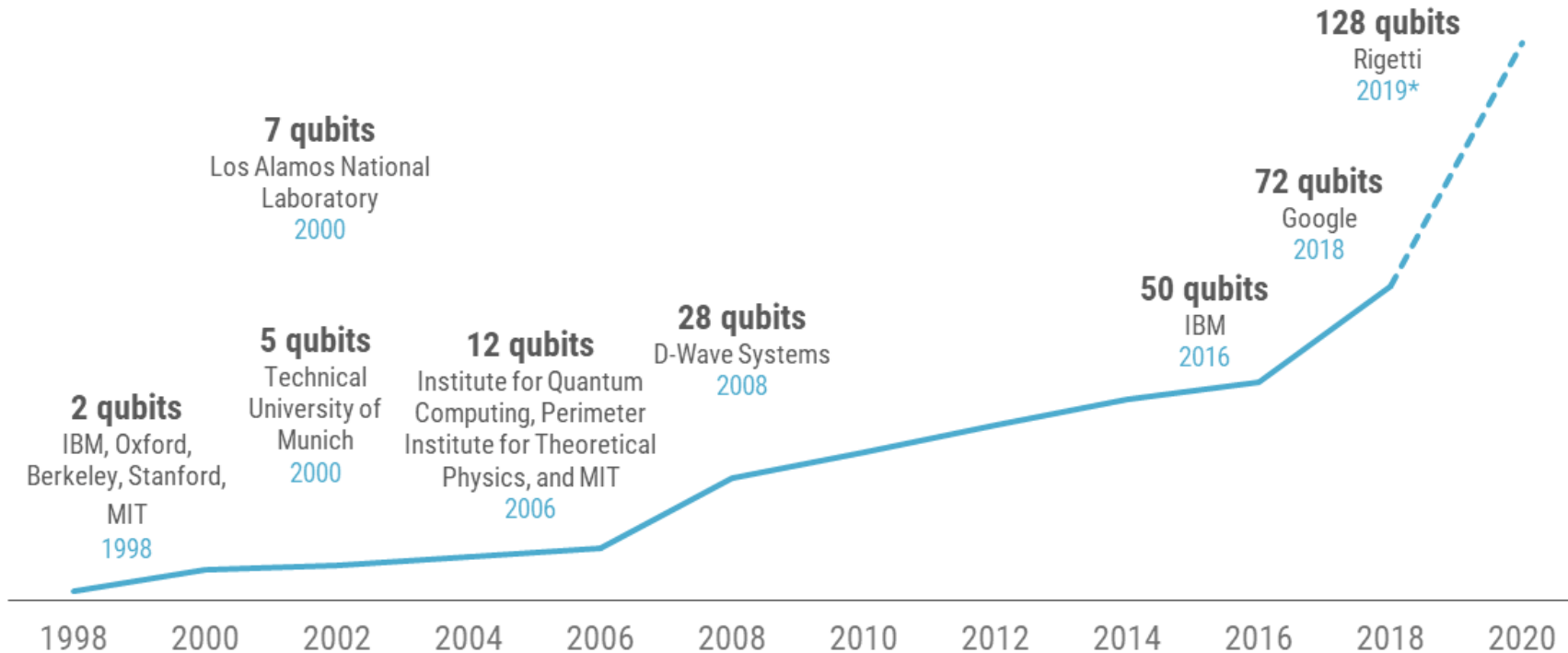
Sycamore chip used by Google



Scaling of qubits

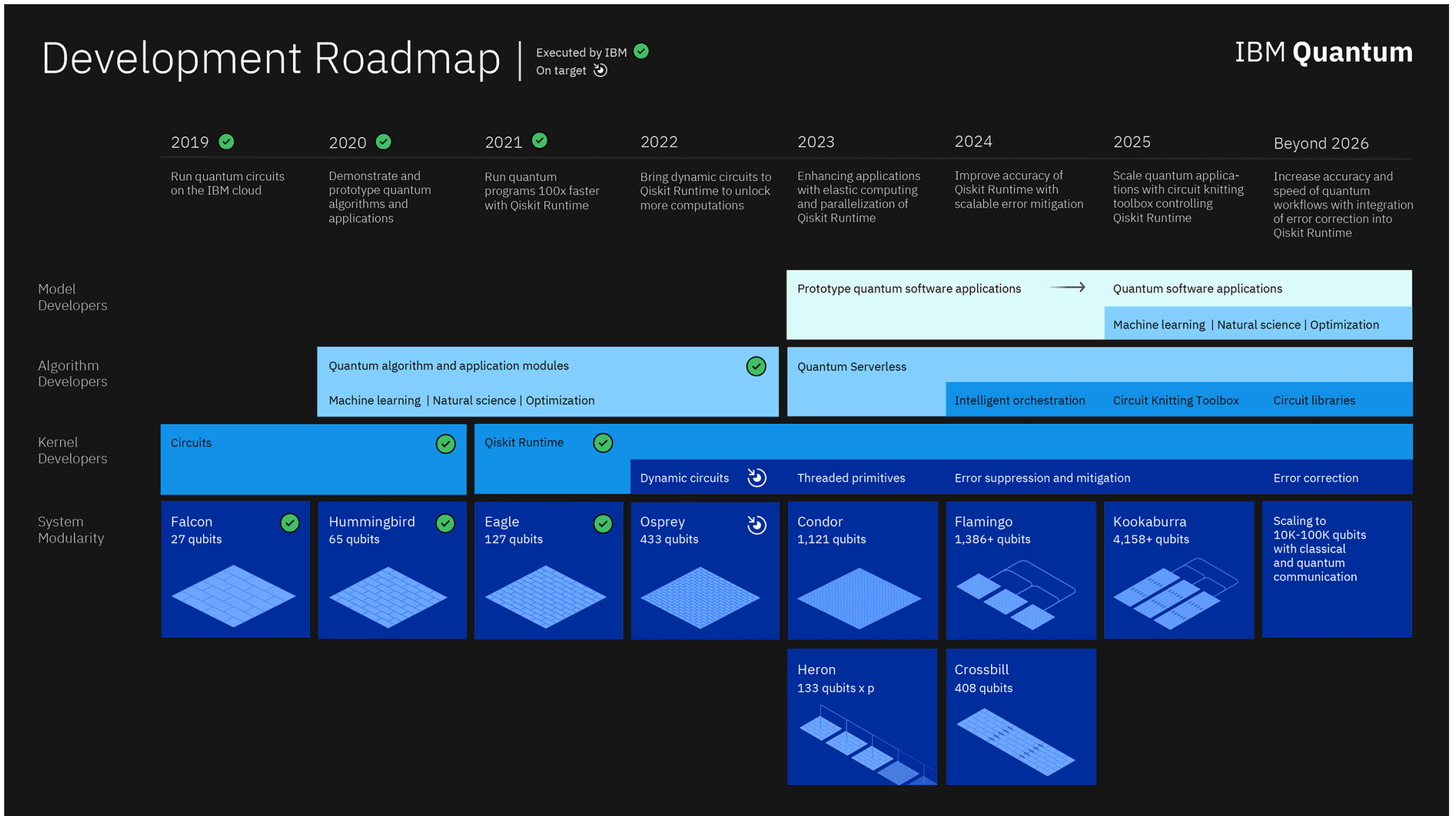
Quantum computers are getting more powerful

