Lecture 10.

1) Historical introduction.

2) Circuit models of computation.

3) Quantum parallelism: the Deutsch - Josza problem.

1) Historical introduction.

A computation is ultimately performed by a physical device. Thus it is a natural question to ask what are the fundamental limitations that the laws of physics impose on a computation. An early work on such issues was that of Landauer who argued that a "bit erasure" — an irreversible process — is always accompagnied by heat dissipation. Consequently any computation using irreversible gates (AND; OR ...) will dissipate heat. But is there a fundamental principle that requires a minimum amount of heat dissipation? A negative answer to this question was put forward by Bennett, Benioff and others. More precisely (as we will see later) any irreversible computation can be made reversible, with appropriate elementary gates, provided we are willing to increase the work space.

If no heat is generated by reversible computations, as the physical support of bits becomes smaller and smaller it is natural to ask the question: what are the effects of the quantum mechanical behavior of matter on computation? Indeed if no heat is dissipated quantum mechanical coherence may become important. Does QM bring any new limitation or does it on the contrary help us?

These issues where raised and discussed by Feynman, Benioff and Manin in the early 1980's. In principle QM does not bring any new limitation, but on the contrary the superposition principle enables us to perform parallel computations thereby speeding up classical computations. This was recognized very early by Feynman who argued that classical computers cannot simulate efficiently quantum mechanical processes. The basic reason is that general quantum states involve a superposition of 2^N classical states:

$$|\psi\rangle = \sum_{b_1 \dots b_N \in \{0,1\}^N} c_{b_1 \dots b_N} \; |b_1 \dots b_N\rangle$$

A classical simulation of the evolution of $|\psi\rangle$ must perform essentially 2^N simulations for the evolution of each $|b_1 \dots b_N\rangle$. On the contrary the unitary quantum dynamics acts on $|\psi\rangle$ as a whole (or on each $|b_1 \cdots b_N\rangle$ in parallel). So physical devices performing a unitary dynamics on $|\psi\rangle$ can be viewed as devices performing a quantum computation.

Feynman developed the concept of quantum computation in the language of Hamiltonian dynamics. It turns out that this is not very practical if we are to build a universal quantum computer which perform diverse tasks.

But a classical universal computer can be represented by a circuit model built out of a given set of elementary gates acting in a recursive way on the input of the computation. Around (1985) David Deutsch showed that the same holds in the quantum case. Namely, any unitary evolution can be approximated well enough by some set of universal elementary quantum gates.

Nowadays it is the Deutsch model [the quantum circuit model] of a quantum computer that is being adopted and the subject of this chapter is to explain this model.

There is also a notion of quantum Turing machine (which is analogous to classical Turing machines) which is the natural and most convenient framework to discuss quantum complexity classes. It has been shown [Yao] that

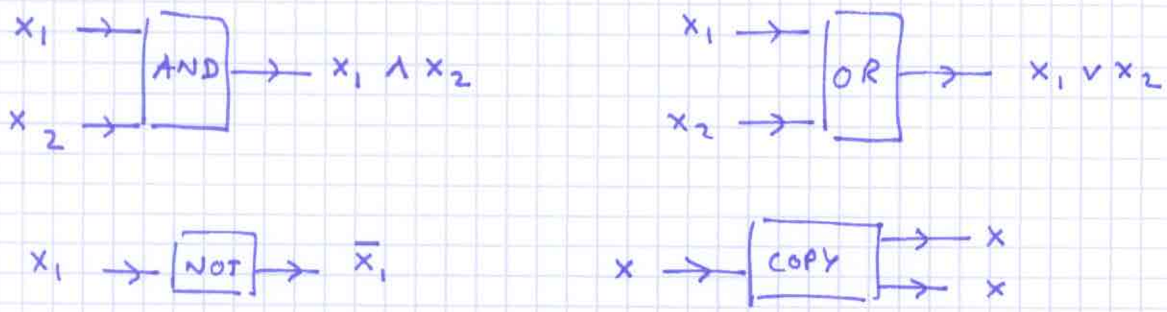
[Vazirani & Bernstein]

the Quantum Turing machine model is equivalent to the quantum circuit model. Complexity issues are briefly discussed at the end of this chapter.

