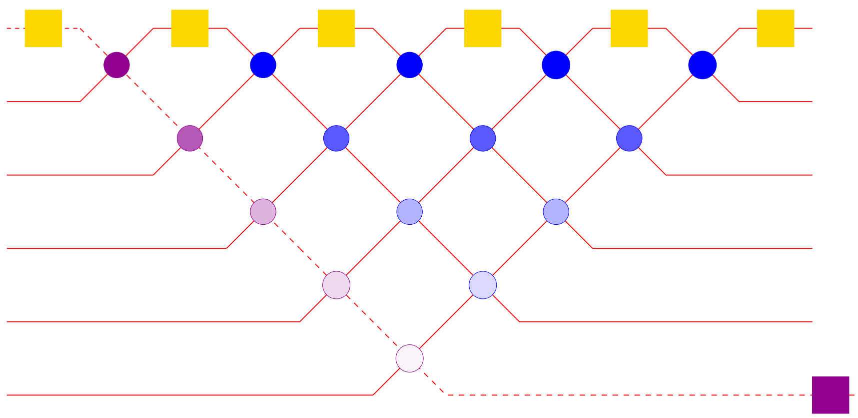
What Has *Quantum Mechanics* to Do With *Factoring?*

Things I wish they had told me about Peter Shor's algorithm



Question:

What has quantum mechanics to do with factoring?

Answer:

Nothing!

Question:

What has quantum mechanics to do with factoring?

Answer:

Nothing!

But quantum mechanics has a lot to do with waves.

And waves have a lot to do with periodicity.

And being good at diagnosing periodicity has a lot to do with factoring.

Case of cryptographic interest:

Factoring $N = pq$, where
p and q are enormous (e.g. 300 digit) primes.

Closely tied to the ability to find the period
of a^x modulo N for integers a
that share no factors with N.

Periodic functions a^x in modular arithmetic

$a \pmod N$ = *remainder when a divided by N*

$$5 = 5 \pmod 7 \qquad 5^2 = 4 \pmod 7$$
$$5^3 = 6 \pmod 7 \qquad 5^4 = 2 \pmod 7$$
$$5^5 = 3 \pmod 7 \qquad 5^6 = 1 \pmod 7$$

$5^x \pmod 7$ *is periodic with period* 6

$$4 = 4 \pmod 7$$
$$4^2 = 2 \pmod 7$$
$$4^3 = 1 \pmod 7$$

$4^x \pmod 7$ *is periodic with period* 3

$$6 = 6 \pmod 7$$
$$6^2 = 1 \pmod 7$$

$6^x \pmod 7$ *is periodic with period* 2

$2^x \pmod 7$ *has period* 3

$3^x \pmod 7$ *has period* 6

Periods mod N, where $N = pq$, and
p and q are enormous primes.

If a shares no factors with N then
$a^s \equiv 1 \pmod{N}$ *for some integer s,*

For there are only N different mod N numbers,
so there must be x and $y > x$ with $a^y = a^x \pmod{N}$.

Then $a^x(a^s - 1)$ is a multiple of N, $y = x + s$.

Since a shares no factors with N, neither does a^x,
so $a^s - 1$ must be multiple of N:

$$a^s = 1 \pmod{N}$$

If r is the *smallest* integer with $a^r \equiv 1 \pmod{N}$
then $a^x \pmod{N}$ is a periodic function of x with period r.

Digression: *The reason all periods modulo 7 divide 6:*

If p is prime all $a < p$ share no factors with p,
so $a^r = 1 \pmod p$ for some (smallest positive) r
$\Rightarrow a$ has an inverse $\pmod p$.

So the $p-1$ integers, $1, 2, \ldots\ p-1$
are a *group* under multiplication $\pmod p$.

The r distinct powers of a are a *subroup* of that group.
And *the number of members of any subgroup*
divides the number of members of the whole group.

Further digression: *Periods modulo $N = pq$ divide $(p-1)(q-1)$*

There are $pq - 1$ integers less than pq.
Among them are $p - 1$ multiples of q,
and another $q - 1$ multiples of p.

So the number of integers $a < pq$
that share no factors with pq is
$(pq-1)-(p-1)-(q-1) = pq-p-q+1 = (p-1)(q-1)$.

These $(p - 1)(q - 1)$ integers are a group
under multiplication modulo pq.

The r distinct powers of a are a subgroup of that group.
And the number of members r of that subgroup
divides the number of members $(p-1)(q-1)$ of the group.

Back to business:

How to factor the product
of two enormous primes, $N = pq$,
using a good period-finding machine
(e.g. a quantum computer).

Pick a random integer a.
(It is astronomically unlikely to be multiple of p or q.)
Use the period-finding machine to get
the smallest r with $a^r = 1 \pmod{N}$.

Pray for two pieces of good luck.

Quantum computer gives smallest r with $a^r - 1$ divisible by $N = pq$

First piece of luck: r even.

Then $(a^{r/2} - 1)(a^{r/2} + 1)$ divisible by N.

but $a^{r/2} - 1$ is *not* divisible by N

(since r is *smallest* number with $a^r - 1$ divisible by N.)

Second piece of luck: $a^{r/2} + 1$ is also not divisible by N.

Then product of $a^{r/2} - 1$ and $a^{r/2} + 1$ is divisible by both p and q although neither factor is divisible by both.

Since p, q primes, one factor divisible by p and other divisible by q.

So p is greatest common divisor of N and $a^{r/2} - 1$

and q is greatest common divisor of N and $a^{r/2} + 1$

FINISHED!

Finished, because:

1. Finding the greatest common divisor of two integers can be done by anybody who can do long division* using a simple and efficient procedure that was know to the ancient Greeks.

2. If a is picked at random, a two-hour argument** shows that the probability is at least 50% that both pieces of luck will hold.

---

* *New York Times*, November 14, 2006: "When my oldest child, an A-plus stellar student, was in sixth grade, I realized he had no idea, no idea at all, how to do long division," Ms. Backman said, "so I went to school and talked to the teacher, who said, 'We don't teach long division; it stifles their creativity.' "

** N. David Mermin, *Introduction to Quantum Computer Science*, Appendix M, Cambridge University Press, August, 2007.

Incorrect (but amazing):

[After the quantum computation] the solutions — the factors of the number being analyzed — will all be in superposition.

— George Johnson, *A Shortcut Through Time.*

[A quantum computer will] try out all the possible factors simultaneously, in superposition, then collapse to reveal the answer.

— *Ibid.*

Correct (but unexciting):

A quantum computer is efficient at factoring because it is efficient at period-finding.

BUT WHAT'S SO HARD ABOUT PERIOD-FINDING?

Given a graph of $\sin(kx)$ it's easy to find the period $2\pi/k$. Since no value repeats inside a period, $a^x \pmod{N}$ is even simpler.

BUT WHAT'S SO HARD ABOUT PERIOD-FINDING?

Given a graph of $\sin(kx)$ it's easy to find the period $2\pi/k$. Since no value repeats inside a period, $a^x \pmod N$ is even simpler.