Number of qubits achieved by date and organization 1998 – 2020*



Source: MIT, Qubit Counter. *Rigetti quantum computer expected by late 2019.

Scaling of qubits



Physical realization of Quantum Computers

Superconducting qubits: superconducting circuits.

Trapped ions: internal states of trapped ions.

Neutral atoms: internal states of atoms in optical lattices.

Quantum dots: trapped electrons in materials

Photons: quantum light manipulated

Cloud-based Quantum Computing

IBM Q Experience (superconducting qubits)

Xanadu (photonic quantum computer)

Forest by Rigetti Computing (superconducting qubits)

Several simulators of quantum computers

Classical computation

Several models studied for the theory of classical computation

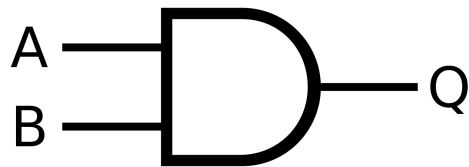
- Turing machines
- High-level programmable languages
- Boolean circuits

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

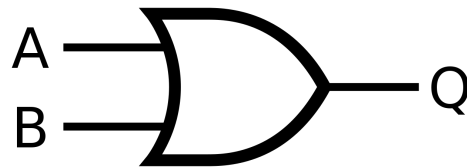
Proposition: Any Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ is computable by a Boolean circuit C using just AND, OR and NOT gates (in other words, AND, OR, NOT are universal for classical computation)

AND $A \wedge B$



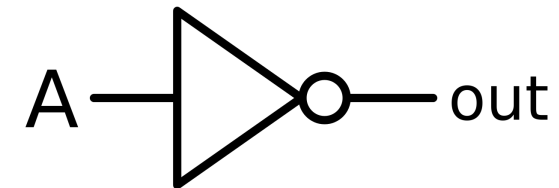
INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR $A \vee B$



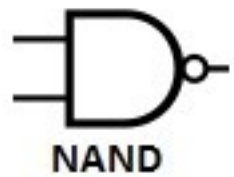
INPUT		OUTPUT
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

NOT $\neg A$

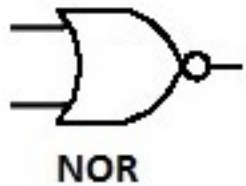
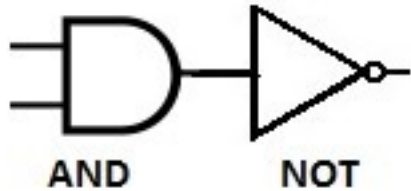


INPUT	OUTPUT
A	NOT A
0	1
1	0

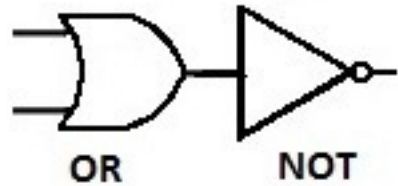
Example 1: NAND, NOR, XOR



same as

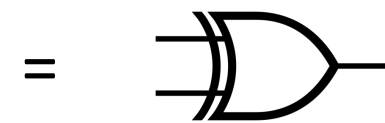
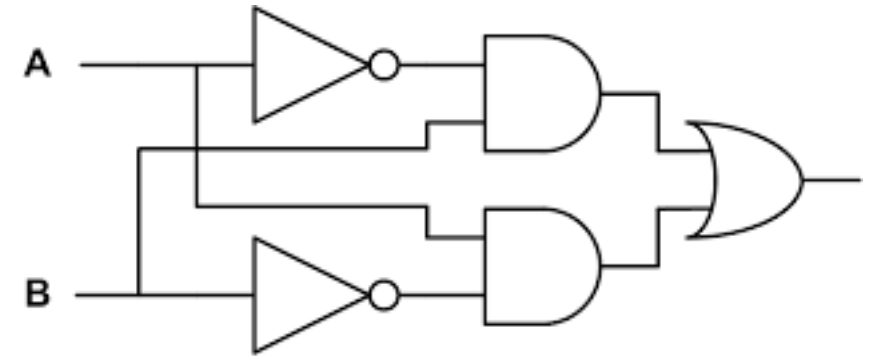