2) Circuit Models of Computation.

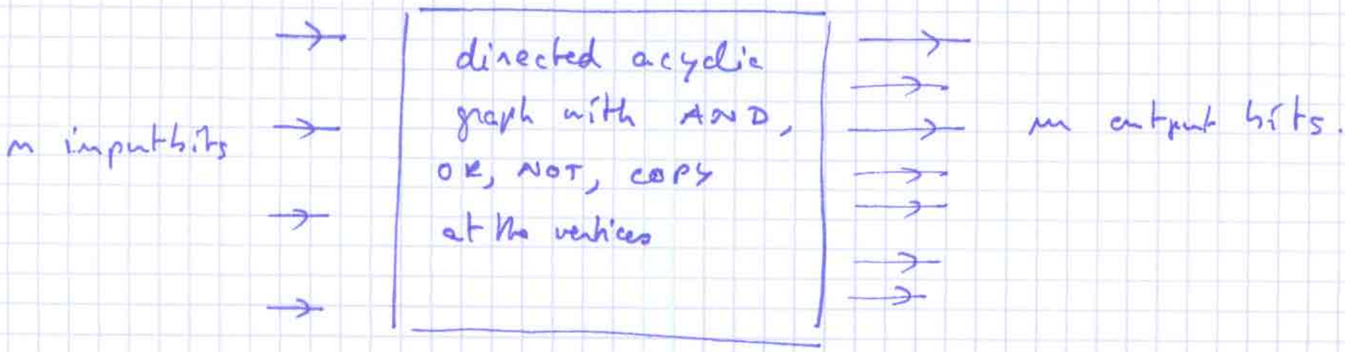
2.A) Boolean Circuits.

We begin with classical computations done with classical circuits. Consider the basic set of logical gates acting on bits $x_i \in \mathbb{F}_2 = \{0, 1\}$.

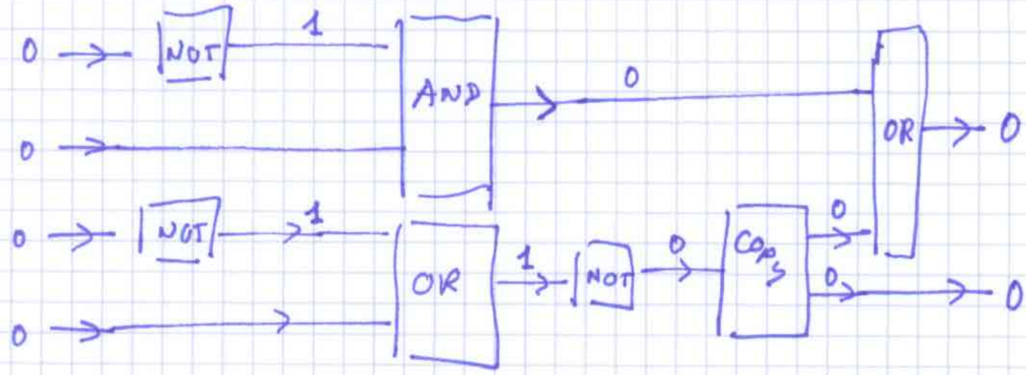


(COPY is also called FANOUT sometimes.).

Déf of Boolean Circuit: A Boolean circuit is a directed acyclic graph with n input bits and m output bits. The input can always be initialized to $(0 \ldots 0)$ because any $(x_1 \ldots x_n)$ is obtained by a series of appropriate NOT gates.



For example:



The circuit computes functions

$$f : \mathbb{F}_2^{n} \longrightarrow \mathbb{F}_2^{m}$$

A celebrated theorem of Emil Post ($\sim$ 1950) says that for any function $f : \mathbb{F}_2^{n} \to \mathbb{F}_2^{m}$ one can construct a Boolean circuit that computes it.

