Bits and Qubits

J. J. García-Ripoll
IFF, CSIC Madrid

(13-4-2009)

What is “computing”?

What is “computing”?

Computar

- “Contar o calcular por números algo”
  RAE online

Compute

- “ to determine especially by mathematical means”
  Merrian-Webster

The natural acception we as physicists tend to think.

What is “computing”?

Computar

- “Contar o calcular por números algo”
  RAE online

Compute

- “ to determine especially by mathematical means”
  Merrian-Webster

The natural acception we as physicists tend to think.

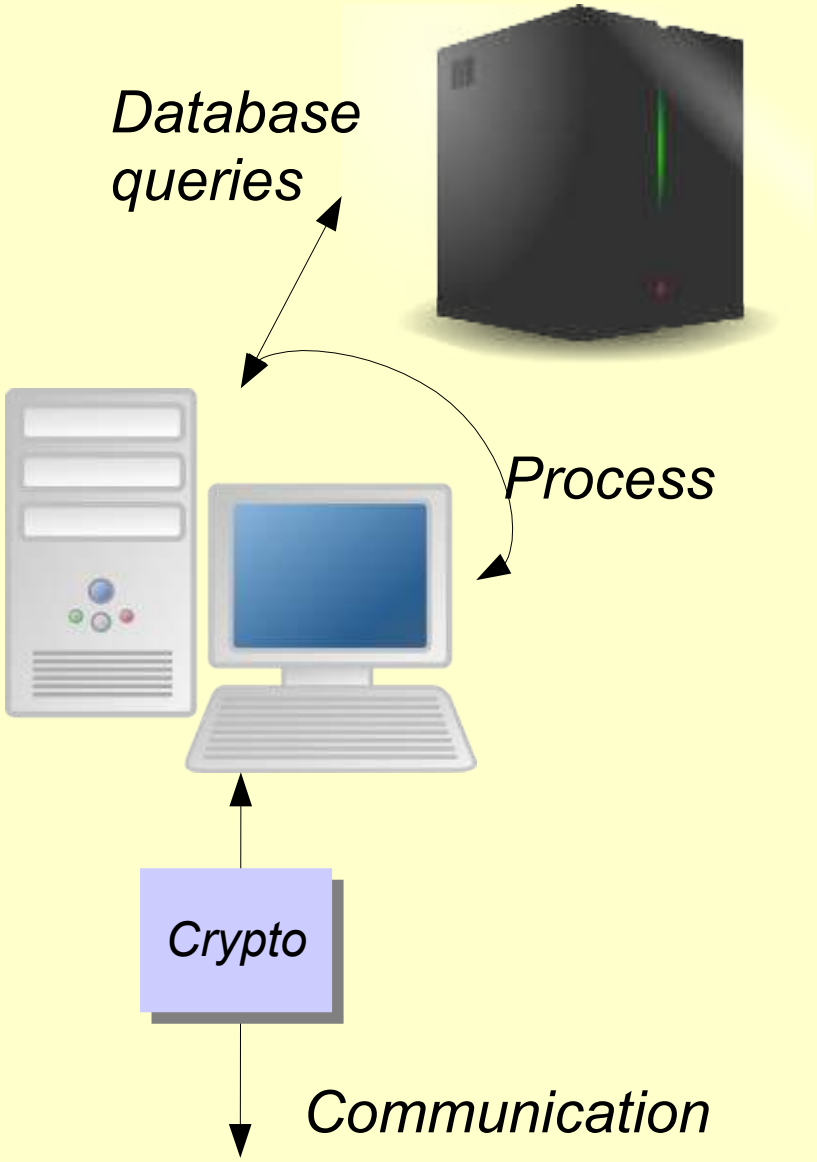
Computer

- “device for processing, storing, and displaying information.”
  Encyclopædia Britannica

Computer

- “a **programmable** usually electronic device that can **store, retrieve, and process data**”
  Merrian-Webster

What is “computing”?



Computer

- “device for processing, storing, and displaying information.” Encyclopædia Britannica

Computer

- “a **programmable** usually electronic device that can **store, retrieve, and process data**” Merrian-Webster

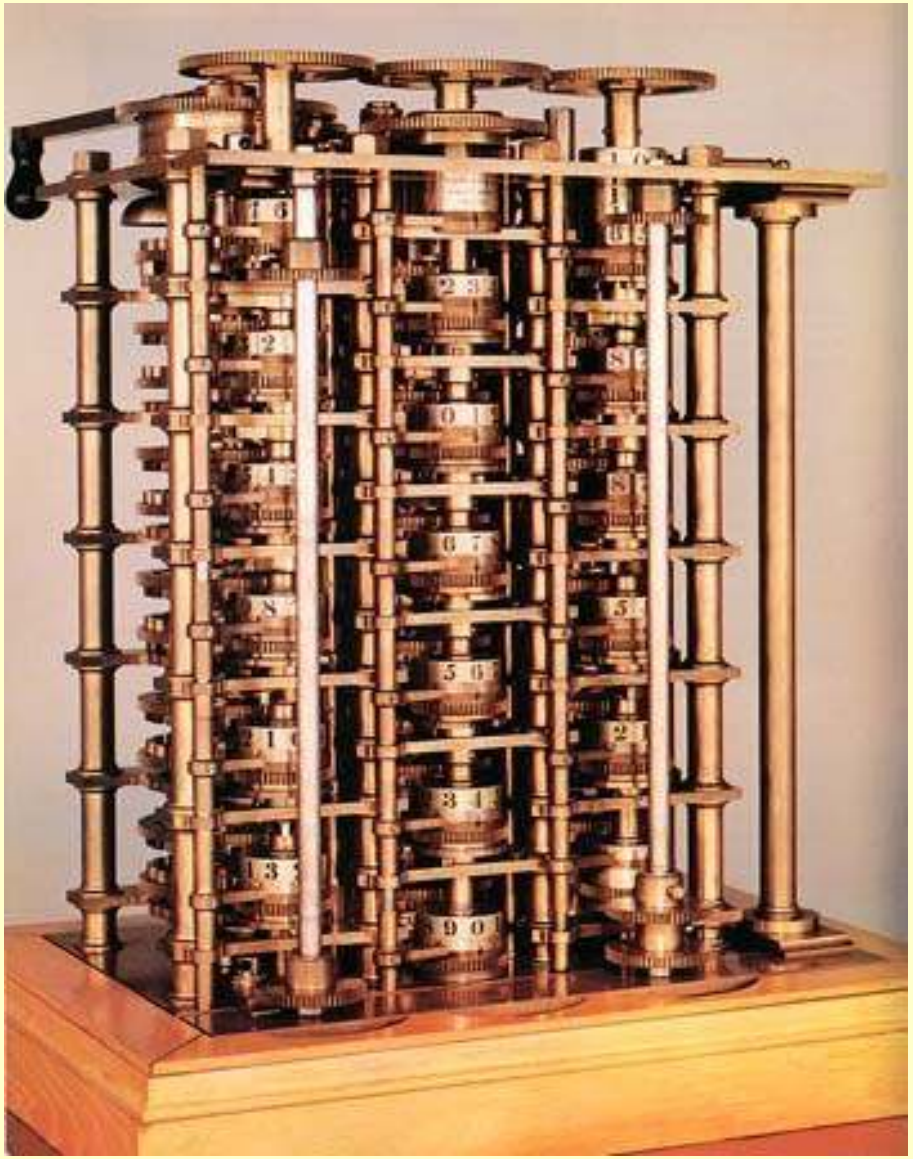
We are going to have to reproduce a lot of things available right now!

The origins of computation

The origins of computation

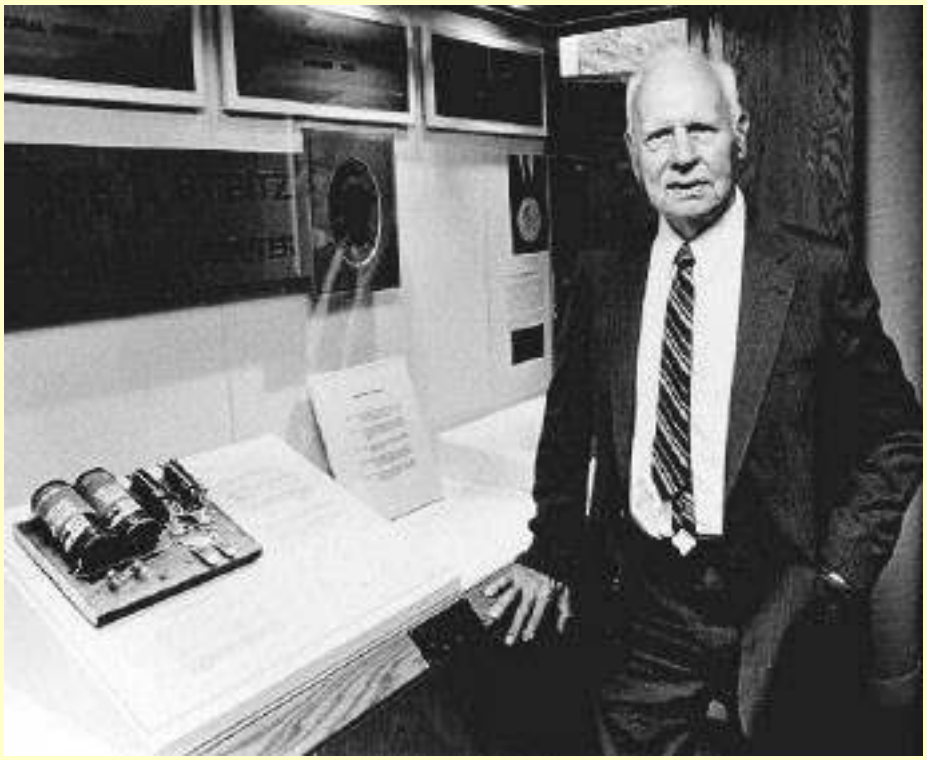
- 1801, J. M. Jacquard invents a loom programmed by punched tapes.

