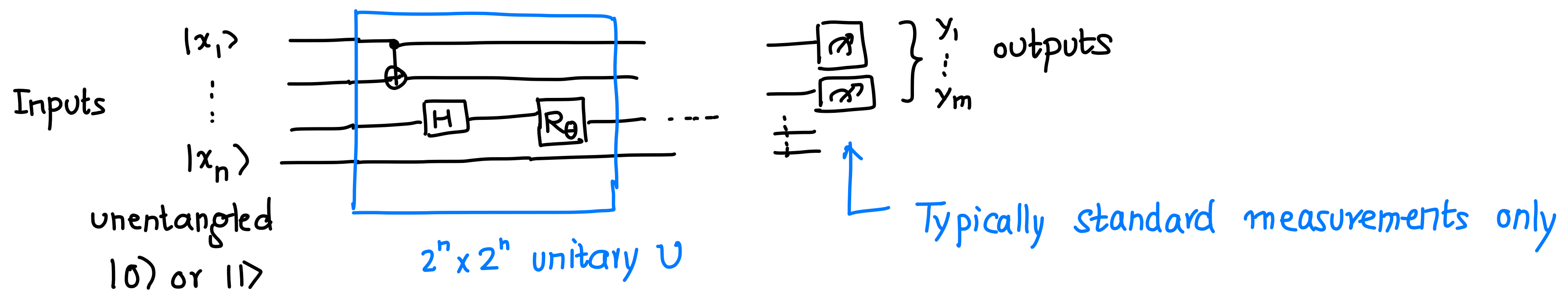
LECTURE 10   February 20th, 2025

PART II Fundamental Quantum Algorithms

$\boxed{\text{Today}}$ Quantum Circuits

Quantum Circuit Model


Inputs
$|x_1\rangle$
$|x_n\rangle$
unentangled
$|0\rangle$ or $|1\rangle$
H
R_θ
$2^n \times 2^n$ unitary U
y_1
y_m
outputs
Typically standard measurements only

Quantum circuit C computes $F: \{0,1\}^m \to \{0,1\}$ if

$\forall$ inputs x, $\underset{\text{measurement}}{\mathbb{P}}[\,C(x) \neq F(x)\,] \leq$ small

Typically we measure only at the end since we can add extra qubits to defer the measurement $\longrightarrow$ "Principle of Deferred Measurement"
Will be on HW 3

This also is nice since we don't have to deal with mixed states since there are no intermediate measurements

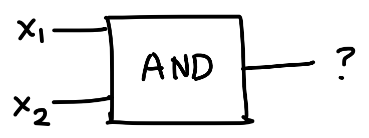
Next few lectures

Q1: Does $\exists$ F which quantum circuits can compute much more efficiently than classical ones?

Q2: Can quantum circuits simulate classical circuits?

Q3: Does the exact quantum gates matter?

Let's talk about Q2 first! Can a Q.C. compute



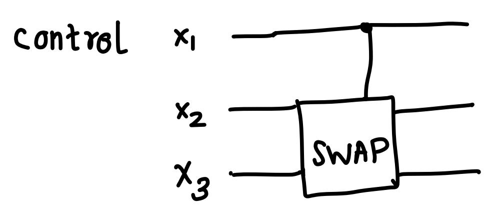
Recall, Q. gates are unitary $UU^\dagger = I \Rightarrow U^{-1} = U^\dagger$ inverse exists

In particular, all quantum gates are reversible meaning if you know the output you can figure out the input

Not true for AND! True for NOT gate!

This topic of reversible computing was studied by physicists in the 1960s-70s
They were interested in energy efficiency