same as



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

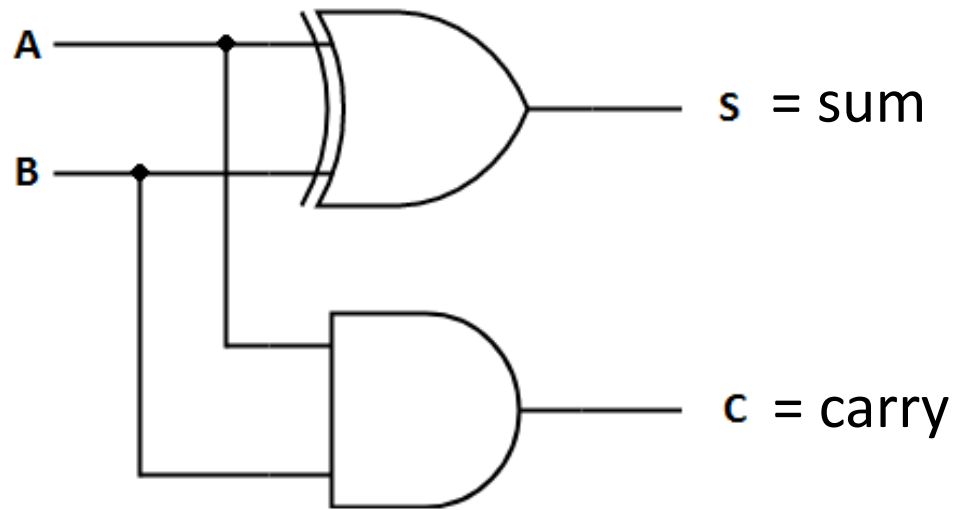
INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0



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

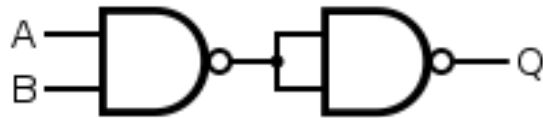
The number of fundamental gates can be reduced

Proposition: The NAND and DUPE gates are universal for computation

Desired AND Gate



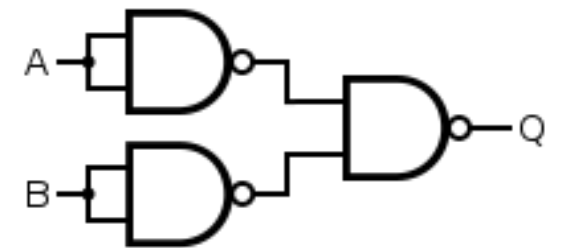
NAND Construction



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 exist a **universal set of reversible gates**.

Reversible Computation

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

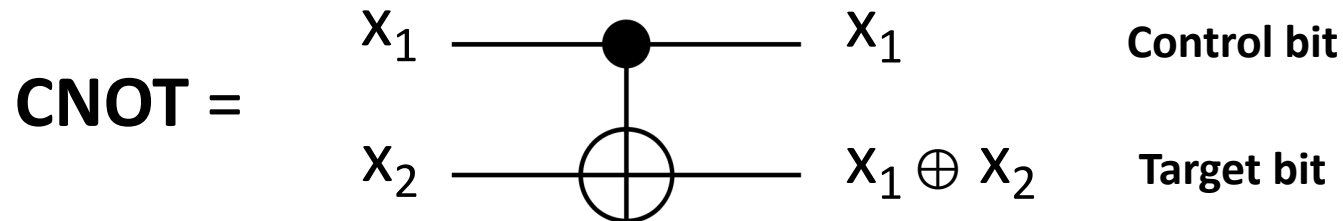
They were considering whether it is possible to have circuits made only of reversible gates, thus dissipating no energy. This was thought to be an important issue at that time. In fact now supercomputers needs **heavy cooling systems**. Yet it is not the most pressing one.

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.

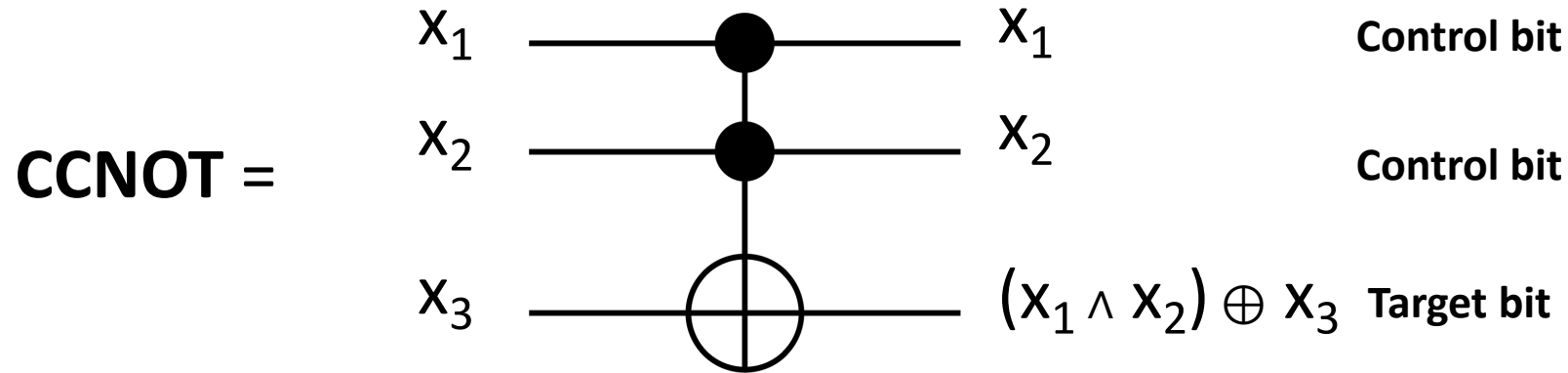
Some important new reversible gates



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



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**.