What makes it hard:

Within a period, unlike the smooth, continuous $\sin(kx)$, the function $a^x \pmod N$ looks like random noise.

Nothing in a list of r consecutive values gives a *hint* that the next one will be the same as the first.

PERIOD FINDING WITH A QUANTUM COMPUTER

Represent n bit number

$$x = x_0 + 2x_1 + 4x_2 + \cdots + 2^{n-1}x_{n-1}$$

(each x_j is 0 or 1)

by product of states $|0\rangle$ and $|1\rangle$ of n 2-state systems:

$$|x\rangle = |x_{n-1}\rangle \cdots |x_1\rangle|x_0\rangle$$

Qbits

Qbits, not *qubits* because:

1. Classical two state systems are *Cbits* (not *clbits*)
2. Ear cleaners are *Qtips* (not *Qutips*)
3. Dirac wrote about *q-numbers* (not *qunumbers*)

(*q-bit* awkward: *2-Qbit gate* OK;
2-q-bit gate unreadable.)

More terminology:

Set of states $|x\rangle = |x_{n-1}\rangle \cdots |x_1\rangle|x_0\rangle$
called the *computational basis.*

Better term: *classical basis.*

Remark:

Because it *is* a basis, linear transformations on Qbits can be defined by specifying their action on the classical basis.

Represent function f taking n-bit to m-bit integers by a linear, norm-preserving (unitary) transformation $\mathbf{U}_f$ acting on n-Qbit *input register* and m-Qbit *output register*:

input register

$$\mathbf{U}_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle.$$

output register

(More generally, $\mathbf{U}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$.

$y \oplus z =$ bitwise modulo 2 sum: $1010 \oplus 0111 = 1101$.)

QUANTUM PARALLELISM

$$\mathbf{U}_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle$$

Put input register into superposition of all possible inputs:

$$|\phi\rangle = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le x<2^n} |x\rangle$$

$$= \tfrac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)\cdots\tfrac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big).^*$$

Applying linear $\mathbf{U}_f$ to input and output registers gives

$$\mathbf{U}_f\big(|\phi\rangle|0\rangle\big) = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le x<2^n} |x\rangle|f(x)\rangle.$$

---

*e.g. $\big(|0\rangle + |1\rangle\big)\big(|0\rangle + |1\rangle\big) = |0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle$

QUANTUM PARALLELISM

$$\mathbf{U}_f\big(|\phi\rangle|0\rangle\big) = \big(\tfrac{1}{\sqrt{2}}\big)^n \sum_{0\le x<2^n} |x\rangle|f(x)\rangle.$$

Question:

Has *one* invocation of $\mathbf{U}_f$ computed $f(x)$ for *all* x?

QUANTUM PARALLELISM

$$\mathbf{U}_f\big(|\phi\rangle|0\rangle\big) = \big(\tfrac{1}{\sqrt{2}}\big)^n \sum_{0\le x<2^n} |x\rangle|f(x)\rangle.$$

Question:

Has *one* invocation of $\mathbf{U}_f$ computed $f(x)$ for *all* x?

Answer:

No. Given a single system in an unknown state, there is no way to learn what that state is.

Information is acquired *only* through measurement.
Direct measurement of input register gives random x_0;
Direct measurement of output register then gives $f(x_0)$.

QUANTUM PARALLELISM

$$\mathbf{U}_f\big(|\phi\rangle|0\rangle\big) = \big(\tfrac{1}{\sqrt{2}}\big)^n \sum_{0\le x<2^n} |x\rangle|f(x)\rangle.$$

Special form when $f(x) = a^x \pmod N$:

$$\sum_{0\le x<2^n} |x\rangle|a^x\rangle = \sum_{0\le x<r} \Big(|x\rangle + |x+r\rangle + |x+2r\rangle + \cdots\Big)|a^x\rangle$$

THE QUANTUM FOURIER TRANSFORM (QFT)

$$\mathbf{V}_{FT}|x\rangle = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le y<2^n} e^{2\pi i x y/2^n}|y\rangle$$

Acting on superpositions, $\mathbf{V}_{FT}$ Fourier-transforms amplitudes:

$$\mathbf{V}_{FT}\sum \alpha(x)|x\rangle = \sum \beta(x)|x\rangle$$

$$\beta(x) = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le z<2^n} e^{2\pi i x z/2^n}\alpha(z)$$

If α has period r, as in $|x\rangle + |x+r\rangle + |x+2r\rangle + \cdots$, then β is sharply peaked at integral multiples of $2^n/r$.

HO-HUM!

$\mathbf{V}_{FT}$ is *boring*:

1. Just familiar transformation from position to momentum representation.
2. Everybody knows Fourier transform sharply peaked at multiples of inverse period.

But $\mathbf{V}_{FT}$ is *not* ho-humish because:

1. x has nothing to do with position, real or conceptual. x is arithmetically useful but physically meaningless:
$$x = x_0 + 2x_1 + 4x_2 + 8x_3 + \cdots,$$
where $|x_j\rangle = |0\rangle$ or $|1\rangle$ is state of j-th 2-state system.
2. *Sharp* means sharp compared with resolution of apparatus. The period r is hundreds of digits long. Error in r of 1 in 10^{10} messes up almost every digit.

Using the QFT : $\mathbf{V}_{FT}|x\rangle = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{0\le y<2^n} e^{2\pi ixy/2^n}|y\rangle$

$$\mathbf{V}_{FT}\left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le x<r} \Big(|x\rangle + |x+r\rangle + |x+2r\rangle + \cdots\Big)|a^x\rangle =$$

$$= \left(\tfrac{1}{2}\right)^n \sum_{0\le y<2^n} \Big(1+\alpha+\alpha^2+\alpha^3+\cdots\Big)|y\rangle \sum_{0\le x<r} e^{2\pi ixy/2^n}|a^x\rangle,$$

$$\alpha = \exp\Big(2\pi iy/(2^n/r)\Big).$$

Sum of phases α sharply peaked at values of y within $\frac{1}{2}$ of integral multiples of $2^n/r$.

Question: How sharply peaked?

Answer: Probability that measurement of input register gives such a value of y exceeds 40%.

Significant ($> 40\%$) chance of getting integer y as close as possible to (i.e. within $\frac{1}{2}$ of) $j(2^n/r)$ for some (more or less) random integer j.

Then $y/2^n$ is within $1/2^{n+1}$ of j/r.

Question: Does this pin down unique rational number j/r?

Answer: It depends.

Suppose second candidate, j'/r' with $j'/r' \neq j/r$.

$$\left|\frac{j'}{r'} - \frac{j}{r}\right| = \frac{|j'r - jr'|}{rr'} \geq \frac{1}{rr'} \geq \frac{1}{N^2}$$

So answer is Yes, if $2^n > N^2$.

Input register must be large enough to represent N^2.

Then have 40% chance of learning a *divisor* r_0 of r.

(r_0 is r divided by factors it shares with (random) j)

A comment:

When $N = pq$, easy to show* period r necessarily $< N/2$. So

$$\left|\frac{j'}{r'} - \frac{j}{r}\right| > \frac{4}{N^2}$$

and therefore don't need y *as close as possible* to integral multiple of $2^n/r$.