Theorem: For any function $f : \mathbb{F}_2^{n} \to \mathbb{F}_2^{m}$ there exists a Boolean circuit that maps inputs $(x_1 \dots x_n)$ to outputs $(y_1 \dots y_m) = f(x_1 \dots x_n)$. The Boolean circuit is constructed out of NOT, AND, OR, COPY and is a directed acyclic graph.

Remark: One says that the set of gates (NOT, AND, OR, COPY) is universal. Note that AND, OR are not reversible. This issue of reversibility is discussed later.

Proof of Theorem :

A function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be represented as m functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$, $i = 1 \dots m$. So if we can COPY the input m times we just need to show that there exists a Boolean circuit for each $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$. The problem is thus reduced to finding Boolean circuits for functions $f : \mathbb{F}_2^m \to \mathbb{F}_2$.

For each $\vec{a} = (a_1 \dots a_n) \in \mathbb{F}_2^n$ we define

$$C_{\vec{a}}(x_1 \dots x_n) = \phi_1(x_1) \wedge \phi_2(x_2) \wedge \dots \wedge \phi_n(x_n)$$

where
$$\begin{cases} \phi_i(x_i) = \bar{x}_i & \text{if } a_i = 0 \\ \phi_i(x_i) = x_i & \text{if } a_i = 1. \end{cases}$$

This is built out of AND, NOT gates only and since $\wedge$ is associative it can be done recursively (directed acyclic graph). We note that $C_{\vec{a}}(x_1 \dots x_n) = 1$ iff $(x_1 \dots x_n) = (a_1 \dots a_n)$.

Now given a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ let $\{\vec{a}^{(1)}, \dots, \vec{a}^{(K)}\}$ be the set of inputs in $\mathbb{F}_2^n$ for which f takes values 1. For all other input f takes value 0. A little thought shows that

$$f(x_1 \dots x_n) = C_{\vec{a}^{(1)}}(x_1 \dots x_n) \vee C_{\vec{a}^{(2)}}(x_1 \dots x_n) \vee \dots \vee C_{\vec{a}^{(K)}}(x_1 \dots x_n)$$

It remains to see that $\vee$ is associative and can thus be done in a recursive way. So f is compute from OR and COPY. ∎

Reversibility versus irreversibility.

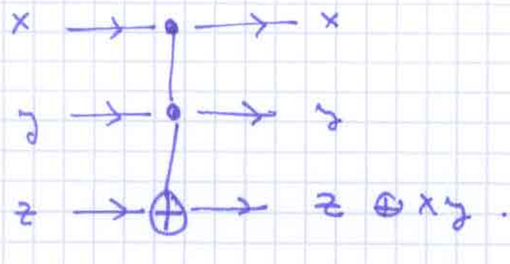
The OR gates is reversible. This means that from the output one can recover the input. However the AND, OR gates are irreversible. We will now show that any Boolean circuit can be simulated by an irreversible circuit. Moreover a universal set of irreversible gates exists.

From $f : \mathbb{F}_2^m \to \mathbb{F}_2$ we construct $\tilde{f} : \mathbb{F}_2^m \oplus \mathbb{F}_2 \to \mathbb{F}_2^m \oplus \mathbb{F}_2$ as follows:

$$\tilde{f}(x_1 \dots x_m, y) = (x_1 \dots x_m, f(x_1 \dots x_m) \oplus y)$$

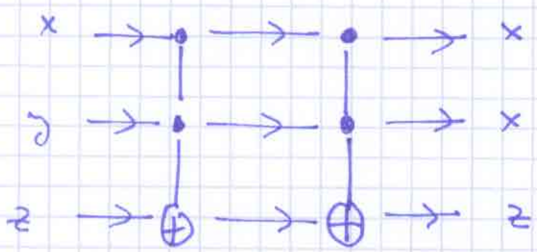
Now, $\tilde{f}$ is invertible since from $(x_1 \dots x_m, f(x_1 \dots x_m) \oplus y)$ we can recover $x_1, \dots x_m$ and then $f(x_1 \dots x_m)$ [since have a circuit for f) and then $y = (f(x_1 \dots x_m) \oplus y) \oplus f(x_1 \dots x_m)$.