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

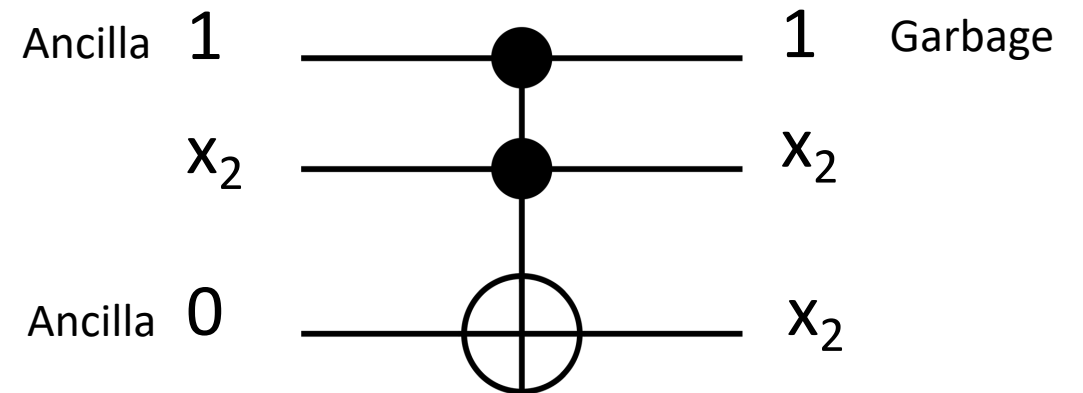
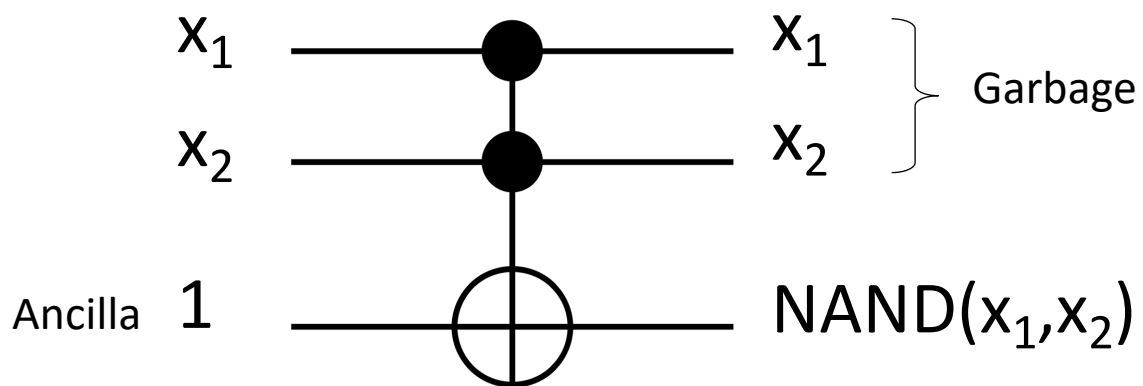
C^n NOT gate

Comments:

- With the same logic, one can build the CCCNOT = C^3 NOT gate and in general the **C^n NOT 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

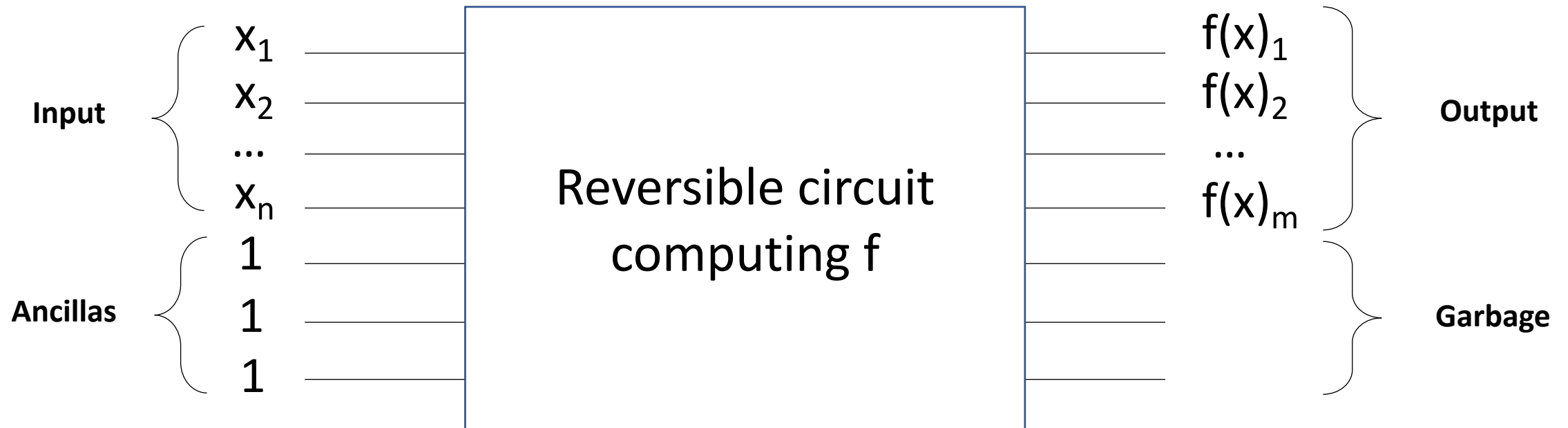
CCNOT can be used to simulate NAND and DUPE



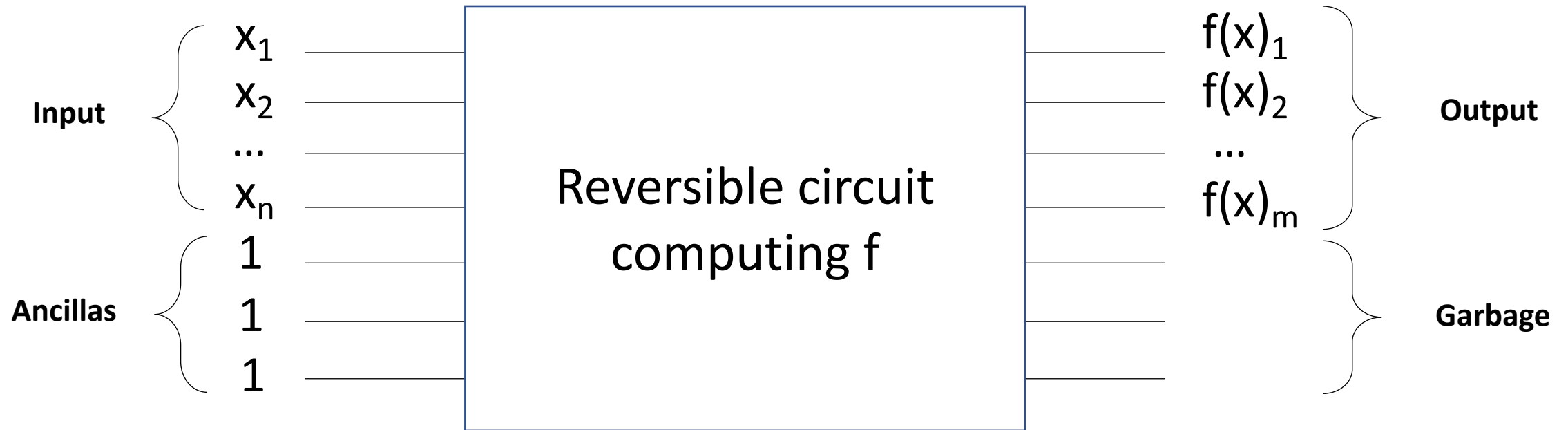
Theorem: The **CCNOT gate is universal**, assuming that ancilla inputs and garbage outputs are allowed. Any standard Boolean circuit can be **efficiently** transformed into a reversible circuit.