The origins of computation



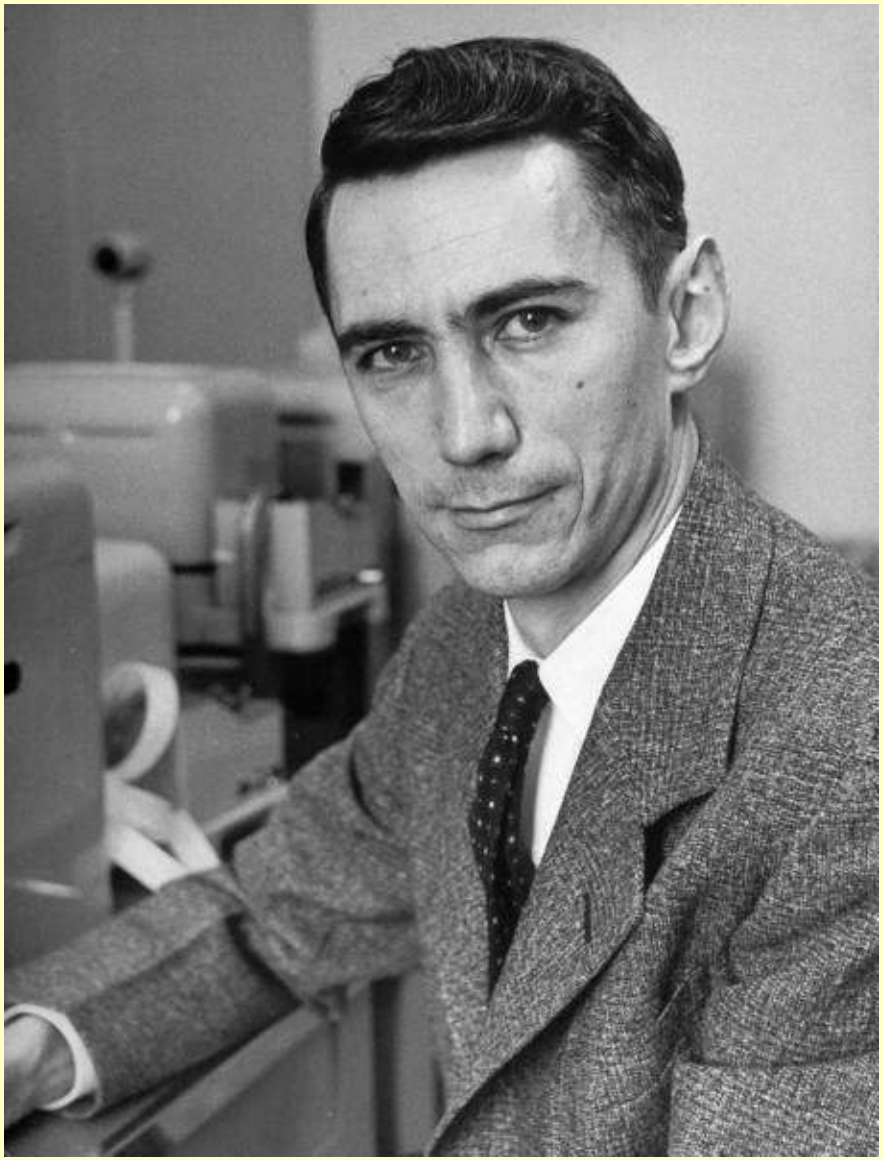
- 1801, J. M. Jacquard invents a loom programmed by punched tapes.
- 1837, Charles Babbage designs the difference machine and later on the “analyical engine”.
- Ada Lovelace develops the first computer program.

The origins of computation



- 1801, J. M. Jacquard invents a loom programmed by punched tapes.
- 1837, Charles Babbage designs the difference machine and later on the “analyical engine”.
- Ada Lovelace develops the first computer program.
- 1937, George R. Stibitz develops the first binary adder (Model K), based on electronic relays.

The origins of computation



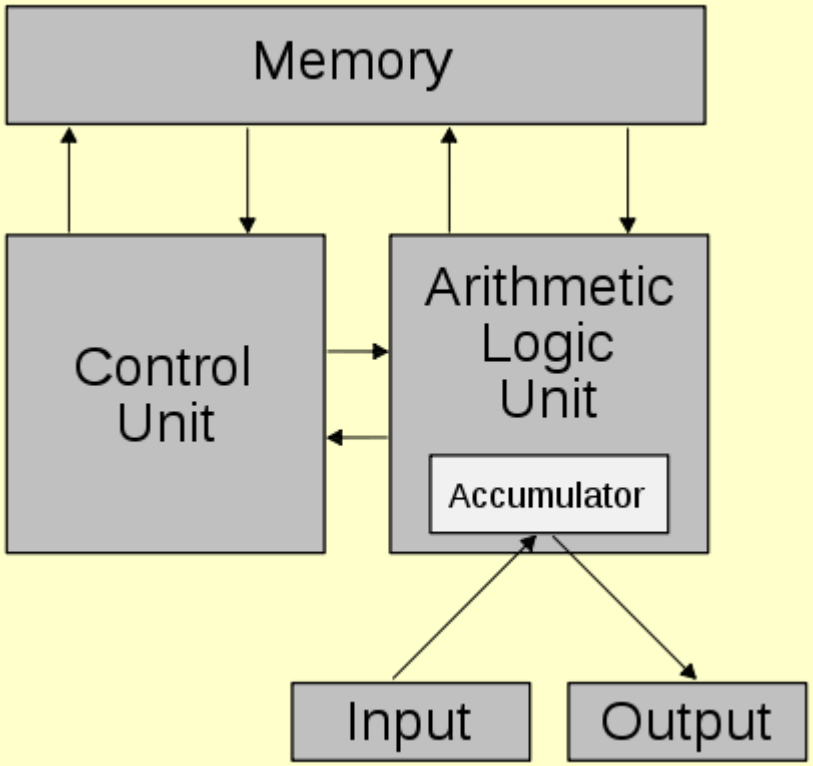
- 1937, Claude Shannon invents digital electronics, introducing Boolean logic.

The origins of computation

Name	First operational	Numeral system	Computing mechanism
Zuse Z3 (Germany)	May 1941	Binary	Electro-mechanical
Atanasoff–Berry Computer (US)	1942	Binary	Electronic
Colossus Mark 1 (UK)	February 1944	Binary	Electronic
Harvard Mark I – IBM ASCC (US)	May 1944	Decimal	Electro-mechanical
Colossus Mark 2 (UK)	June 1944	Binary	Electronic
ENIAC (US)	July 1946	Decimal	Electronic
Manchester Small-Scale Experimental Machine (UK)	June 1948	Binary	Electronic
Modified ENIAC (US)	September 1948	Decimal	Electronic
EDSAC (UK)	May 1949	Binary	Electronic
Manchester Mark 1 (UK)	October 1949	Binary	Electronic
CSIRAC (Australia)	November 1949	Binary	Electronic

- 1937, Claude Shannon invents digital electronics, introducing Boolean logic.
- Progress continued using electromechanical elements, punched cards, vacuum tubes.

The origins of computation



- 1937, Claude Shannon invents digital electronics, introducing Boolean logic.
- Progress continued using electromechanical elements, punched cards, vacuum tubes.
- 1940, J. von Neumann introduces the “stored program architecture”.
- 1947, 1st transistor computer
- 1960, microprocessors.

Bits and Boolean logic

Bits and Boolean logic

$\{0,1\}$ *on/off*

$f : \mathbb{Z}_2^N \to \mathbb{Z}_2$ *circuit*

$\{0,1\}^{\otimes n}$ *integers*

floats

alphabet

…

- Systematized by Leibniz as numeral system.
- George Boole uses it as a basis for logical calculus.
- C. Shannon showed the perfect match between boolean logic and circuits
- Simple and absolutely general encoding of information: letters, numbers, signals, etc.
- Small number of functions, systematically mappable to digital circuits.