Thus any f can be computed in a reversible way from the circuit for $\tilde{f}$. To compute f reversibly we start with the input $(x_1 \dots x_m, 0)$ compute $\tilde{f}(x_1 \dots x_m, 0) = (x_1 \dots x_m, f(x_1 \dots x_m))$ copy the last bit $f(x_1 \dots x_m)$ and run back the computation to get $\tilde{f}^{-1}(x_1 \dots x_m, f(x_1 \dots x_m)) = (x_1 \dots x_m, 0)$. In this way we have $f(x_1 \dots x_m)$ and the circuit is left in its initial state $(x_1 \dots x_m, 0)$. What remains to be shown is that $\tilde{f}$ can be represented by a circuit containing only irreversible elementary gates. We already know that $\tilde{f}$ can be represented by a circuit containing AND, OR, NOT and COPY. We want to replace AND, OR by a reversible gate. This can be achieved by using the 3 bit gate known as "Toffoli gate" which is a CCNOT (controlled-controlled NOT):



or $T(x,y,z) = (x, y, z \oplus xy)$.

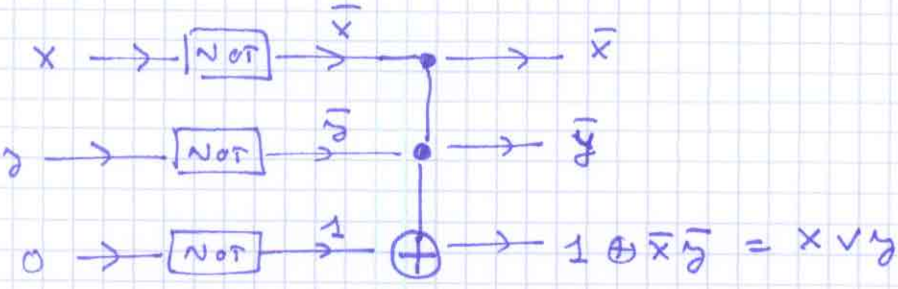
This gate flips the target bit z if both control bits x and y are equal to 1. Otherwise z is left unchanged. The Toffoli gate is reversible because:



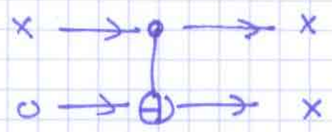
or $T^2(x,y,z) = (x,y,z)$.

If we set $z=0$ $T(x,y,0) = (x,y,xy)$ outputs $x \wedge y$ in the third bit.

Thus the AND gates can be replaced by a Toffoli gate provided we increase our work-space to have a target input bit $z=0$ and the additional outputs x & y. For the OR gate we can use



Finally the COPY gate can be replaced by



which is a CNOT gate (reversible) with the target bit set to 0 in the input. Note that the pure COPY gate, if reversed: $\left.\begin{array}{l} x \to \\ x \to \end{array}\right] \to x$ erases a bit so there is heat dissipation. This is the reason why we replace it by a CNOT.

Summarizing, we have shown that a Boolean circuit made of the universal set {AND, OR, COPY, NOT} can be simulated by a reversible circuit made of the universal set {CNOT; Toffoli; NOT}.

Remark: The set {AND, OR, COPY, NOT} involves single and two bit gates. On the other hand {CNOT, Toffoli, NOT} involves single, two bit and three bit gates. Is it possible to build a reversible circuit using only single and two bit gates?
It is possible to show that the answer to this question is no. In fact it suffices to produce a counterexample: the Toffoli gate cannot be simulated reversibly with single & two bit gates. We will see that (perhaps surprisingly) in the quantum case single & two bit gates suffice for reversible computation.

2. B) Deutsch Model of Quantum circuits.

As we will see the quantum circuits are build out of a small set of gates. For this reason it is useful to start by listing a few of the most important gates that we will encounter.

- Single Qbit gates.