Universal reversible gates

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



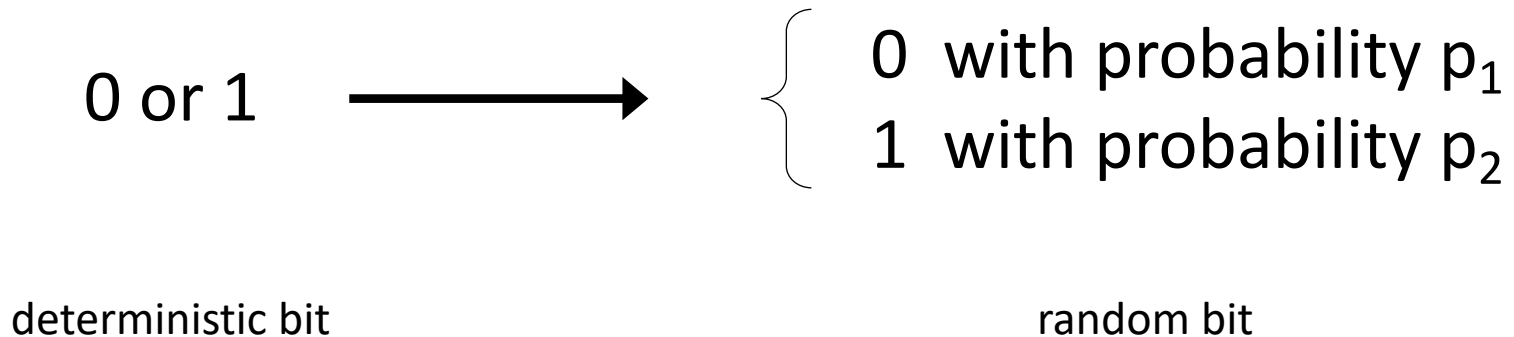
Universal reversible gates



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

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



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

Probabilistic (randomized) computation

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.

New notation

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

Standard notation

0

1

0 with probability p_1
1 with probability p_2

Vector notation

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

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

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

Abstract (Dirac) notation

$|0\rangle$

$|1\rangle$

$$p_1|0\rangle + p_2|1\rangle$$

Gates in the new notation: the NOT gate

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

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

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

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

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

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

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

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

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

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

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

Two-bit gates: the AND gate

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

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

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

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

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

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

Two-bit gates: the CNOT gate

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

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

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

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

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

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

A truly probabilistic gate

We introduce two new gates

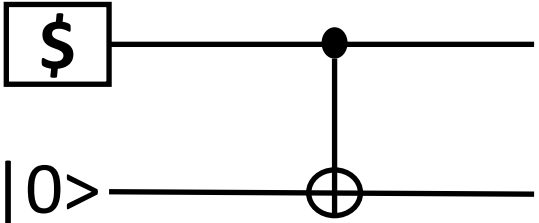
$$\text{COIN} = \boxed{\$} \text{---} \rightarrow \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$$

It has no input and a single bit output. It generates randomly either a 0 or a 1, with probability $1/2$ each. It is like **fair coin tossing**.

$$\text{1COIN} = \text{---} \boxed{1\$} \text{---} = \begin{pmatrix} 1 & 1/2 \\ 0 & 1/2 \end{pmatrix}$$

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

Example 1



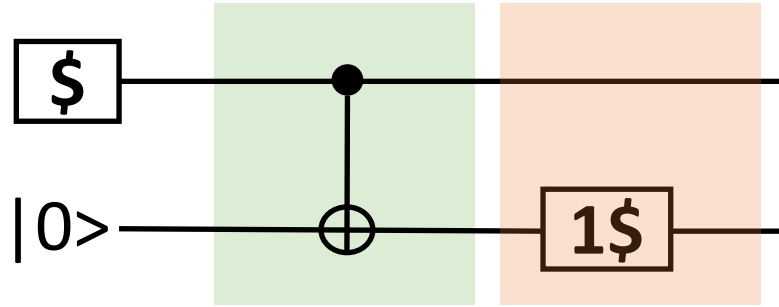
With probability $\frac{1}{2}$ the input bit 00 and with probability $\frac{1}{2}$ it is 10. In the first case the CNOT will leave in unchanged, in the second case it will be changed into 11.