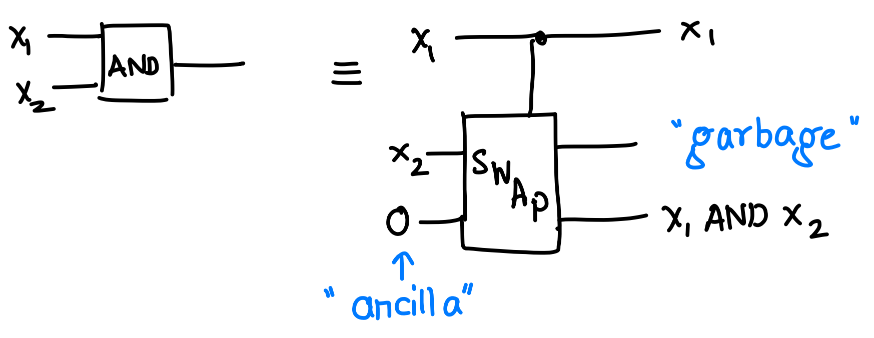
To make AND gate reversible, we need more qubits and one reversible gate CSWAP
(Controlled-SWAP)



SWAPS x_2 and x_3 if $x_1 = 1$
does nothing if $x_1 = 0$

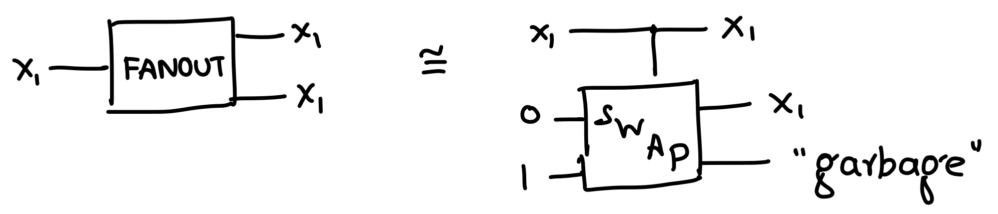
Gives a permutation of $\{|000\rangle, \dots, |111\rangle\} \Rightarrow$ Unitary gate
It's it own inverse in fact!

How to implement AND gate?



Similarly one can implement OR gate using CSWAP

How to implement FANOUT gate?



Corollary Any classical circuit computing $F : \{0,1\}^n \to \{0,1\}^m$ can be efficiently converted to a reversible (and hence quantum) circuit

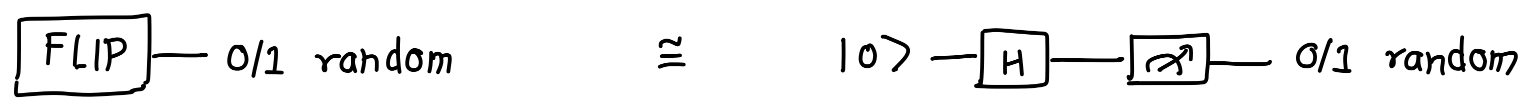
$$QC : \{0,1\}^{n+a} \longrightarrow \{0,1\}^{m+g} \qquad n+a = m+g$$

a = # of ancilla bits ← typically initialized to all 0's
g = # of garbage bits

$$(\underbrace{x}_{n}, \underbrace{0,\dots,0}_{a}) \xrightarrow{QC} (\underbrace{F(x)}_{m}, \underbrace{g(x)}_{g})$$

# gates in quantum circuit ≤ (# gates in classical circuit) · constant

How about probabilistic computing ?



Can defer measurement till the end using more ancillas

$\boxed{\text{Summary}}$ Quantum circuits are at least as powerful as probabilistic circuits

Dealing with unwanted garbage bits

Suppose we want to use a classical circuit as a subroutine in a quantum circuit

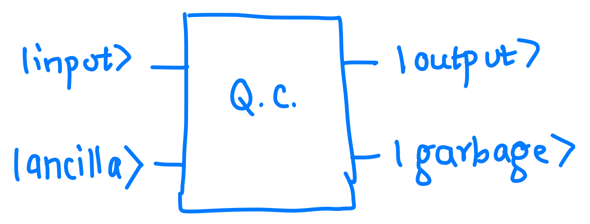
$$F : \{0,1\}^n \longrightarrow \{0,1\}^m$$

If we implement the classical circuit reversibly i.e.

$$|x_1, \ldots x_n\rangle \underbrace{|00\cdots0\rangle}_{\text{ancilla}} \longrightarrow |F(x)\rangle \underbrace{|G(x)\rangle}_{\text{garbage}}$$

Puzzle (in-class) Suppose we have a reversible circuit $(P, Q, 0\cdots0) \longrightarrow (P\cdot Q, \text{garbage})$
Why can't we reverse it & get an efficient algorithm for factoring?