* The three Pauli - gates $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$; $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$.

$|b\rangle \rightarrow \boxed{X} \rightarrow |\bar{b}\rangle$ is the quantum NOT gate.

$$\left.\begin{array}{l} |b\rangle \rightarrow \boxed{Y} \rightarrow i(-1)^b |\bar{b}\rangle \\ |b\rangle \rightarrow \boxed{Z} \rightarrow (-1)^b |b\rangle \end{array}\right\} \text{NOT gate up to a phase multiplicatic.}$$

Quantum mechanically the input can also be a coherent superposition of the states $\{|0\rangle, |1\rangle\}$. For example

$$X \left(\alpha |0\rangle + \beta |1\rangle \right) = \alpha |1\rangle + \beta |0\rangle .$$

* The Hadamard gate $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

$$|b\rangle \rightarrow \boxed{H} \rightarrow \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^b |1\rangle \right) .$$

* The "$\frac{\pi}{8}$ gate" denoted $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$.

[Up to a global phase this is multiplication by $e^{\pm i\pi/8}$.

$e^{ib\pi/4}|b\rangle$ $\quad \begin{cases} |0\rangle \to |0\rangle \\ |1\rangle \to e^{i\pi/4}|1\rangle. \end{cases}$

for superpositions: $\alpha|0\rangle + \beta|1\rangle \to \alpha|0\rangle + \beta e^{i\pi/4}|1\rangle$.

* The gate $S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$. $:\begin{cases} |0\rangle \to |0\rangle \\ |1\rangle \to i|1\rangle. \end{cases}$

An important lemma that we give here without proof is

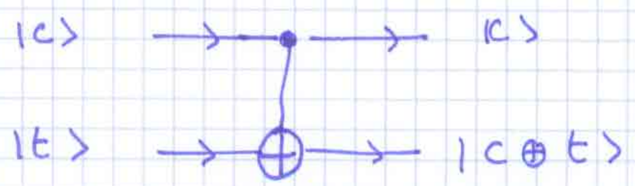
Lemma 1 [Approximation of single qbit gates by H and T].

Any unitary single bit U can be approximated to arbitrary precision δ by a concatenation of "Hadamard H" and "$\frac{\pi}{8}$-T gates".
Moreover if V is the concatenation of H and T gates approximating U and $\| U - V \| < \delta$ we need at most $O(\ln \delta)$ gates H & T [This last statement is known as the Solovay-Kitaev theorem].

Remark: The main idea of the proof is to represent U by a succession of rotations, themselves represented by Pauli-gates, themselves represented by H & T. It is not very difficult to prove that a circuit size $O(1/\delta)$ is sufficient. The Solovay-Kitaev $O(\ln \delta)$ is more difficult.

• Controlled two-bit gates.

* The CNOT gate (controlled not) is the prototypical two-bit gate:



control bit

target bit.

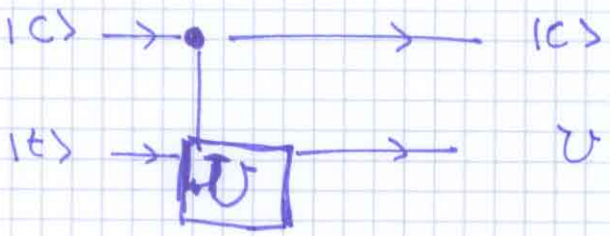
In the basis $|0\rangle \otimes |0\rangle$; $|0\rangle \otimes |1\rangle$; $|1\rangle \otimes |0\rangle$; $|1\rangle \otimes |1\rangle$

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

The matrix representation is

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

* The Controlled $-U$ gate where U is a single bit operation.