In mathematical terms

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

00
11

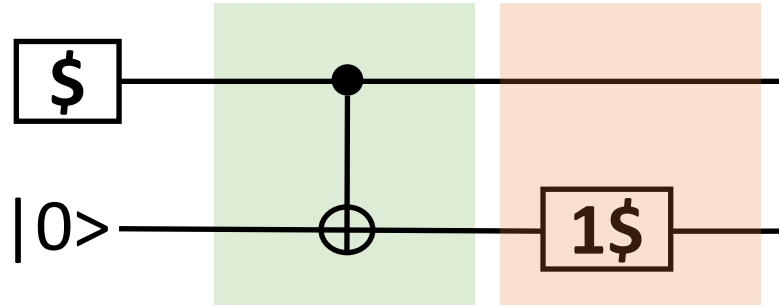
Example 2



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

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

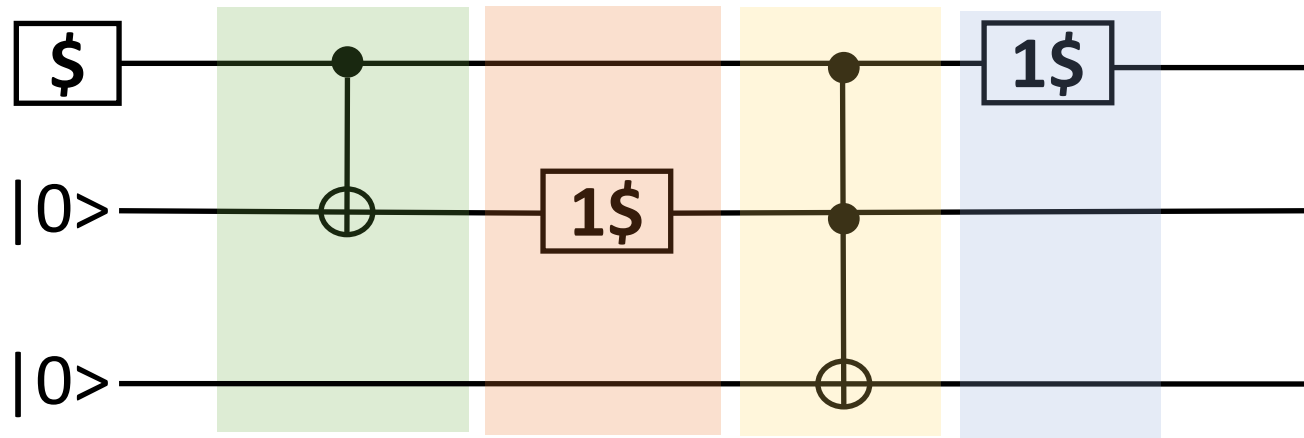
Example 2



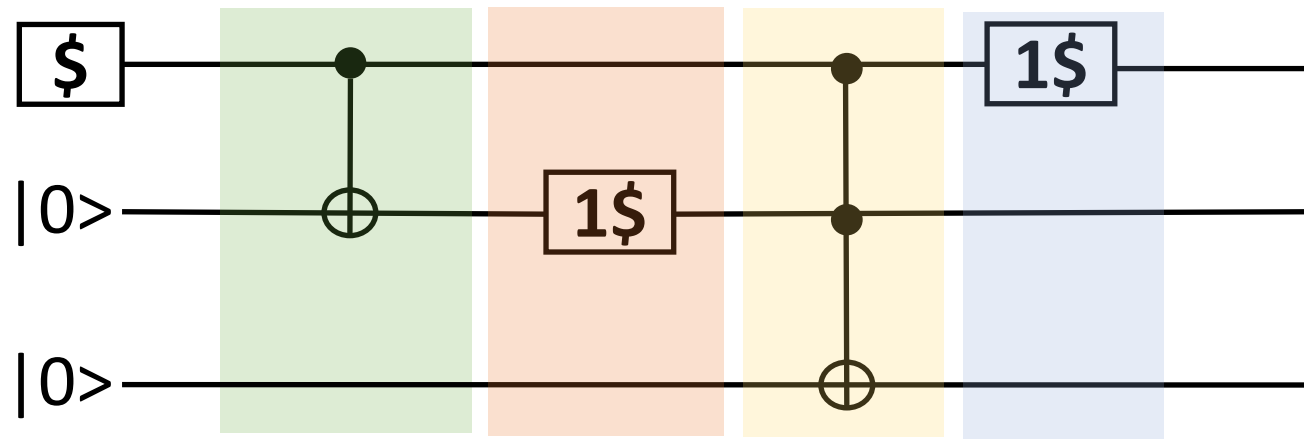
Using the Dirac notation ($|0\rangle \otimes |0\rangle = |00\rangle$, and same for others)

$$\begin{aligned} \frac{1}{2} |00\rangle + \frac{1}{2} |10\rangle &\rightarrow \frac{1}{2} |00\rangle + \frac{1}{2} |11\rangle \rightarrow \frac{1}{2} |00\rangle + \frac{1}{2} \left(\frac{1}{2} |10\rangle + \frac{1}{2} |11\rangle \right) \\ &= \frac{1}{2} |00\rangle + \frac{1}{4} |10\rangle + \frac{1}{4} |11\rangle = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} \end{aligned}$$

Example 3



Example 3



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

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

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

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

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

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:

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

This happens with probability $\frac{5}{8}$

Comment 2

2. We measure the first bit and find 0; this happened with probability

$$\frac{5}{8} + \frac{1}{8} = \frac{3}{4}$$

$$\begin{aligned} \frac{5}{8} |000\rangle + \frac{1}{8} |100\rangle + \frac{1}{8} |011\rangle + \frac{1}{8} |111\rangle &\rightarrow \frac{\frac{5}{8} |000\rangle + \frac{1}{8} |011\rangle}{\frac{3}{4}} \\ &= \frac{5}{6} |000\rangle + \frac{1}{6} |011\rangle \end{aligned}$$

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:

$$P(|000\rangle | \text{“0”}) = P(\text{“0”} | |000\rangle) P(|000\rangle) / P(\text{“0”}) = 1 \times \frac{5}{8} \div \frac{3}{4} = \frac{5}{6}$$

Rules of probabilistic classical computation

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

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

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

States for **multiple bits** are constructed via **tensor product** of \mathbb{R}^2

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

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

Rules of probabilistic classical computation

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 (**$N \times N$ invertible stochastic matrices**).

Of course, they have to preserve probabilities.