Fundamental gates

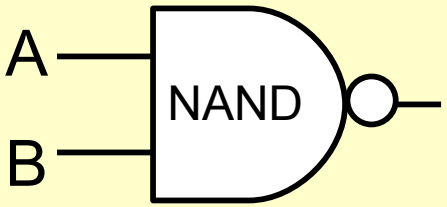
A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

- NAND = Not-AND
- “Set unless both 1”
- It is not invertible.

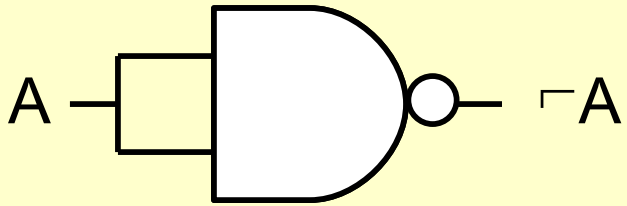
A	B	NOR
0	0	1
1	0	0
0	1	0
1	1	0

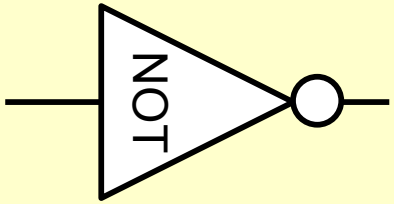
- NOR = Not – OR
- “Set when both zero”
- Again, not invertible.

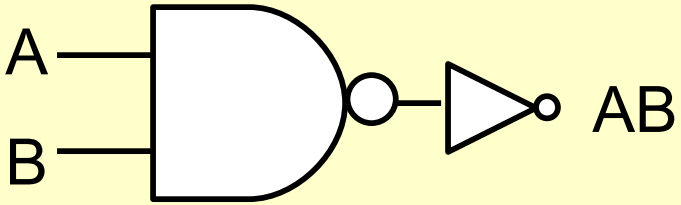
Functional completeness

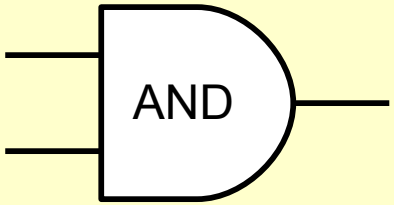


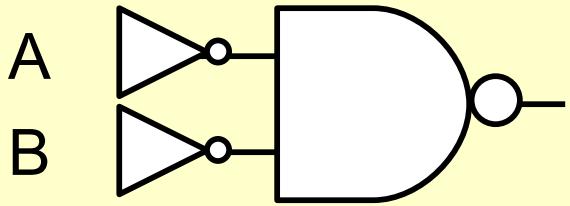
A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

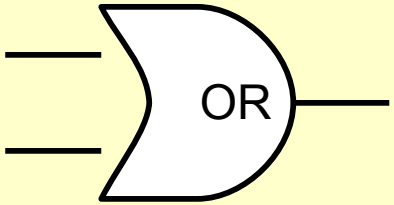


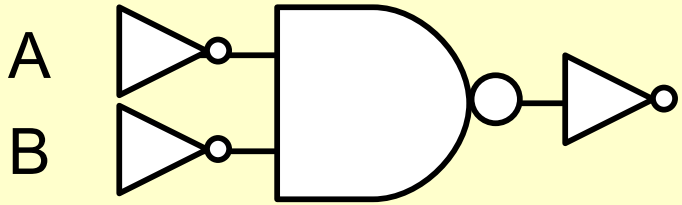


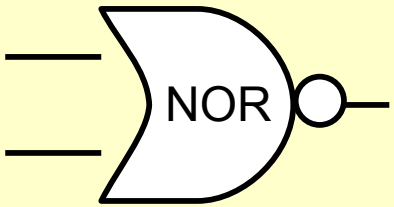




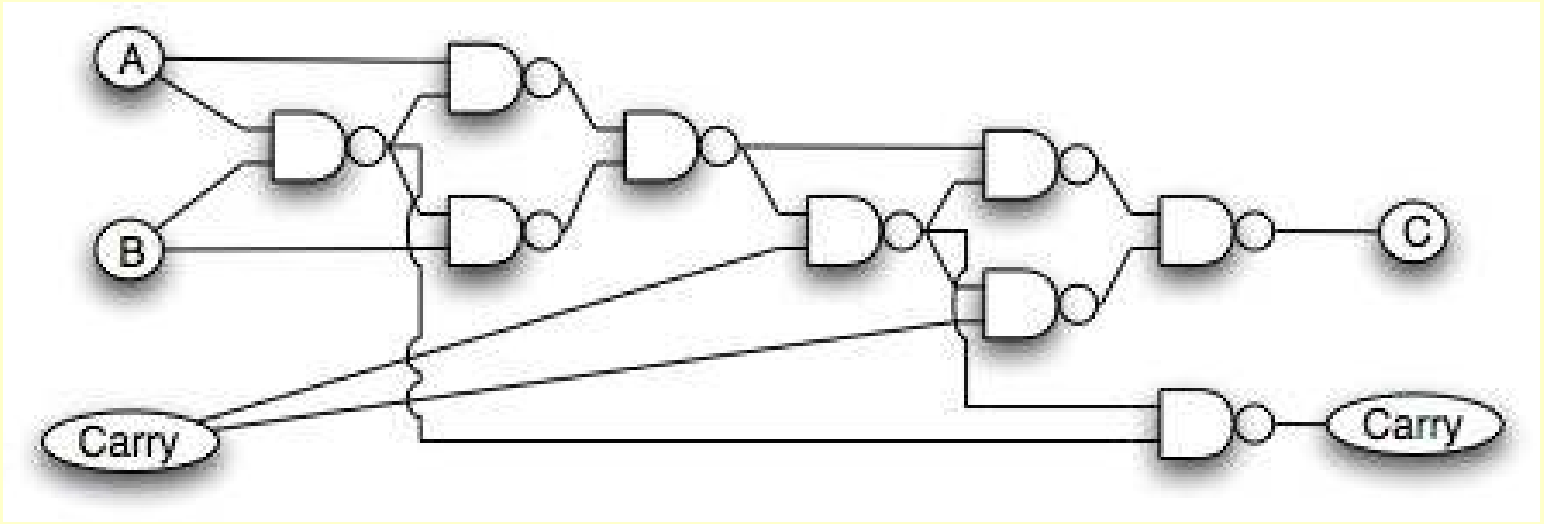








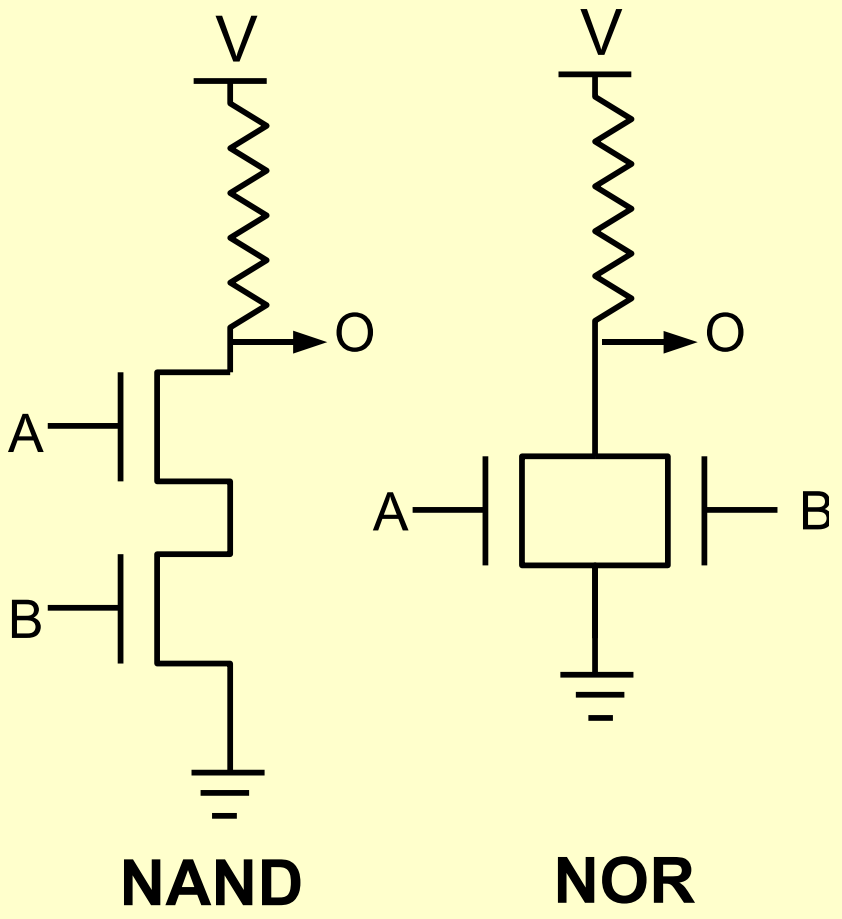
Adding



A	B	C	Carry
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

- Using logical operations and ancillas, it is also possible to implement any computation.
- For instance sum A+B with carry (not just OR)