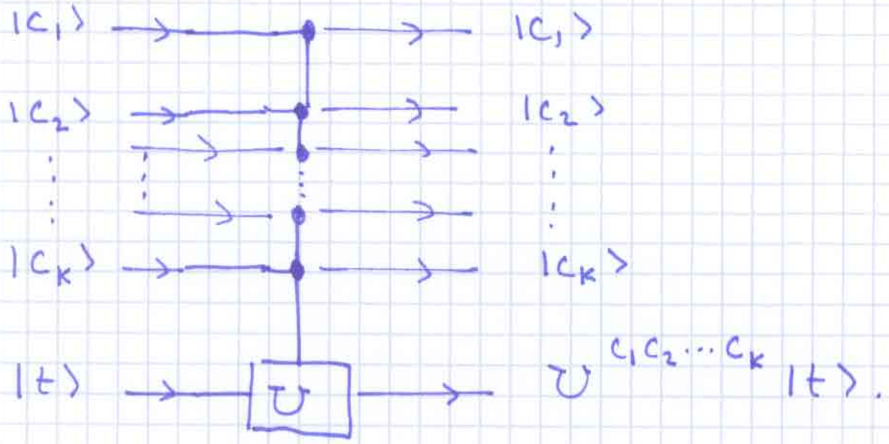


$$U^c |t\rangle . = \begin{cases} |t\rangle & \text{if } c = 0 \\ U|t\rangle & \text{if } c = 1 . \end{cases}$$

[Exercise: Build a CNOT with a controlled Z and two Hadamard H].

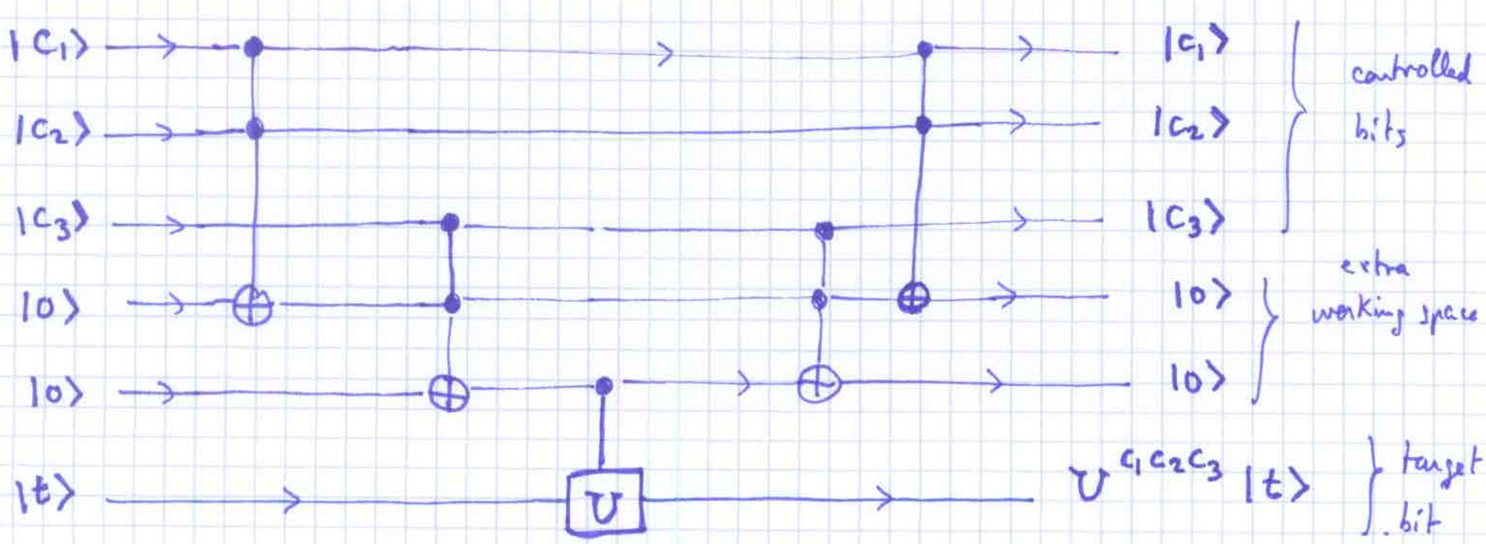
* Multibit - controlled gates.

A generalization of the previous gate is



So U acts on the target bit if and only if all control bits are set to 1.