Second, third, or fourth closest do just as well.

Raises probability of learning divisor of r from 40% to 90%.

---

* $a^{p-1} = 1 \pmod{p} \;\Rightarrow\; a^{(p-1)(q-1)/2} = 1 \pmod{p}$,
$a^{q-1} = 1 \pmod{q} \;\Rightarrow\; a^{(q-1)(p-1)/2} = 1 \pmod{q}$,
$\Rightarrow\; a^{(p-1)(q-1)/2} = 1 \pmod{pq}$.

Another comment:

Should the period r be 2^m, then $2^n/r$ is itself an integer, and probability of y being multiple of that integer is easily shown to be 1, even if input register contains just a single period.

A pathologically easy case.

Question: When are all periods r powers of 2?
Answer: When p and q are both of form $2^j + 1$.
(Periods are divisors of $(p-1)(q-1)$.)

Therefore factoring $15 = (2 + 1) \times (4 + 1)$
— i.e. finding periods modulo 15 —
is not a serious demonstration of Shor's algorithm.

SOME NEAT THINGS ABOUT THE QFT

$$\mathbf{V}_{FT}|x\rangle = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le y<2^n} e^{2\pi i xy/2^n}|y\rangle$$

1. Constructed entirely out of 1-Qbit and 2-Qbit gates.

2. Number of gates (and therefore time) grows only as n^2.

3. With just *one* application,

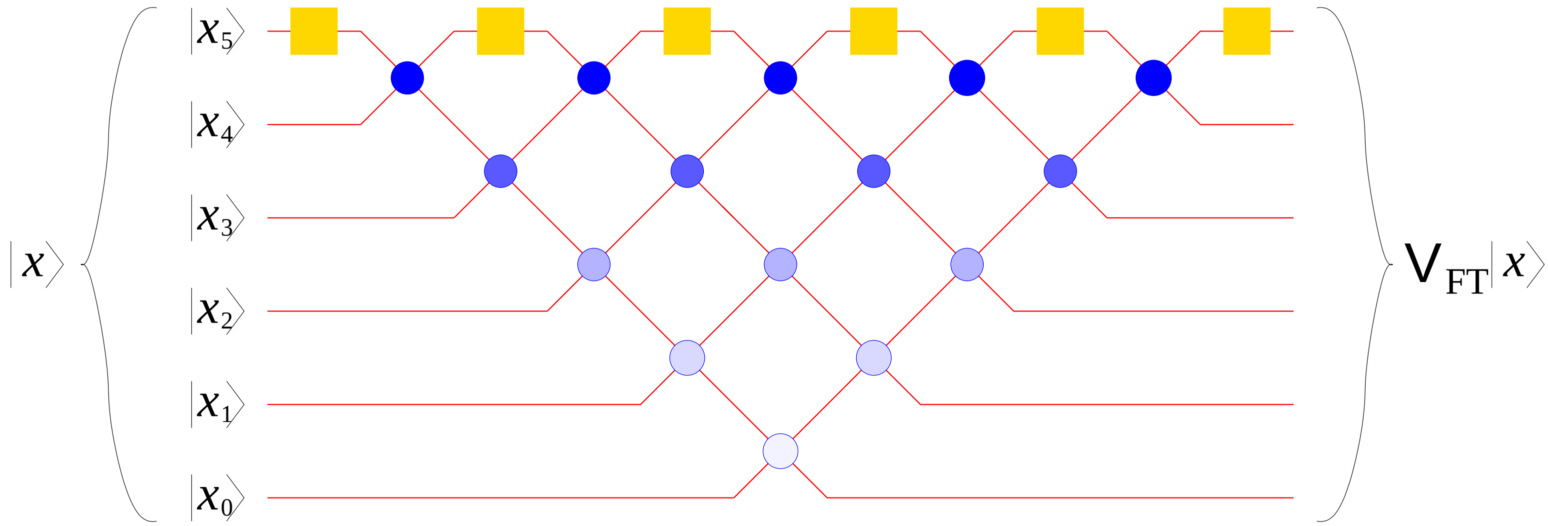
$$\sum \alpha(x)|x\rangle \longrightarrow \sum \beta(x)|x\rangle,$$

$$\beta(x) = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0\le z<2^n} e^{2\pi i xz/2^n}\alpha(z)$$

In *classical* "Fast Fourier Transform" time grows as $n2^n$.

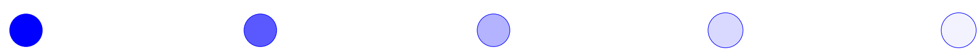
But (as usual) classical FFT gives all the $\beta(x)$,
While QFT only gives $\sum \beta(x)|x\rangle$.

Can't learn any $\beta(x)$ from one application of QFT.
But *can* get powerful clues about *period* of $\alpha(x)$.



$$\left.\begin{matrix}|0\rangle\\|1\rangle\end{matrix}\right\}$$

$$\left\{\begin{matrix}\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\\ \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)\end{matrix}\right.$$



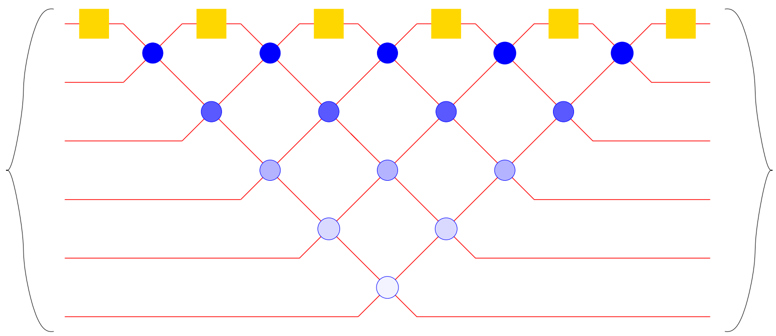
$e^{\pi i \mathbf{nn}'/2}$ $\quad$ $e^{\pi i \mathbf{nn}'/4}$ $\quad$ $e^{\pi i \mathbf{nn}'/8}$ $\quad$ $e^{\pi i \mathbf{nn}'/16}$ $\quad$ $e^{\pi i \mathbf{nn}'/32}$

$$|0\rangle|0\rangle,\ |0\rangle|1\rangle,\ |1\rangle|0\rangle \text{ invariant;}$$