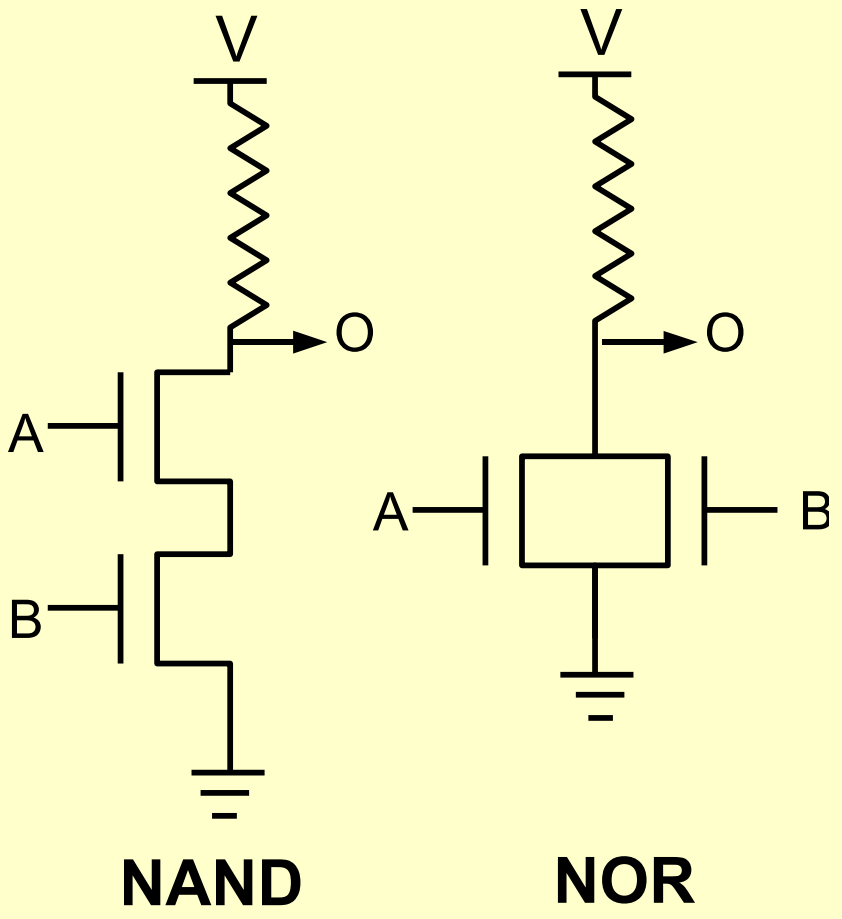
Physical implementation



NAND **NOR**

- Design based on NMOS field transistors
  - A,B = 0 = Low bias signal, low resistance
  - A,B = 1 = High bias, high resitance
- In case of open paths (A,B=1), current may flow to ground (O=0)
- Otherwise signal, O=1.

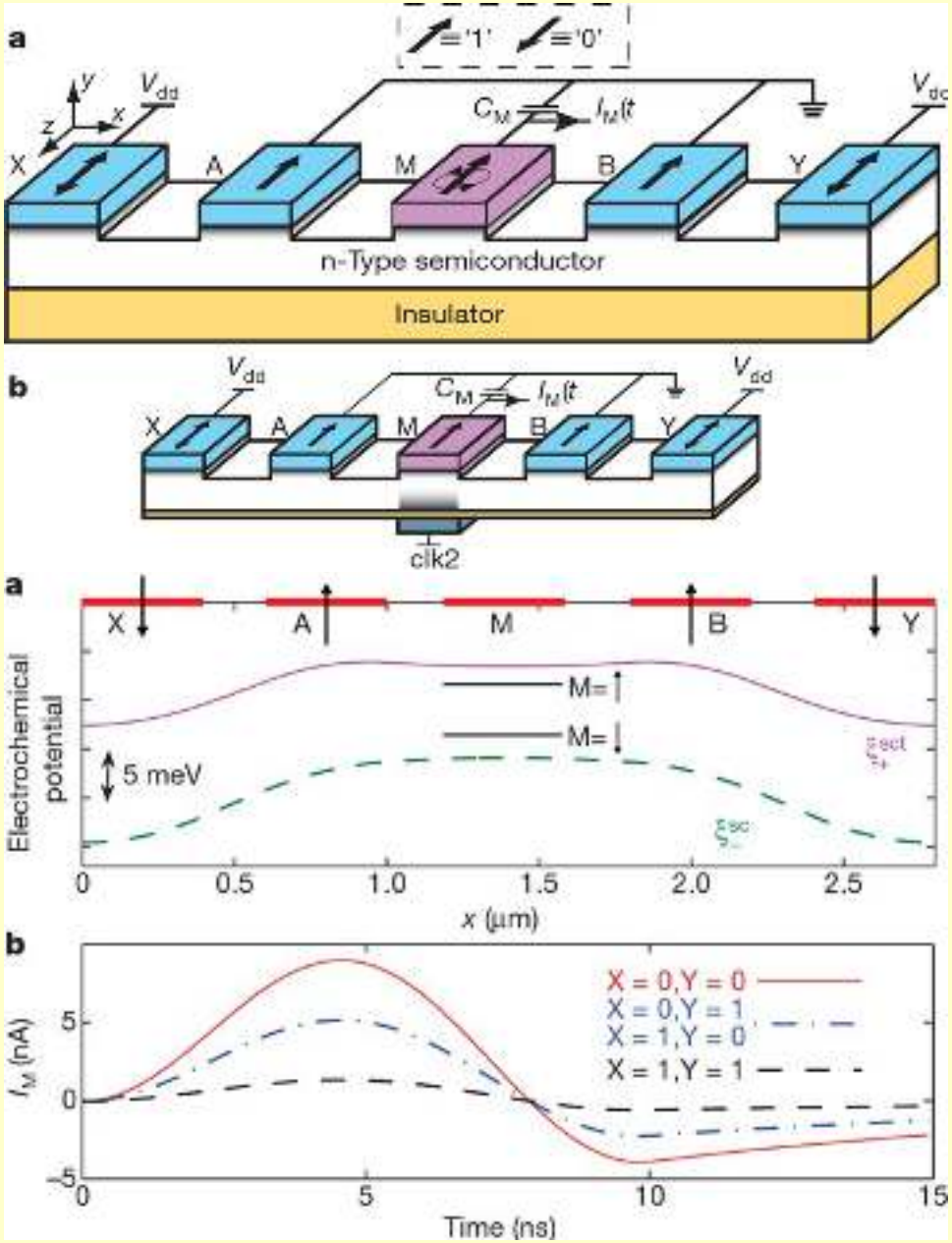
Problems



NAND **NOR**

- **Dissipation and heating**
  - Resistive elements
  - Resistance increases with temperature
- Threshold diminishes with smaller elements
  - More difficult to differentiate on/off
  - Quantum tunneling
- Long range capacitances between different gates

One solution: spintronics



- Use spin polarized currents and giant magnetoresistance
- Faster switching times
- Larger integration
  - High density hard disk
  - Cheaper, faster RAMs
- Still based on dissipative currents.

Reversible computing

- Engineer gates that are reversible.
- No dissipation, just logical operations.
- Ancillas will be needed.
- Valid paradigm.