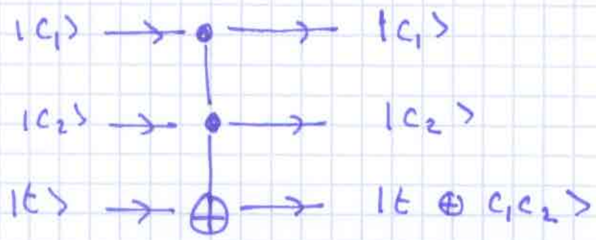
By increasing the work space this gate can be represented by a concatenation of a ~~doubly contro~~ controlled-controlled-NOT and a controlled U. Indeed:



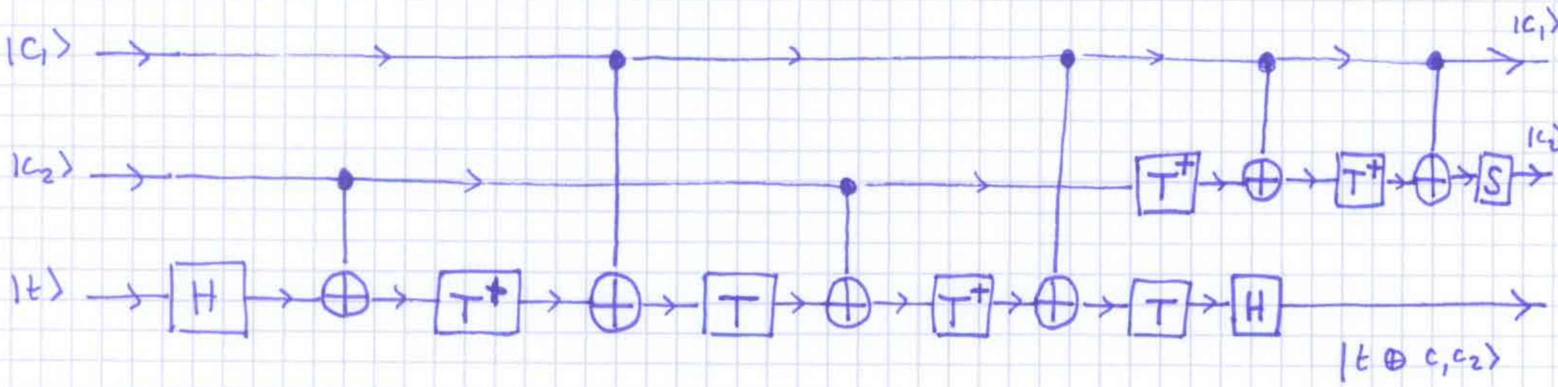
The Controlled-Controlled-NOT gate is also called Quantum Toffoli.

Remarkably it can be represented by one & two-bit gates $\{T, S, H, CNOT\}$.

Remember that classically this is not possible! The reader can check that:



is equivalent to the following circuit:



Summarizing we arrive at the following lemma:

Lemma 2.

acting on N qbits ($2^N \times 2^N$ matrix).

Any multibit-controlled gate U can be represented by the set $\{T, S, H, CNOT, U\}$.

acting on last qbit
(2x2 matrix)

- A universal set of quantum gates.

An important lemma that we give here without proof is:

Lemma 3.

Any unitary U acting on N-qbit states, i.e. states in $\underbrace{\mathbb{C}_2 \otimes \mathbb{C}_2 \cdots \otimes \mathbb{C}_2}_{N \text{ times}}$ can be decomposed as a finite product of "two level unitaries":

$$U = U^{(i_1 j_1)} \cdot U^{(i_2 j_2)} \cdots U^{(i_K j_K)}$$

where $U^{(ij)}$ acts from $\mathbb{C}_2 \otimes \cdots \otimes \mathbb{C}_2 \to \mathbb{C}_2 \otimes \cdots \otimes \mathbb{C}_2$ non trivially on spaces i and j and trivially on all other spaces.