$$|1\rangle|1\rangle \longrightarrow e^{\pi i/2^j}|1\rangle|1\rangle$$

ACTION OF QFT ON SUPERPOSITION

$\sum \alpha(x)|x\rangle$

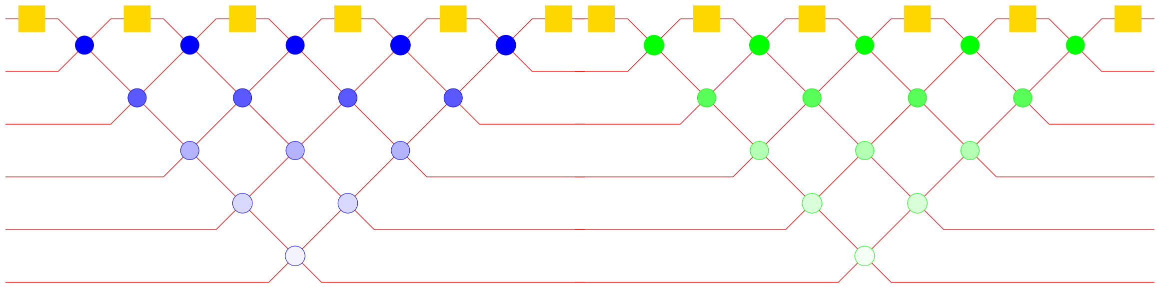


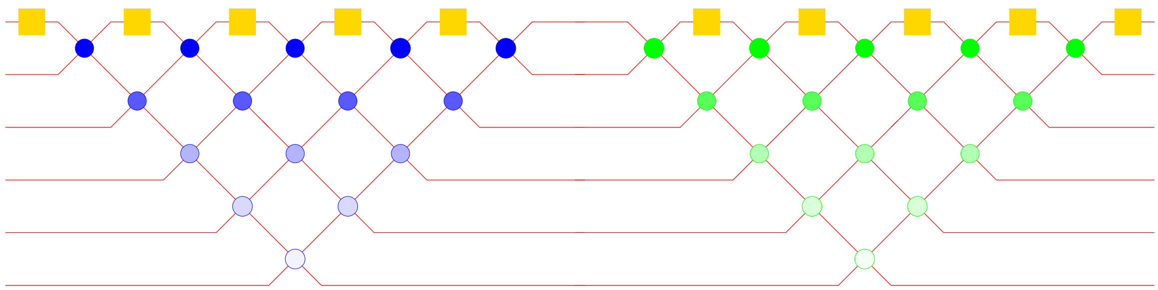
$\sum \beta(x)|x\rangle$

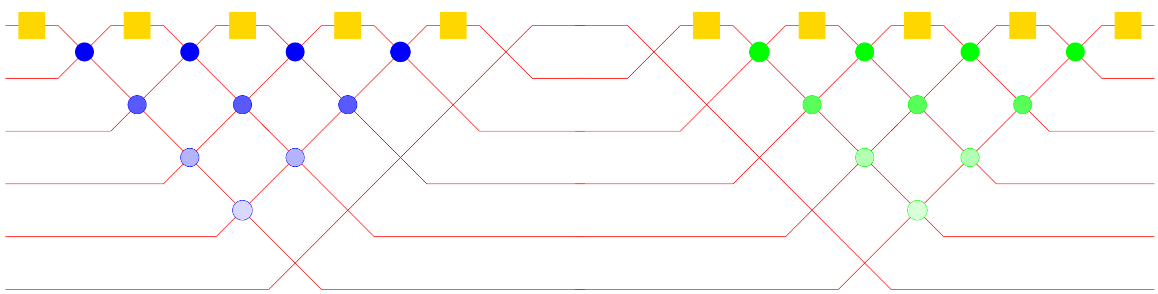
$$\beta(x) = \sum_y \alpha(y) e^{2\pi i x y / 2^6}$$

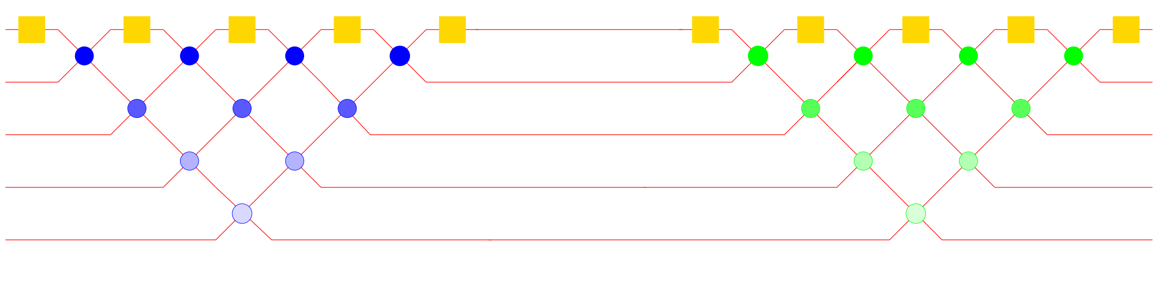
Replaces amplitudes by their Fourier transforms.

INVERSE OF QFT

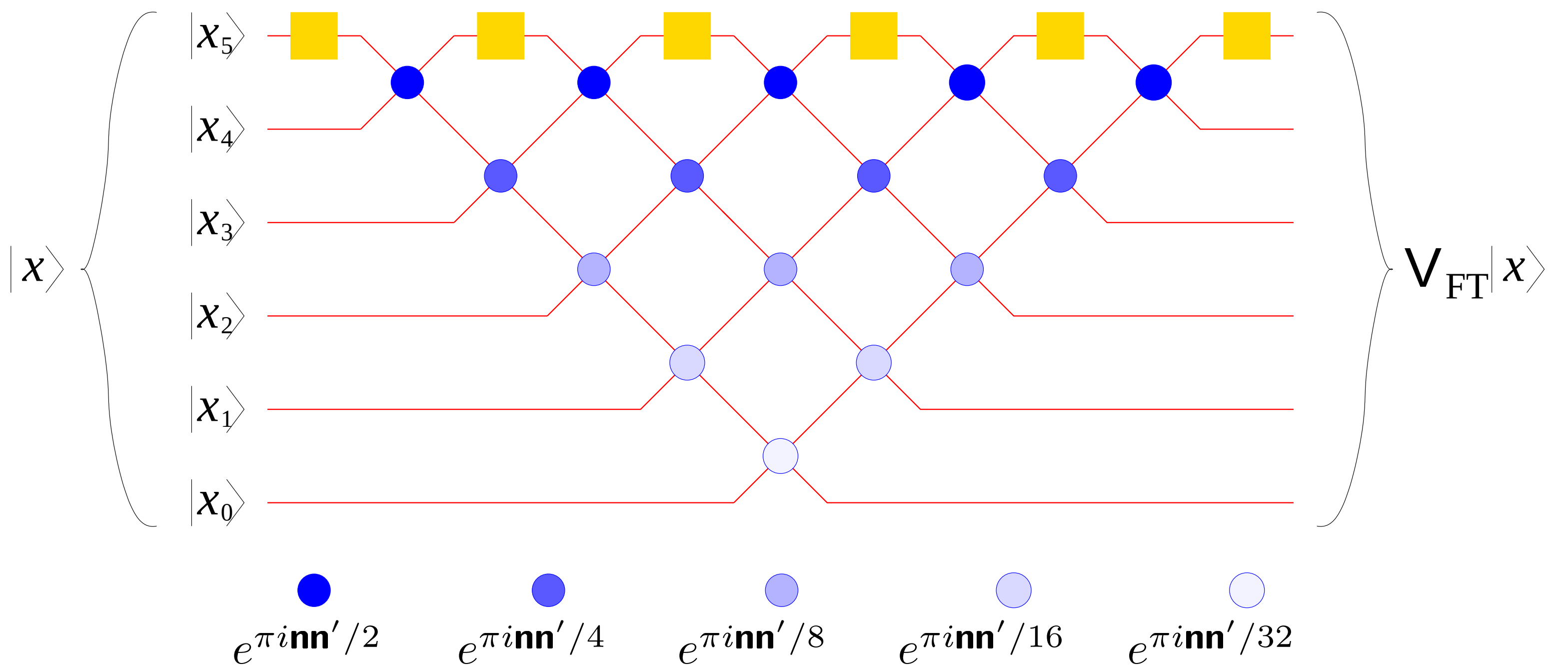








A PROBLEM?