Toffoli gate

I1	I2	I3	O1	O2	O3
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

Reversible computing

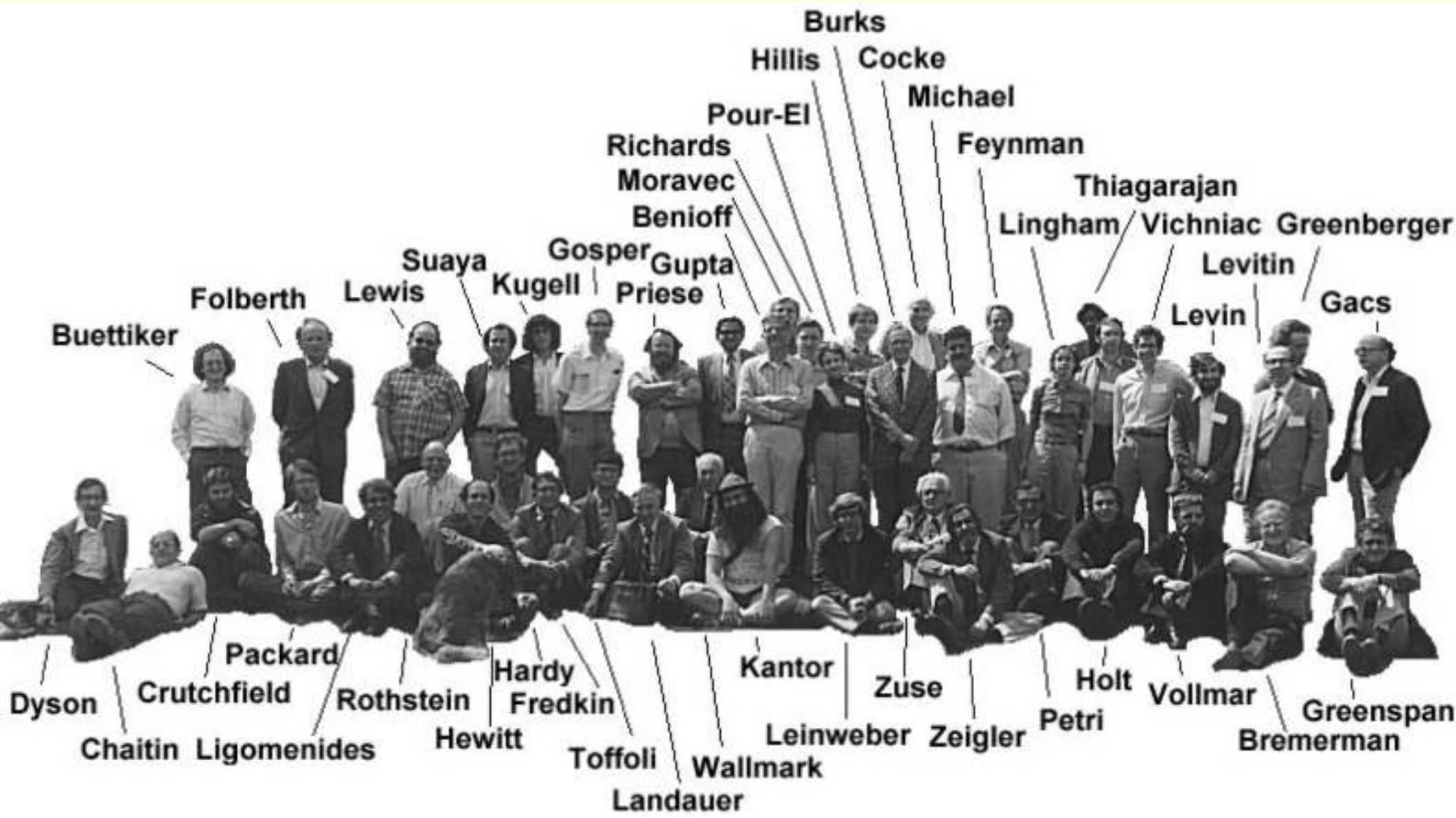
- Engineer gates that are reversible.
- No dissipation, just logical operations.
- Ancillas will be needed.
- Valid paradigm.
- All optical computing?
  ...
- Disregards the entropic cost of ancillas
  - Landauer principle

Toffoli gate

I1	I2	I3	O1	O2	O3
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

NAND subset

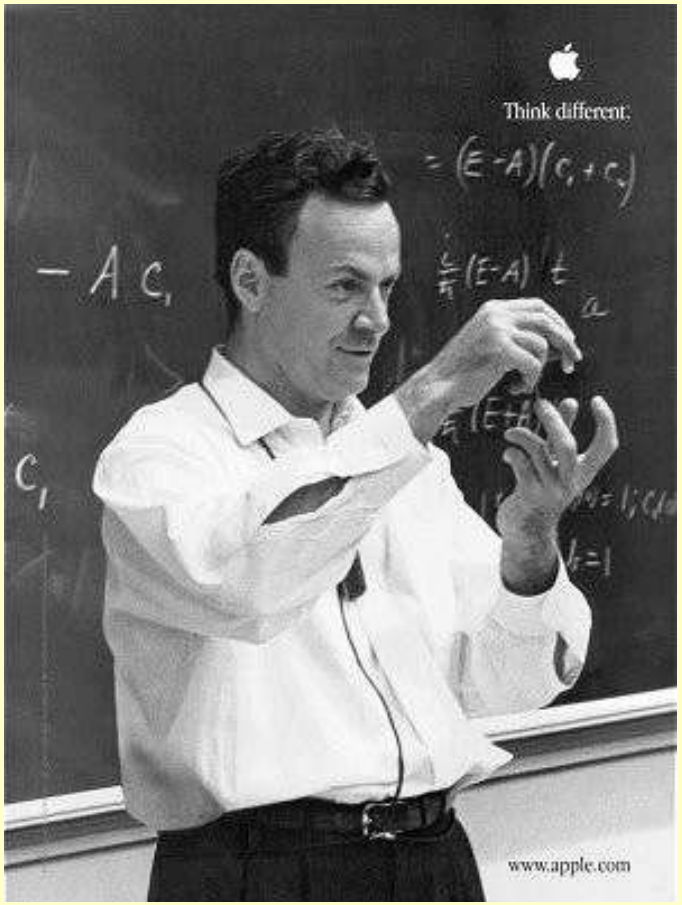
Quantum computing



1st conference on Physics and Computation

Feynman on Q Computers (1981)

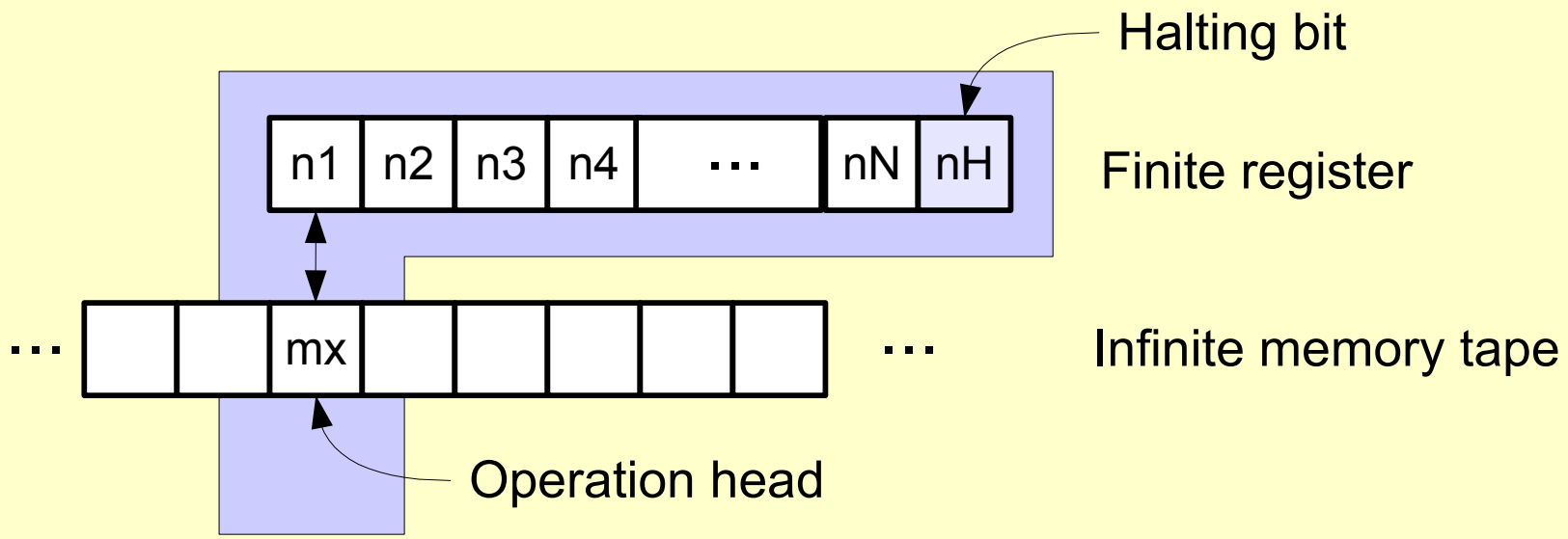
The full description of quantum mechanics is given by a function [with] too many variables [...] How can we simulate quantum mechanics? [...]



- Let the computer be built with quantum mechanical elements that obey quantum mechanical laws
- Let the computer be a logical universal automaton, can we imitate [quantum mechanics]?