If things are in superposition, garbage is difficult to deal with

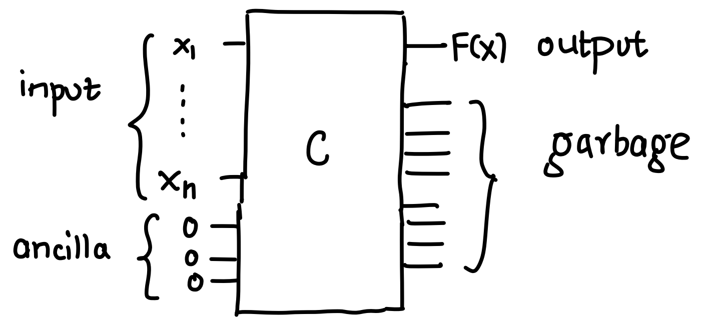


If garbage is entangled with output we can't easily get rid of it
Also, makes it difficult to use it as a subroutine

So, often we want to remove garbage

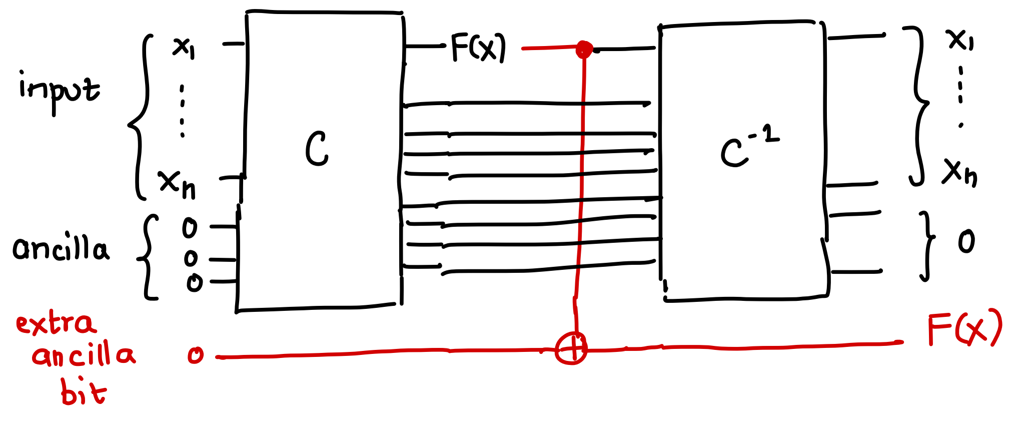
Uncomputing Garbage [Bennett '80s]

Say $F : \{0,1\}^n \rightarrow \{0,1\}$ computed reversibly



$$(x_1, \ldots x_n, 0\cdots0) \longrightarrow (F(x), g(x))$$

Trick: copy the answer somewhere and undo the computation



$$(x_1, \dots x_n, 0 \dots 0, 0) \longrightarrow (x_1, \dots x_n, 0 \dots 0, F(x))$$

If we initialize the extra ancilla bit to 1

$$(x_1, \dots x_n, 0 \dots 0, 1) \longrightarrow (x_1, \dots x_n, 0 \dots 0, 1 \oplus F(x))$$

So, overall we get a quantum circuit implementing

$$|x_1, \dots x_n\rangle \underbrace{|0 \dots 0\rangle}\, |y\rangle \longrightarrow |x_1, \dots x_n\rangle \underbrace{|0 \dots 0\rangle}\, |y \oplus F(x)\rangle$$

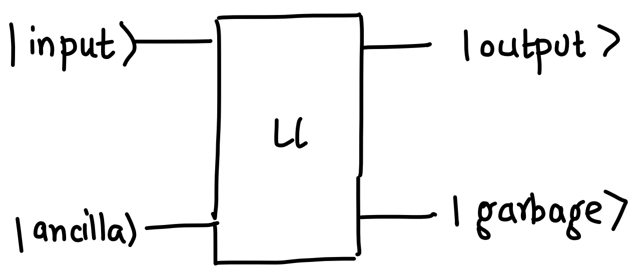
We can even throw these away now
since they are unentangled!