Number n of Qbits: $2^n > N^2$, N hundreds of digits.

Phase gates $e^{\pi i \mathbf{nn}'/2^m}$ *impossible to make* for most m,

since can't control strength or time of interactions

to better than parts in $10^{10} = 2^{30}$.

But need to learn period r to parts in 10^{300} or more!

Question:

So is it all based on a silly mistake?

Answer:

No, all is well.

Question:

How can that be?

Answer:

Because of the quantum-computational interplay between analog and digital.

Quantum Computation is Digital

Information is acquired *only* by measuring Qbits.
The reading of each 1-Qbit measurement gate is only 0 or 1.

The 10^3 bits of the output y of Shor's algorithm are given by the readings (0 or 1) of 10^3 1-Qbit measurement gates.

There is no imprecision in those 10^3 readings.
The output is a definite 300-digit number.

But is it the number you wanted to learn?

Quantum Computation is Analog

Before a measurement the Qbits
are acted on by unitary gates with
continuously variable parameters.

These variations affect the amplitudes
of the states prior to measurement
and therefore they affect the *probabilities*
of the readings of the measurement gates.

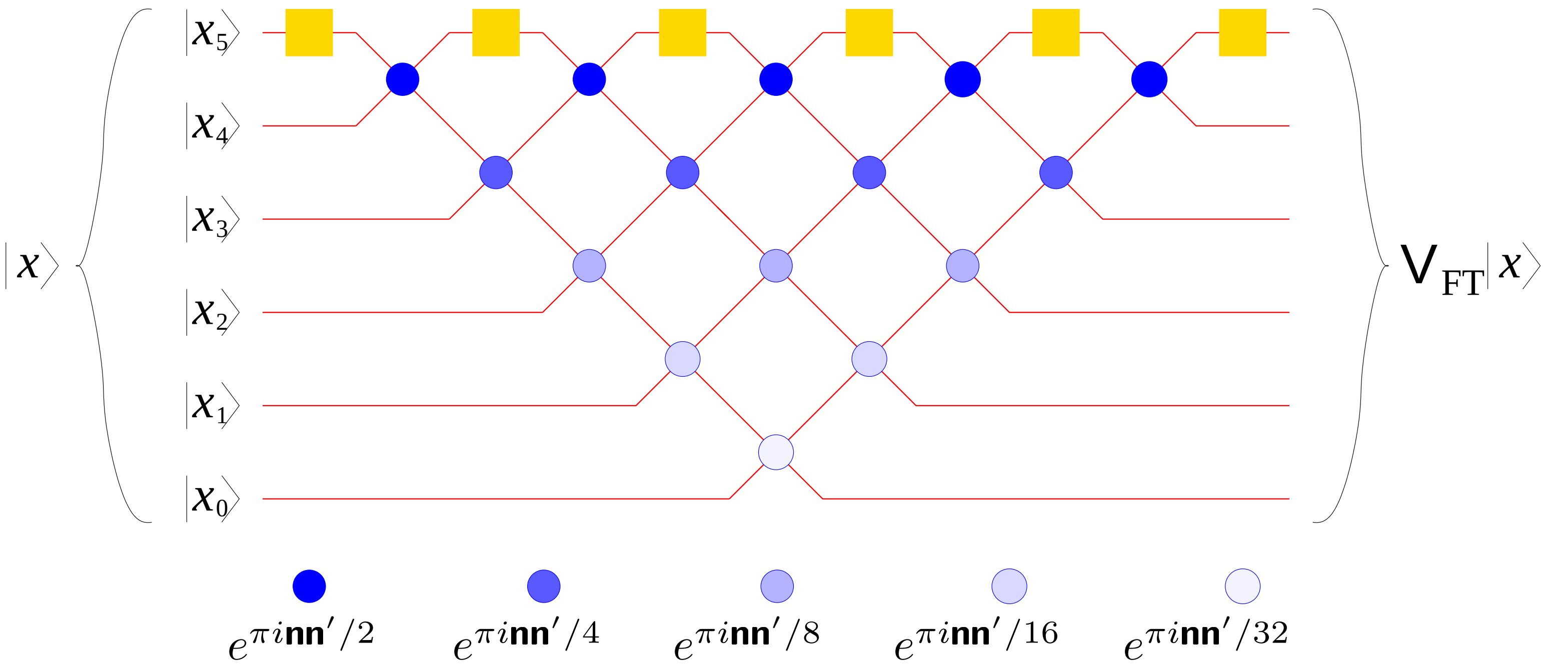
So all is well

“Huge” errors (parts in 10^4) in the phase gates
may result in comparable errors in the *probability* that
the 300 digit number given *precisely* by the measurement gates
is *the right* 300 digit number.

So the probability of getting a useful number
may not be 90% but only 89.99%.

Since “90%” is actually “about 90%”
this makes no difference.

In fact this makes things even better



Since only the top 20 layers of phase gates can matter, once you get to $N > 2^{20} = 10^6$,

the running time scales not quadratically

but only linearly in the number of Qbits.

Quantum Versus Classical Programming Styles

Question:

How do you calculate a^x when x is a 300 digit number?

Answer:

Not by multiplying a by itself 10^{300} times!

How else, then?

Write x as a binary number: $x = x_{999}x_{998}\cdots x_2x_1x_0$.

Next square a, square the result, square *that* result,... , getting the 1,000 numbers a^{2^j}.

Finally, multiply together all the a^{2^j} for which $x_j = 1$.

$$\prod_{j=0}^{999}\left(a^{2^j}\right)^{x_j} = a^{\sum_j x_j 2^j} = a^x$$

Classical: Cbits Cheap; Time Precious

$$a^x = \prod_{j=0}^{999} \left(a^{2^j} \right)^{x_j}$$

Once and for all, make and store a look-up table:

$$a,\ a^2,\ a^4,\ a^8, \ldots ,\ a^{2^{999}}$$

A thousand entries, each of a thousand bits.

For each x multiply together all the a^{2^j} in the table for which $x_j = 1$.