For example of $N = 4$ we may have $U^{(14)}$

$$U^{(14)} = \begin{pmatrix} a & 0 & 0 & b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c & 0 & 0 & d \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ unitary.}$$

By implementing suitably permutations of basis vectors with CNOT.

Lemma 4.

Any two level unitary $U^{(ij)}$ acting as U on bits i & j and as the identity on all others can be implemented by a concatenation of CNOT and a multicontrolled single bit U.

Remark: One can show that there exist N-qbit unitary matrices ($2^N \times 2^N$ matrices) such that the decomposition obtained by lemmas 3 & 4 requires $O(e^N)$ gates.
For some special problems such as factoring we will see that $O(\text{poly}\, N)$ suffices!

Combining lemmas 1, 2, 3, 4 we arrive at the following basic theorem on which the Quantum Circuit Model of quantum Computation is based:

Theorem: Any $2^N \times 2^N$ unitary matrix U acting on $\underbrace{\mathbb{C}_2 \otimes \cdots \otimes \mathbb{C}_2}_{N \text{ times}}$ can be represented to arbitrary accuracy by a concatenation of the finite set of single and two-bit gates $\{T, S, H, CNOT\}$.

Remark: If the required accuracy is δ one can argue that the maximal number of gates is of the form $O(\ln\delta \cdot e^N)$. One can show that there exists unitary U for which it is not better to have $O(\ln\delta \cdot \text{poly}(N))$.

• Deutsch Model of Quantum Computation.

The basic theorem just explained justifies the following model for quantum computation:

Definition of a Quantum circuit:

a) A quantum circuit is a directed acyclic graph whose vertices are gates among the finite set $\{T, S, H, CNOT\}$. The wires "carry" single qbits $(\alpha|0\rangle + \beta|1\rangle)$.

b) The input is set to the simple tensor product state

$$|0\rangle \otimes \cdots \otimes |0\rangle$$

or more generally to $|b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_N\rangle$.

c) The output is the result of the unitary evolution operating on the input. The output is in general a state of the form

$$|\psi\rangle = \sum_{c_1 \cdots c_N} A(c_1 \cdots c_N)\ |c_1\, c_2 \cdots c_N\rangle$$

d) Finally a measurement is performed on $|\psi\rangle$ with an apparatus measuring in the basis $\{|0\rangle, |1\rangle\}$. The outcome of the measurement is "the result of the computation"

$|c_1 \cdots c_N\rangle$ obtained with probability $|A(c_1 \cdots c_N)|^2$.

A few remarks about this model are in order :

* Acting on "Qutrits" instead of "Qbits" would not change anything fundamental (e.g the size or complexity of the circuit).

* Performing measurements at intermediate stages instead of at the end does not change anything.

* Performing measurements in another basis simply amounts to first unitarily rotate the basis so this can be viewed as an adjunction to the circuit and finaly does not change anything.

* Other sets of universal gates exist. It may be surprising that in the classical case three bit gates are needed whereas this is not the case for quantum computation. But from a more physical point of view this is not surprising because the classical three bit gates can be viewed as an effect of "two-body interaction" [see Billiard-Ball-Model & Fredkin gate].

* A quantum computation is reversible as long as the measurement has not been performed.

* A reversible classical computation can be represented by a unitary operator. Indeed

$$\tilde{f}(x_1 \ldots x_N, y) = (x_1 \ldots x_N, y \oplus f(x_1 \ldots x_N))$$

induces the unitary

$$U_f \, |x_1 \ldots x_N, y\rangle = |x_1 \ldots x_N ; y \oplus f(x_1 \ldots x_N)\rangle .$$

That U_f is unitary is easily checked by checking that it preserves the scalar product.

Thus any classical reversible computation is included in the model of quantum computation.

* The power of quantum computation comes from the simultaneous acting of the unitary evolution on all strings $|c_1 \ldots c_N\rangle$. The complexity of the calculation is given by the size of the circuit. Since the result is obtained with some probability, typically one must repeat a certain number of times the computation to get a result with high probability (hopefully). This repetition may add to the complexity.