R. P. Feynmann, *Simulating Physics with Computers*,
Int. Jour. Theor. Phys. 21, 1982

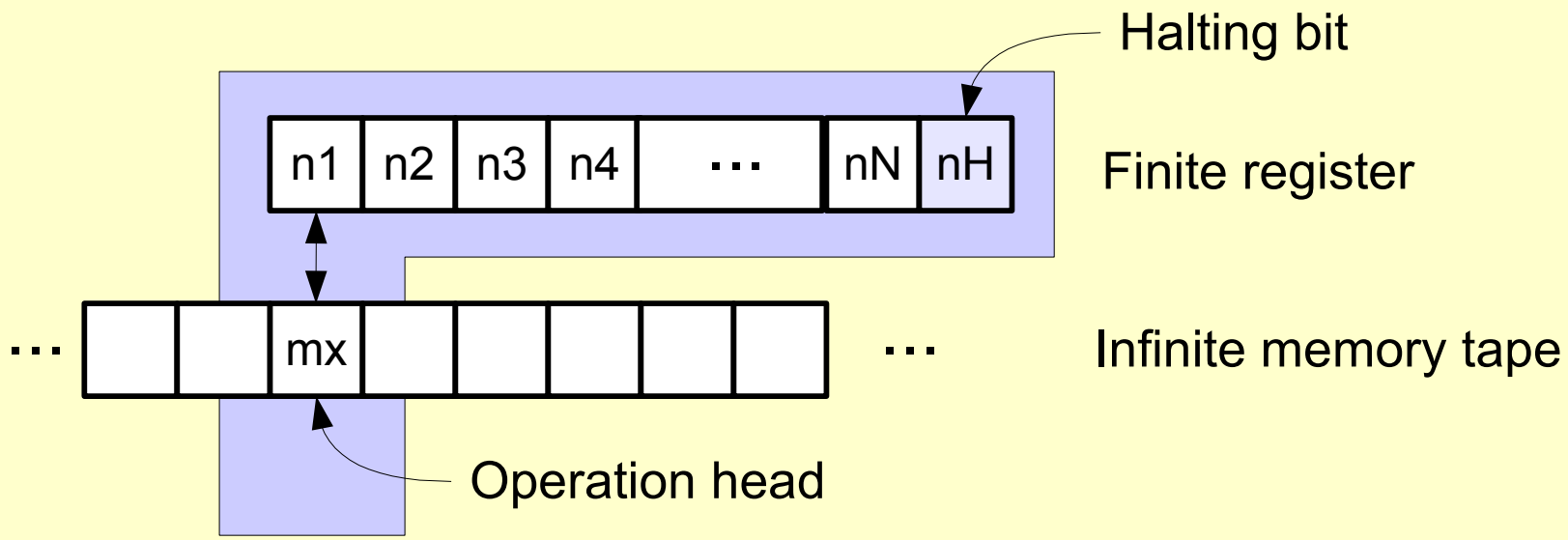
Turing machine (1936)

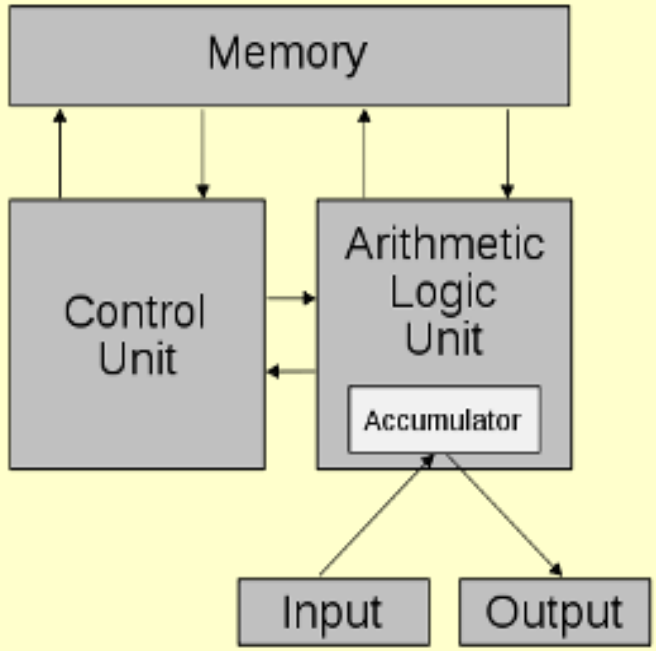


Computation gate U: (n,m,x) → (n´,m´,x±1)

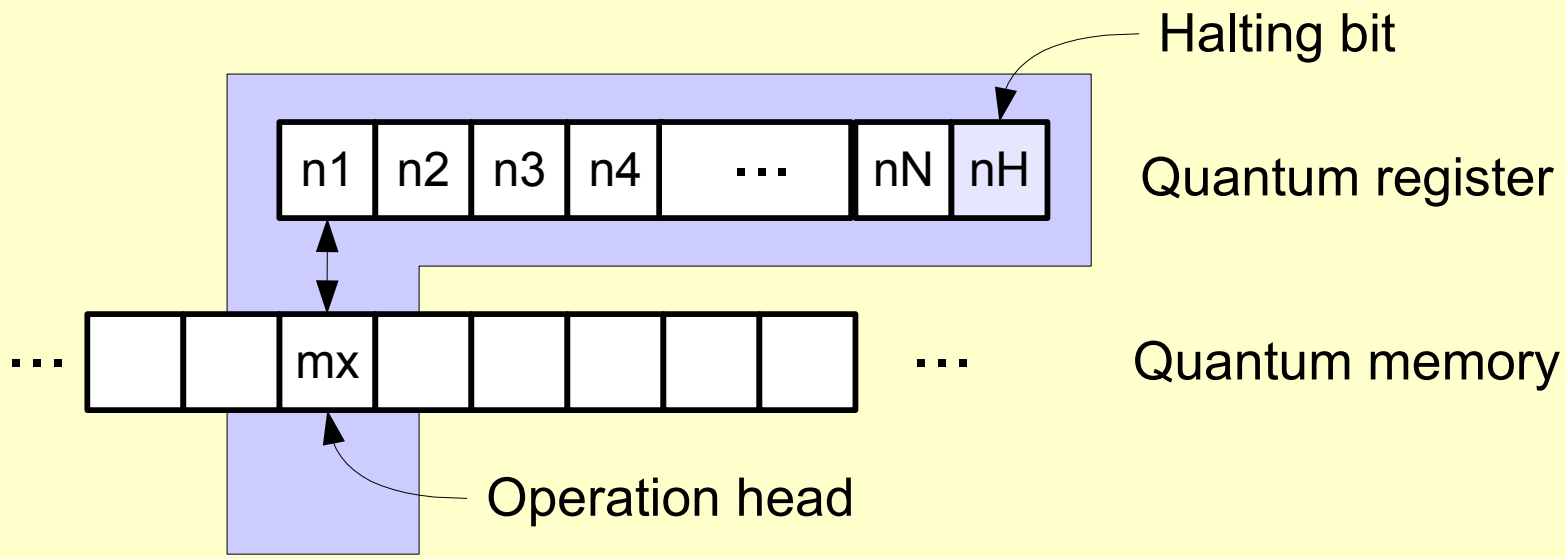
- Simplest computer model, mathematically tractable and universal.
- Program and data are both in the tape and processed using the same operation U, which changes both and decides motion of the head.
- Turing thesis: “every calculable function is computable by a T.M.”
- Strong TT: “every reasonable model of computation can be simulated by a probabilistic T.M.”

Turing machine (1936)





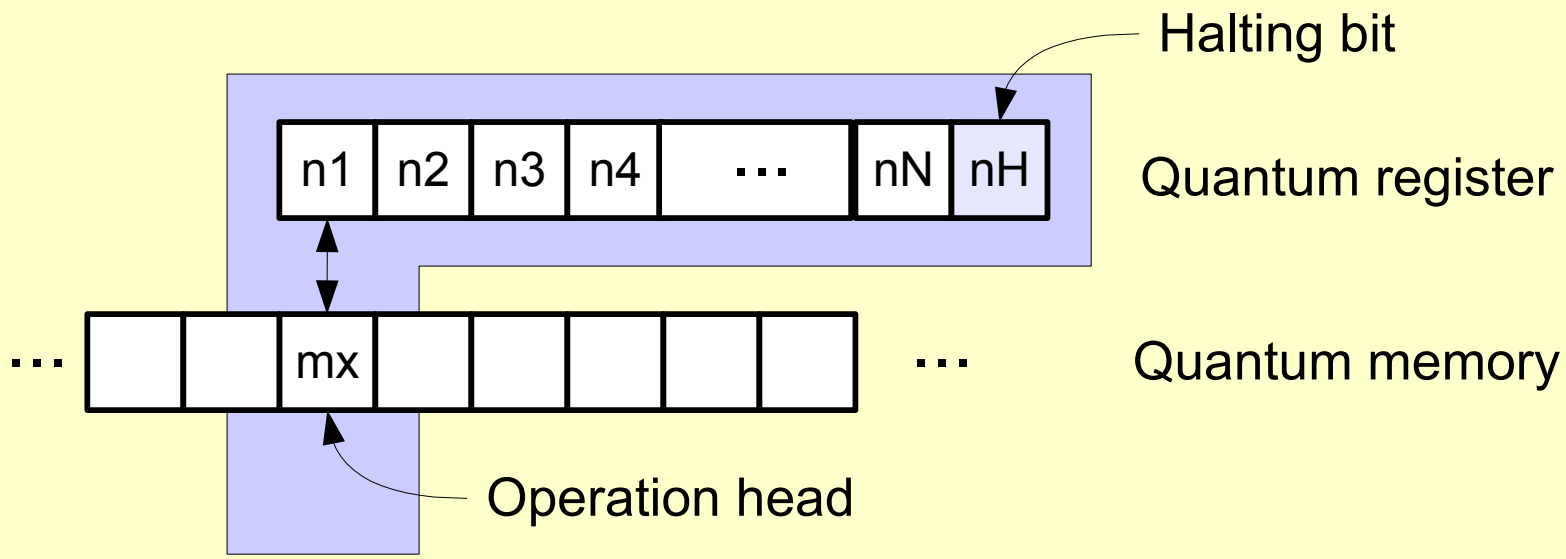
Quantum Turing machine (1985)



Universal gate U $|n,m,x) \rightarrow |n',m',x\pm1)$

- Generalization of the classical Turing machine.
- The register and memory are quantum states.
- The operation U is a unitary acting on the register and memory.
- Can implement **reversibly** any computer program TM $\subset$ QTM.
- Can simulate an arbitrary quantum operation.