Quantum: Time Cheap; Qbits Precious

Circuit that executes

$$a^x = \prod_{j=0}^{999} \left(a^{2^j}\right)^{x_j}$$

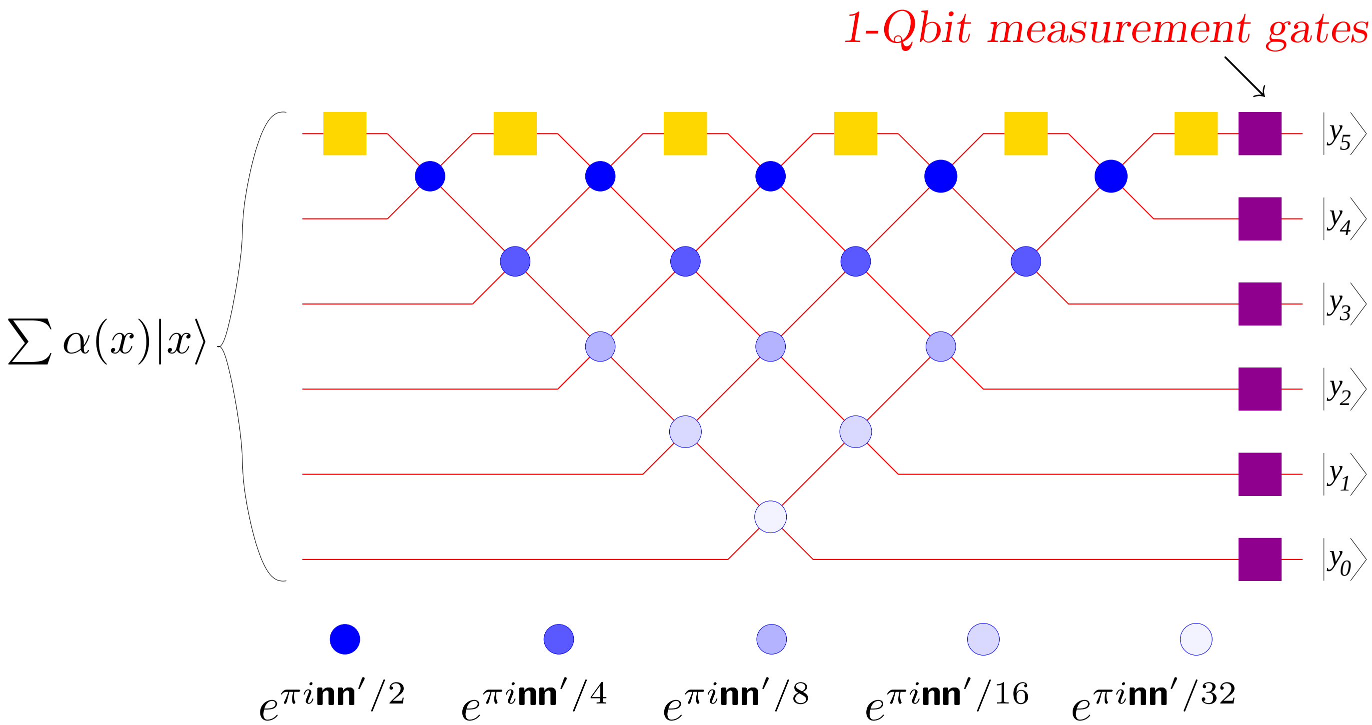
is not applied 2^n times to input register for each $|x\rangle$.

It is applied *just once* to input register in the state

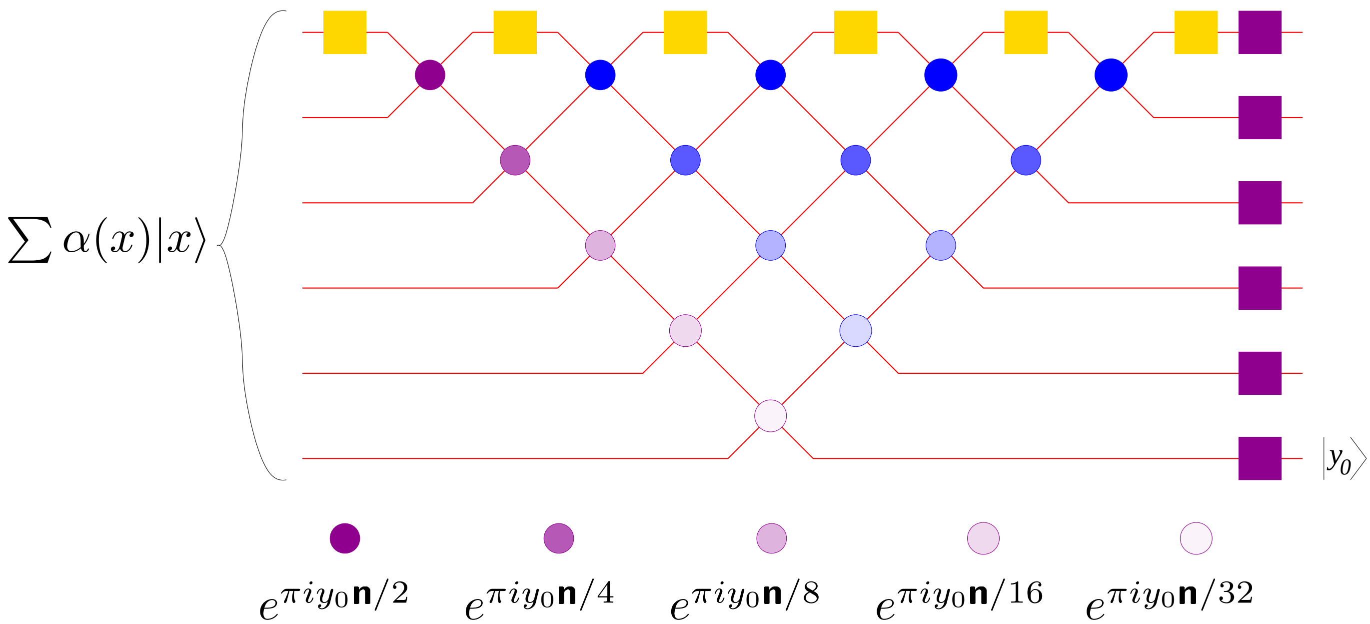
$$|\phi\rangle = \left(\tfrac{1}{\sqrt{2}}\right)^n \sum_{0 \le x < 2^n} |x\rangle.$$

So after each conditional (on $x_j = 1$) multiplication by a^{2^j} can store $\left(a^{2^j}\right)^2 = a^{2^{j+1}}$ using *same* 1000 Qbits that formerly held a^{2^j}.