$\boxed{\text{Definition}}$ A quantum circuit implements $F: \{0,1\}^n \to \{0,1\}^m$ if it computes it as

$$|x_1 \dots x_n\rangle \, |\underset{\substack{\| \\ y_1 \dots y_m}}{y}\rangle \longrightarrow |x_1, \dots x_n\rangle \, |y \oplus F(x)\rangle$$

Let's try to answer Q3 now! What kind of gates do we need for a quantum circuit?

Consider a general quantum circuit which implements a unitary on $n+a$ qubits
$2^{n+a} \times 2^{n+a}$ unitary



Extremely big unitary & difficult to build in general

Are one or two qubit gates enough to implement U?
YES!

Universal Gate Set

$\{H, S, CNOT, T\}$ is universal for quantum computing $\rightarrow$ similar to how AND, OR, NOT is universal for classical circuits

where $S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ Phase gate

$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$

Since $\exists$ uncountably many unitaries, here we can only get ε-approximation meaning the output state of the quantum circuit using $\{H, S, CNOT, T\}$ gate is ε-close on any input state to the real output state:

$$\| U|\psi\rangle - \underset{\text{universal quantum circuit}}{C}|\psi\rangle \| \le \varepsilon \longrightarrow \text{All measurement outcomes will only differ by } \varepsilon$$

One wants to choose $\varepsilon = \frac{1}{\text{poly}(n)}$ or even 2^{-n} where $n = \#$ qubits

So, dependence on ε matters a lot! Does the gate set matter for this?

Solovay-Kitaev theorem says that using a different universal gate set can only reduce the number of gates by a factor of $\log 1/\varepsilon$ i.e.

$$\#\text{ Gates with universal gate set } G \lesssim \#\text{ Gates with } \{H, CNOT, S, T\} \cdot \log\frac{1}{\varepsilon}$$

Analog of Shannon's Theorem for Quantum Setting

Any unitary U on n qubits can be approximated, with any universal gate set, to ε-precision using

$$O(4^n \cdot \log^{c} 1/\varepsilon) \text{ gates}$$

Most unitaries also require (roughly) this many gates.

Now we can start with answering Q1 — can quantum circuits be more efficient in computing a function?

Deutsch's Algorithm

This was arguably the first ever quantum algorithm!

What is the problem?

Given a function $f: \{0,1\} \to \{0,1\}$, determine if it is constant or balanced

constant: $f(0) = f(1)$
$$\begin{tikzcd} x & f(x) \\ 0 \arrow[r] & 0 \\ 1 \arrow[ur] & \end{tikzcd} \quad \text{or} \quad \begin{tikzcd} x & f(x) \\ 0 \arrow[dr] & \\ 1 \arrow[r] & 1 \end{tikzcd}$$

balanced: $f(0) \neq f(1)$
$$\begin{tikzcd} 0 \arrow[r] & 0 \\ 1 \arrow[r] & 1 \end{tikzcd} \quad \text{or} \quad \begin{tikzcd} 0 \arrow[dr] & 0 \\ 1 \arrow[ur] & 1 \end{tikzcd}$$

How's the function given to you? Only "black-box" or "query" access

$$x \;-\!\boxed{f}\!-\; f(x)$$

One is given an API which allows you to get the value of f on any input. This is the only way to access f. In particular, you can't see the code of how f is implemented.

How many queries are needed classically to solve the problem (with no error)?

▷ 2 queries are necessary and sufficient

Deutsch's algorithm does it in a single query ← $2\times$ speedup over classical algorithms

Quantum Queries

What does it mean to query a function as a black-box?

$$|x\rangle \;-\!\boxed{f}\!-\; |f(x)\rangle$$ is not reversible and hence not unitary in general
e.g. $f = \text{AND}$

We need to implement it reversibly as we have seen:

$$\underbrace{|x\rangle}_{\text{input}}\underbrace{|y\rangle}_{\text{output}} \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$$ Let us call this unitary U_f

A quantum algorithm can query in superposition

It is easy to see that 2 queries are still required if we only use a classical reversible circuit

NEXT TIME Deutsch's & Simon's Algorithm