Quantum Turing machine (1985)



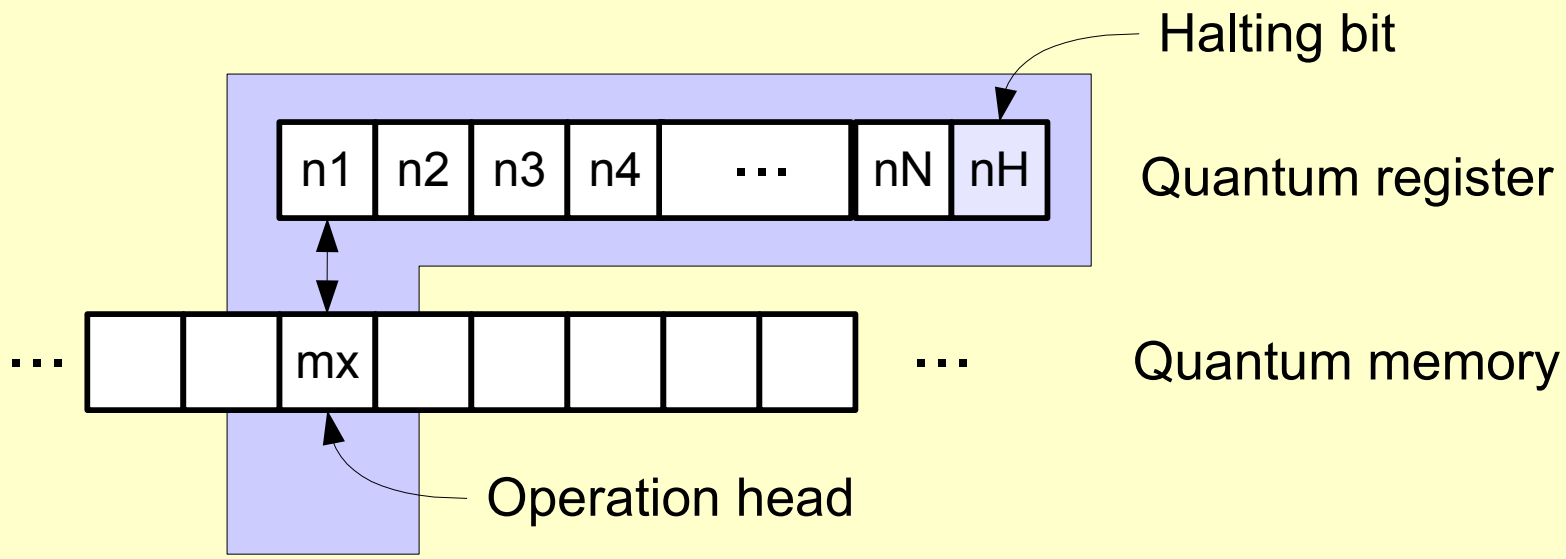
Universal gate U + decision on motion

Faster computers

One day it will become technologically possible to build quantum computers, perhaps using flux quanta (Likharev 1982; Leggett 1985) as the fundamental components. It is to be expected that such computers could operate at effective computational speeds in excess of Turing-type machines built with the same technology. This may seem surprising since I have established that no recursive function can be computed by $\mathcal{Q}$ on average more rapidly with the help of quantum programs than without. However, the idealizations in $\mathcal{Q}$ take no account of the

Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer
D. Deutsch, Proc. Royal Soc. London A 400, (1985), pp. 97-117

Quantum Turing machine (1985)



Universal gate U + decision on motion

- If the computer is fed with a superposition of input states, it can perform **both computations in parallel**
- In particular, can compute f(0) ⊕ f(1) in a single run, discovering whether f(x) is biased or not

(Deutsch algorithm)

Timeline

- 1982, Feynman, universal quantum simulator
- *1984, Bennett, Brassard, Quantum Key Distribution*
- 1985, Deutsch algorithm for a biased coin
- 1992, Deutsch-Jozsa algorithm
- *1992, Experimental implementation of BB84*
- *1992, Ekert, EPR based quantum crypto*
- 1994, Shor, factoring algorithm
- 1996, Grover, quantum database searching

...

Fixing ideas

Qubits

- Two-level quantum systems.

$$|\psi\rangle \in \mathbb{C}^2 \sim lin\{|0\rangle, |1\rangle\}$$

- Vectors in a complex space.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

- May create arbitrary superpositions.

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}} i|1\rangle, |0\rangle, \ldots$$

- Multiple qubits form a register

$$|\psi_N\rangle \in \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = \mathbb{C}^{2N}$$

- Configuration space size grows exponentially.

$$dim\{|s_1, s_2 \ldots s_N\rangle\} = 2^N$$

Statistical mixtures

- Previous description based on pure states.
- Pure states are rare.
- We must introduce density matrices.

$$\rho = \frac{1}{2}\begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix} = |\psi\rangle\langle\psi|$$

- Statistical mixtures of different realizations or outcomes.
- These superpositions can be classically prepared.

$$\rho = \begin{pmatrix} 1-e^{-\beta T} & 0 \\ 0 & e^{-\beta T} \end{pmatrix} \neq |\psi\rangle\langle\psi|$$

$$= (1-e^{-\beta T})|0\rangle\langle 0| + e^{-\beta T}|1\rangle\langle 1|$$

Single qubit gates

- Simplest operations are **unitary rotations**

$$U_{NOT} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

- In the language of complex vectors, SU(2) matrices

$$U(\theta, \vec{n}) = \exp(-i\theta \vec{n}\cdot\vec{\sigma}) = \cos(\theta) + i\sin(\theta)(\vec{n}\cdot\vec{\sigma})$$

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- This does not include two-bit logical operations (NAND, NOR) even if reversible (Toffoli)

Single qubit measurements

- The simplest class are **projective measurements** of a Hermitian observable.
- For example