Another Important Simplification

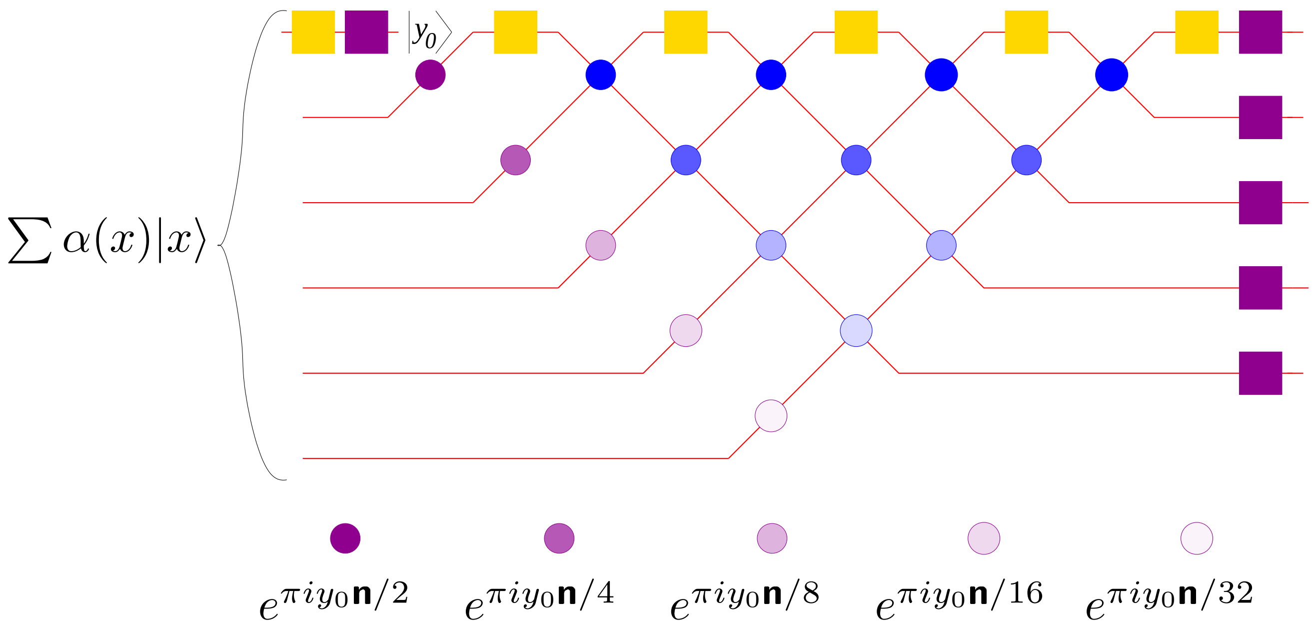


The Important Simplification



2-Qbit operators replaced by 1-Qbit operators, conditional on measurement outcome.

The Important Simplification



You don't need anything but 1-Qbit gates!

Things I wish they had told me about Peter Shor's algorithm (and more general morals for the beginner):

1. Shor algorithm finds periods. Period!
   Periods $\longrightarrow$ factors solely via number-theory.

2. Period-finding is non-trivial for functions that look like random noise within a period.

3. Quantum parallelism doesn't calculate all values of a function using 10^{300} computers in parallel universes.

4. Shor's quantum Fourier transform (QFT) doesn't transform from position to momentum representation.

5. To factor $N = pq$ need enough Qbits to hold N periods of $a^x \pmod N$ except in pathological cases (like $N = 15$).

6. Quantum Fourier transform for n Qbits is built from just $O(n^2)$ gates each of which acts only on single Qbits or on pairs of Qbits.

7. To use it for period finding you need only $O(n)$ such gates.

8. To use it for period finding you can replace the 2-Qbit gates by 1-Qbit gates conditional on measurement outcomes.

9. Quantum computation is a unique blend of digital (measurement gates) and analog (unitary gates).

10. *Classical:* Cbits cheap, time precious.
    Quantum: Time cheap, Qbits precious.

11. Write *Qbit*, not *qubit*.

Some other things I wish they had told me:

Question:

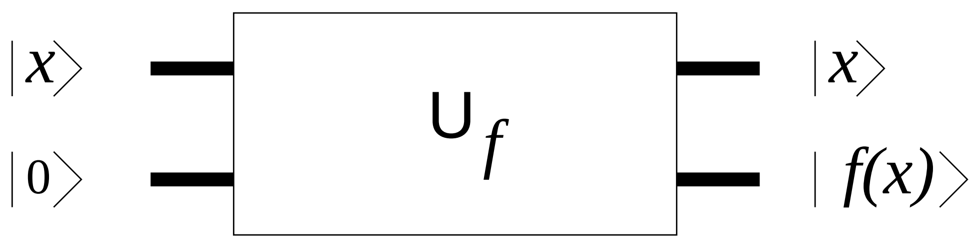
Why must a quantum computation be reversible (except for measurements)?

Superficial answer:

Because linear + norm-preserving ⇒ unitary and unitary transformations have inverses.

Real answer:

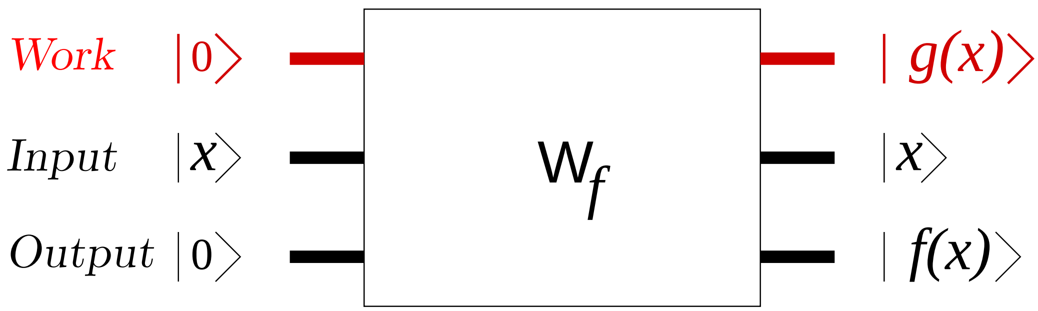
Because standard architecture for evaluating $f(x)$,



oversimplifies the actual architecture:

Need additional work registers for doing the calculation:

Registers

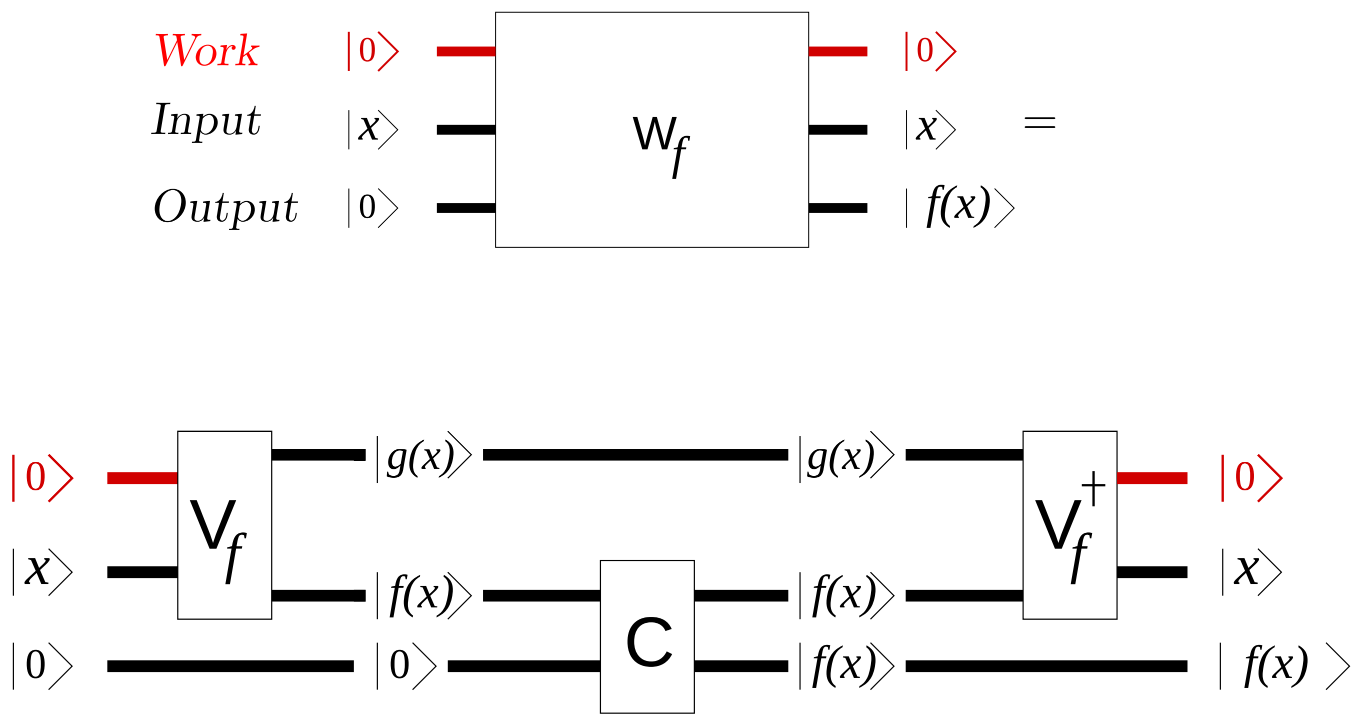


If input register starts in standard state $\sum_x |x\rangle$
then final state of all registers is $\sum_x |g(x)\rangle|x\rangle|f(x)\rangle$.

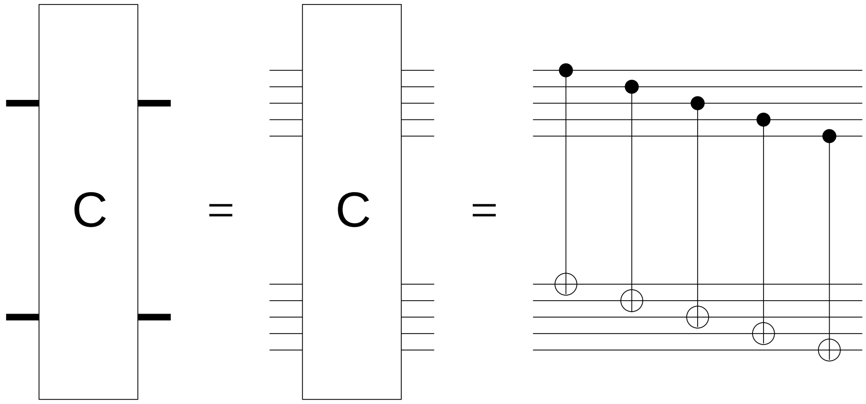
Work register entangled with input and out registers,
unless final state of work register independent of x.
Quantum parallelism breaks down.

Quantum parallelism maintained
if $|g(x)\rangle = |0\rangle$, independent of x.
Final state is then $|0\rangle\left(\sum_x |x\rangle|f(x)\rangle\right)$.

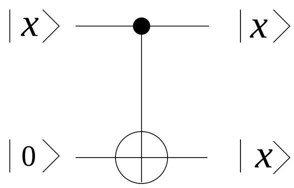
How to keep the work register unentangled:



C *is built out of 1-Qbit controlled-NOT gates:*



controlled-NOT:



Question:

How do you do arithmetic on a quantum computer?

Answer:

By copying the (pre-existing) classical theory of reversible computation.

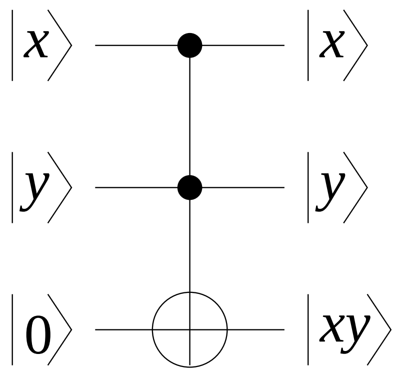
Question (from reversible-classical-computer scientist):

But that theory requires an irreducibly 3-Cbit doubly-controlled-NOT (Toffoli) gate!

Answer:

In a quantum computer 3-Qbit Toffoli gate can be built from a few 2-Qbit gates.

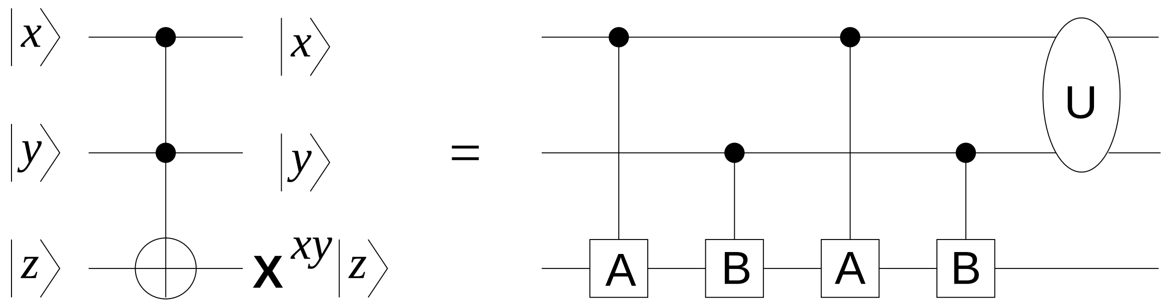
The 3-Cbit Doubly-Controlled-NOT (Toffoli) gate:



↑

logical AND
of x and y

How to build the 3-Qbit Doubly-Controlled-NOT gate out of 2-Qbit gates:



$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x \qquad \mathbf{U} = e^{-\pi i \mathbf{n}\mathbf{n}'/2}$$

$$\mathbf{A} = \hat{\mathbf{a}} \cdot \sigma \qquad \mathbf{B} = \hat{\mathbf{b}} \cdot \sigma \qquad \hat{\mathbf{a}} \times \hat{\mathbf{b}} = \hat{\mathbf{x}} \sin\theta$$

$$\mathbf{A}^2 = \mathbf{B}^2 = \mathbf{1}$$

$$\mathbf{AB} = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} + i\hat{\mathbf{a}} \times \hat{\mathbf{b}} \cdot \sigma = \cos\theta + i\sigma_x \sin\theta$$

$$\left(\mathbf{AB}\right)^2 = \cos 2\theta + i\sigma_x \sin 2\theta$$

If angle θ between $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ is $\pi/4$ then $\left(\mathbf{AB}\right)^2 = i\mathbf{X} = e^{\pi i/2}\mathbf{X}$

Reference:

Quantum Computer Science

N. David Mermin

Cambridge University Press, August 2007