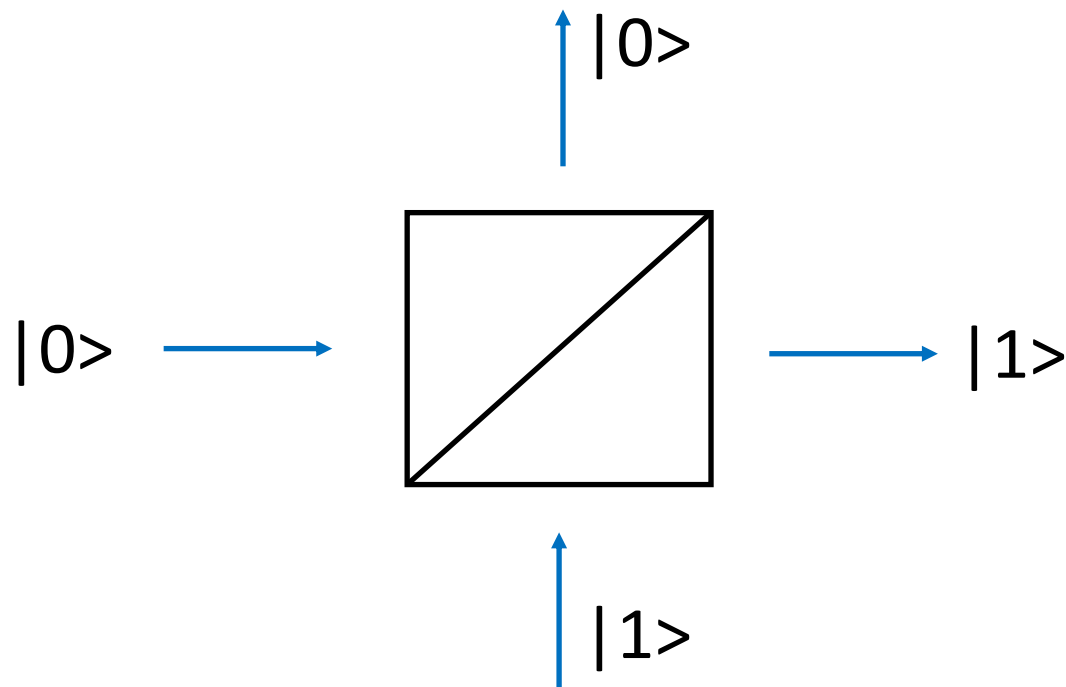
Rules of probabilistic classical computation

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 $\frac{1}{2}$ probability that it goes one way, and $\frac{1}{2}$ 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

$$|0\rangle \rightarrow \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle$$

$$|1\rangle \rightarrow \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle$$

Preview of Quantum Computation

$$\text{---} \boxed{\text{BS}} \text{---} = \boxed{\text{\$}} \text{---} \quad |x\rangle \rightarrow \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle$$

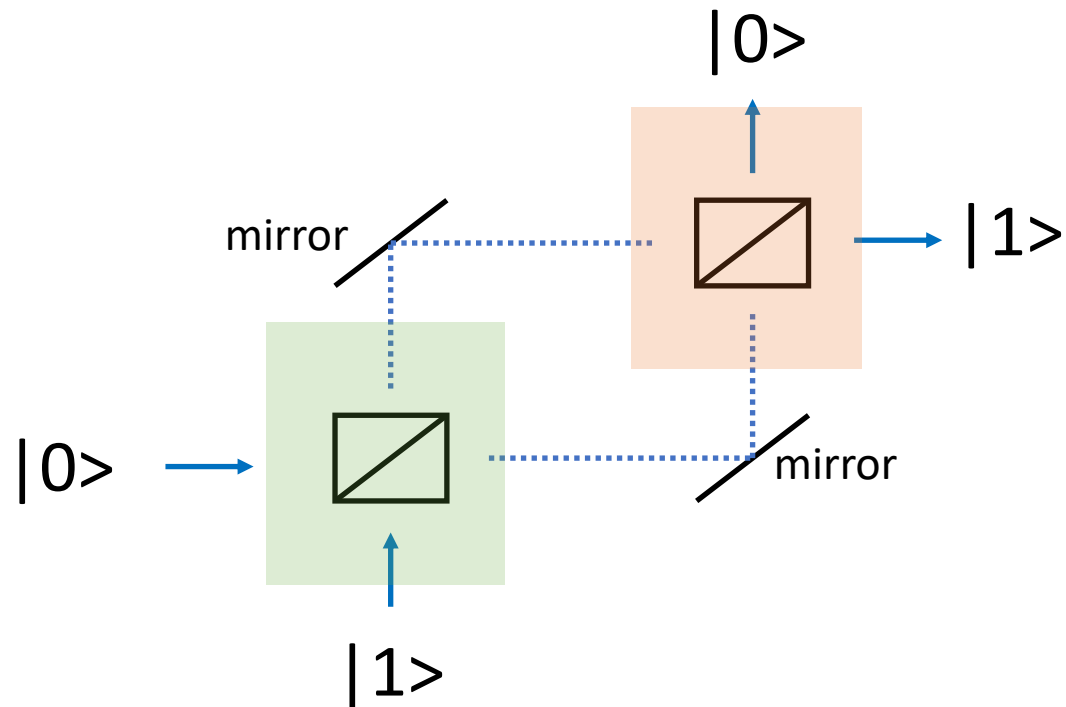
Whatever the input state, it generates an equal weighted distributions of 0 and 1. The matrix representation is:

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

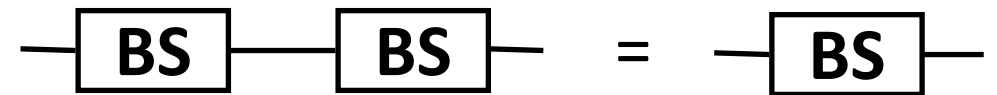
In fact: $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ since $p+q=1$

Preview of Quantum Computation

But now we can do the following optical construction:



This is equivalent to the following circuit



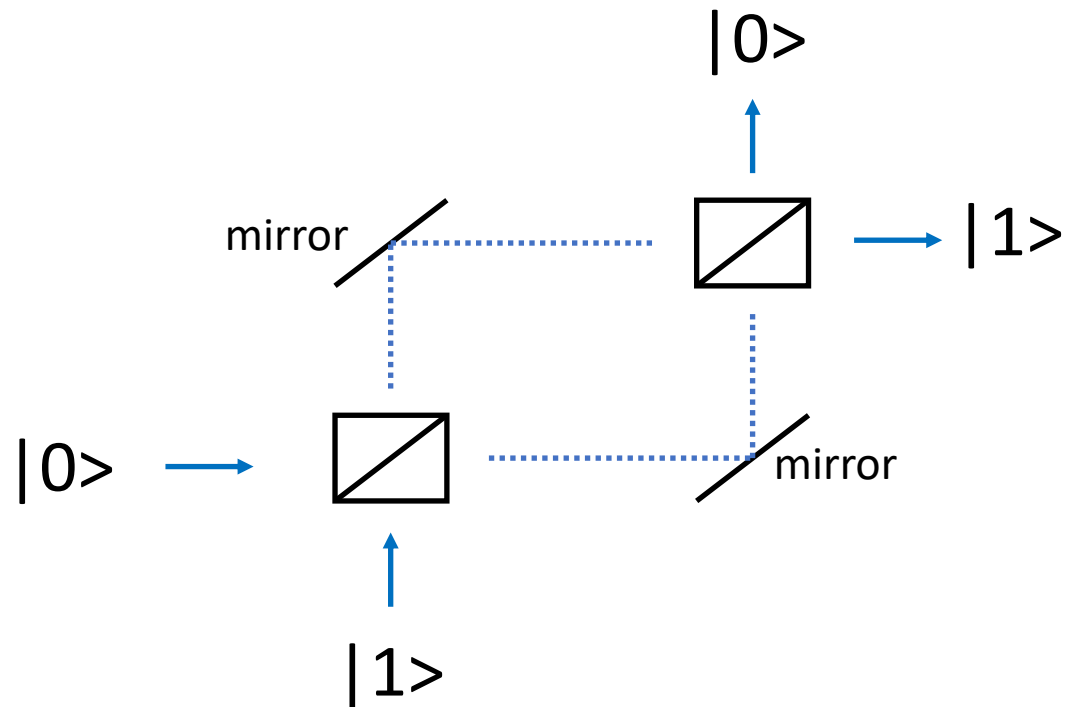
Since

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

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

Preview of Quantum Computation

But this is not what happens. What happens is:



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

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

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

$$|0\rangle \rightarrow \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle \xrightarrow{\text{second BS}} |0\rangle$$

first BS

Instead of

$$|0\rangle \xrightarrow{\text{first BS}} \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle \xrightarrow{\text{second BS}} \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle$$

Preview of Quantum Computation

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 $p, q \in \mathbb{R}^+$ and $p+q=1$

probabilities

changed into

Qubit $\rightarrow a|0\rangle + b|1\rangle$ with $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$

amplitudes

Probabilities
(they remain always positive)

Preview of Quantum Computation

The BS is mathematically described by

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

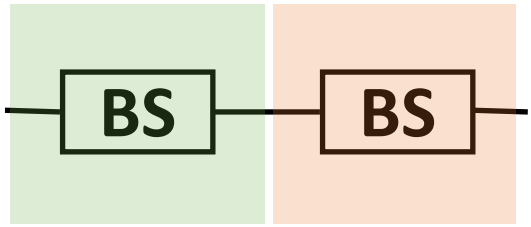
 !!!

Then

$$\begin{array}{l} \text{---} \boxed{\text{H}} \text{---} \\ |0\rangle \rightarrow \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \\ |1\rangle \rightarrow \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \end{array} \left. \vphantom{\begin{array}{l} |0\rangle \\ |1\rangle \end{array}} \right\} \begin{array}{l} \text{In both cases,} \\ \text{probabilities are 50\% of} \\ \text{getting the value 0 or 1} \end{array}$$

Preview of Quantum Computation

But now



The diagram shows two beam splitters (BS) connected in series. The first BS is highlighted with a light green background, and the second BS is highlighted with a light orange background. A horizontal line representing a photon path enters from the left, passes through the first BS, then the second BS, and exits to the right.

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

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.

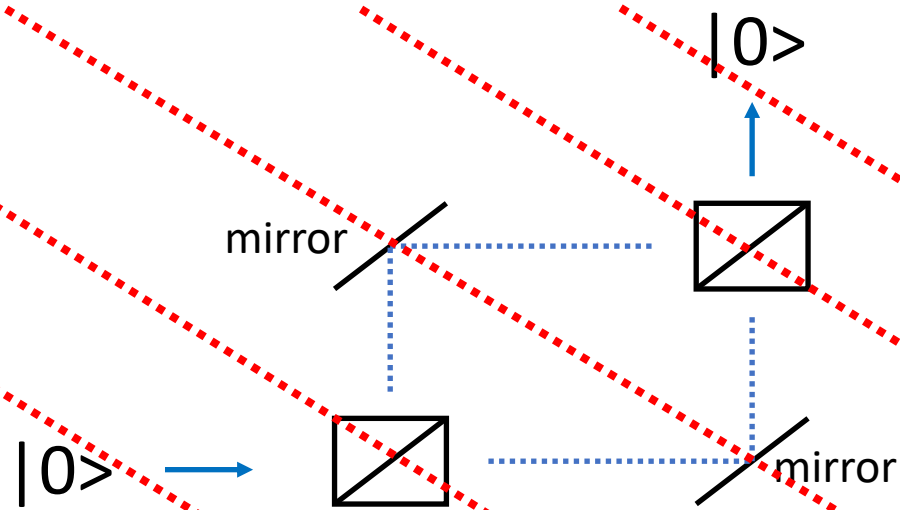
This behaviour can be modelled by classical waves

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 Algorithms



Initialize the state

Create the superposition of all states
Like parallel processing

Compute the function

Let the state interfere so
that the correct answer
has higher probability

Read the output