$$\sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

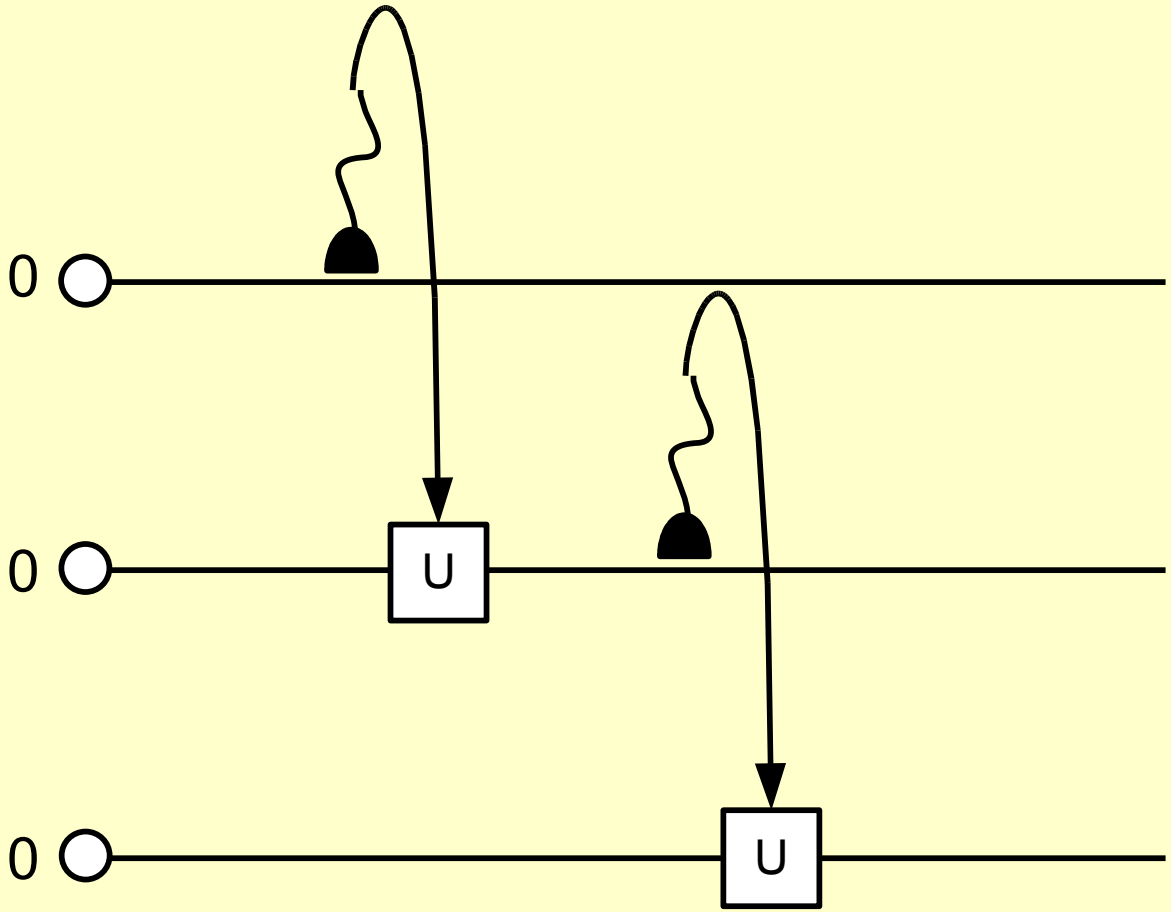
with probabilities

$$p_{\sigma_z=+1} = tr(|0\rangle\langle 0|\rho) \qquad p_{\sigma_z=-1} = tr(|1\rangle\langle 1|\rho)$$

the state will be projected to |0) and |1) and obtain (+1) and (-1) as the outcome of the measurement.

- This can be used to **prepare states locally**: measure the qubit and correct if it is in wrong state.

LOCC



- LOCC: Local Operations and Classical Communication
- Measurement + local unitaries + comm. of classical data

Entanglement

- There are states that can not be created with the previous tools

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}} i |11\rangle$$

- A product state is one that can be created by local operations and classical communication

$$|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_N\rangle$$ ← Local preparation

$$\rho = \sum_i p_i \rho_1^{(i)} \otimes \cdots \otimes \rho_N^{(i)}$$ ← + classical comm.

- Anything else, like above, is said to be an **entangled state**.

Entanglement

- Bell basis of two-qubit states

$$|\psi_{\pm}\rangle = \frac{1}{\sqrt{2}}|00\rangle \pm \frac{1}{\sqrt{2}}|11\rangle \qquad |\phi_{\pm}\rangle = \frac{1}{\sqrt{2}}|01\rangle \pm \frac{1}{\sqrt{2}}|10\rangle$$

- Maximally entangled states of two qubits.
- The measurements on one qubit are perfectly correlated or anticorrelated with the other qubit
- For instance

$$P_{\phi_-}(0,1) = P_{\phi_-}(1,0) = \frac{1}{2}, else\ 0$$

even if we rotate both qubits around the same angle.

Interaction

- How to create entanglement? We need some unitary operation among 2 or more qubits.
- For instance

$$U_{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \sigma_x$$

$$U_{CNOT}(|0\rangle + |1\rangle)|0\rangle \sim |\psi_+\rangle$$

- Any physical implementation of a unitary comes from a Hamiltonian
  - The qubits have to interact

Interaction and computation

Classically

- Bits
- Local gates (NOT)
- Many-qubit functions (&, $\wedge$, $\vee$)
- Universal gates (NAND)
- Gate composition
- Read-out

- **Quantum mechanics**
- Qubits
- Local unitaries
- **Multi-qubit unitaries**
- **Universal quantum gate**
- Gate composition
- Measurement

- We need some kind of interaction to perform **multi-qubit gates**.
- These quantum gates will be unitaries and thus **reversible**.
- The gates will not be the same as classically.

Multiqubit operations

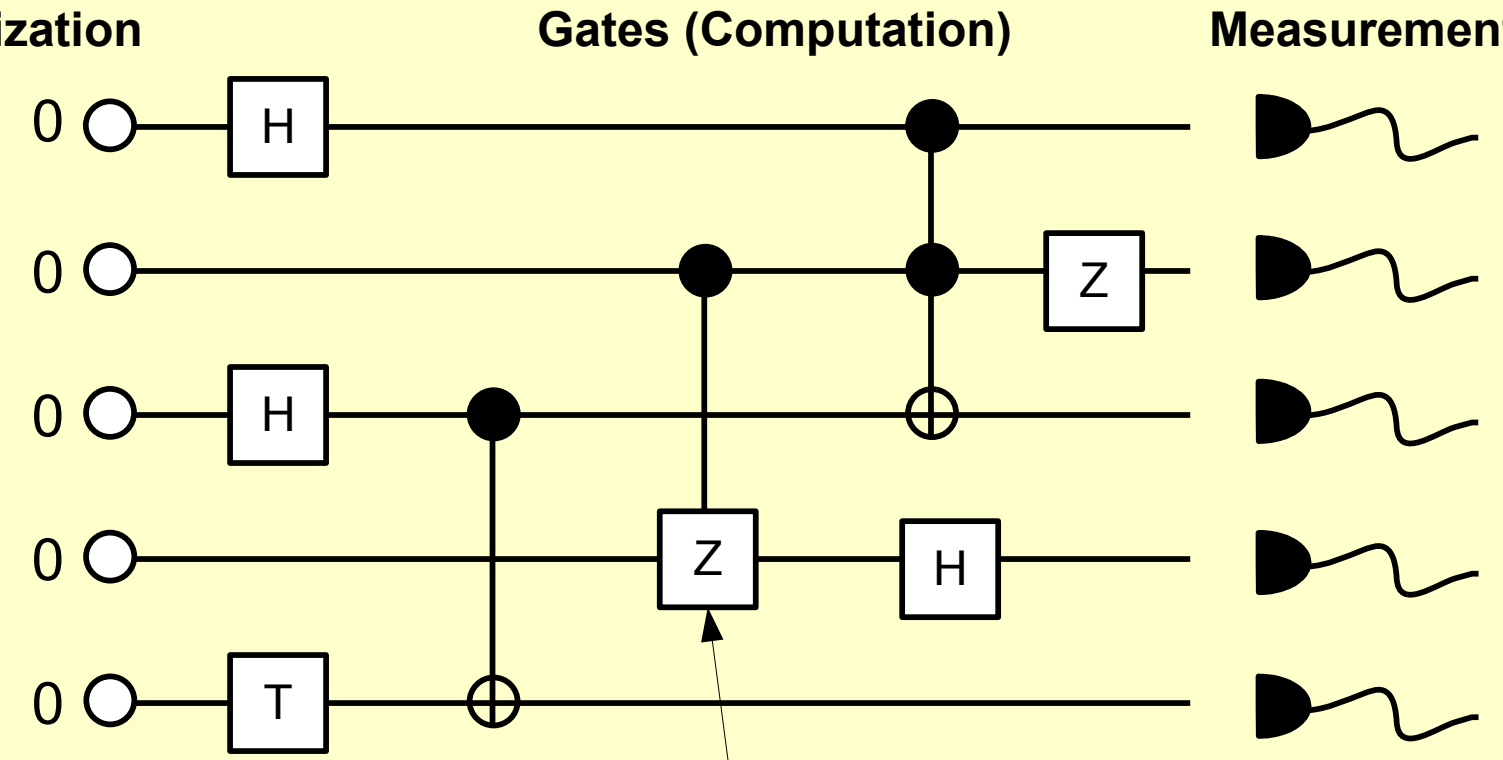
- **Version 1:** Having arbitrary local operations U(θ,n) and a CNOT, one can approximate an arbitrary N-qubit operator
  - This is called a universal set of gates
  - The approximation may be exponential in resources

- Solovay-Kitaev: an arbitrary local operation can be approximated by a sufficiently dense set of generators in SU(2) in $\log^2(1/\varepsilon)$ steps.

- **Version 2**: The Hadamard gate, phase gate and a CNOT form a universal set.

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \quad CNOT$$

Circuit model



$$U_{CNOT}$$

$$U_{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Three important ideas

- A quantum computer can do classical computations
  - Toffoli gate
  - Friedkin gate
- Reversible classical computation is not “natural”
  - Three body interactions
  - Less efficient
- New computational models available
  - Other unitaries
  - Entanglement

Toffoli gate

I1	I2	I3	O1	O2	O3
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

$$U_T \sim \frac{1}{4}(\sigma_z - 1)(\sigma_z - 1)(\sigma_x - 1) + 1$$