Quantum Computers

Peter Shor
MIT

What is the difference between a computer and a physics experiment?

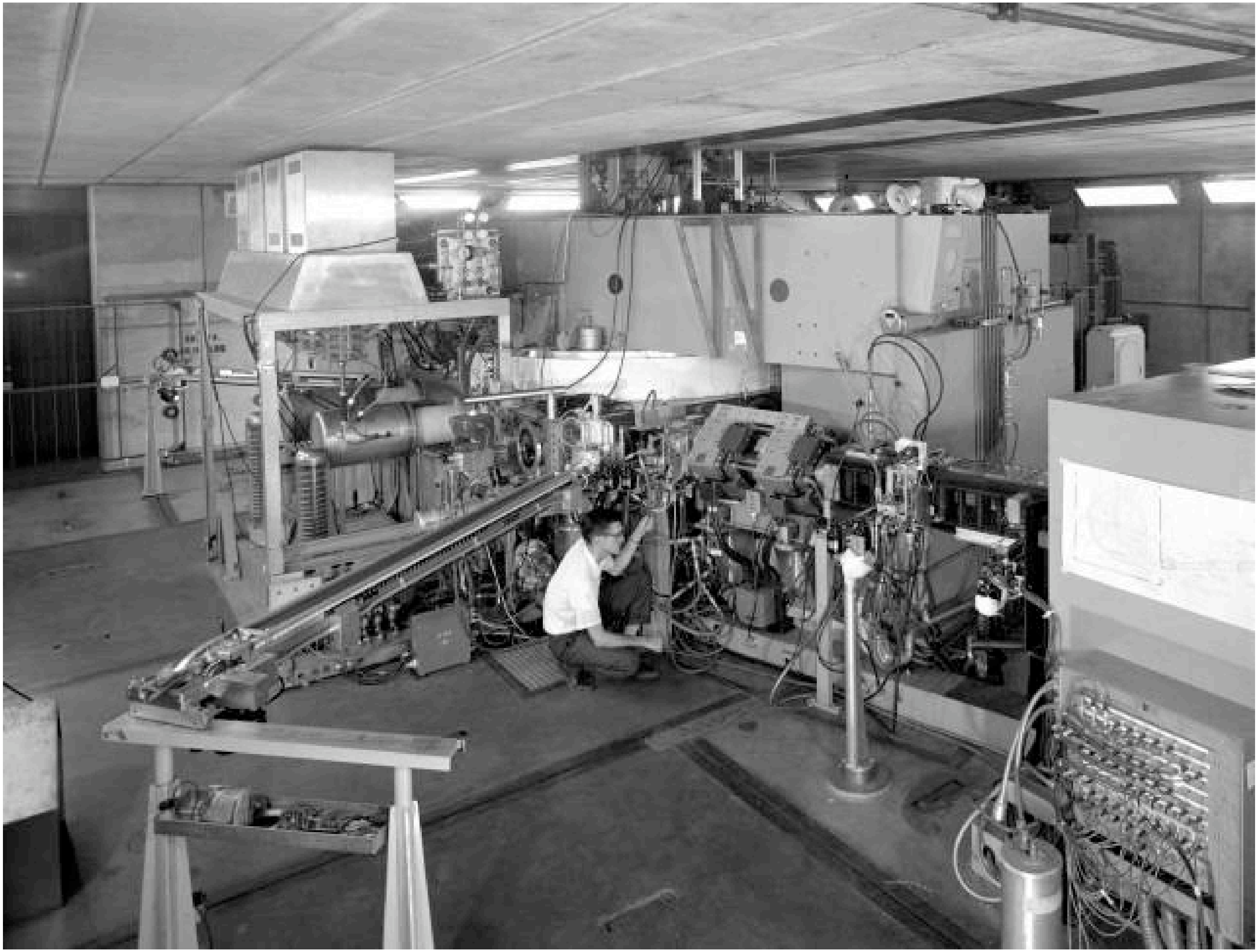
One answer:

A computer answers mathematical questions. A physics experiment answers physical questions.

MAC
PC
TJ Pickell
1999

Another answer:

A physics experiment is a big, custom-built, finicky, piece of apparatus.
A computer is a little box that sits on your desk (or in your briefcase).



A third answer:

You don't need to build a new computer for each mathematical question you want answered.

The mathematical theory of computing started in the 1930's (before computers)

After Gödel proved his famous incompleteness theorem, it was followed by four papers giving a distinction between computable and uncomputable functions
(Church, Turing, Kleene, Post, ca. 1936)

These papers contained three definitions of computable functions which looked quite different.

Universality of computers I.

This led Church and Turing to propose

Church-Turing thesis:

A Turing machine can perform any computation that any (physical) device can perform.
(Turing, Church, ca. 1936).

Until recently, it was not generally realized this is a statement about physics, and not about mathematics.

With the development of practical computers, the distinction between uncomputable and computable become much too coarse.

To be practical, a program must return an answer in a reasonable amount of time (minutes? days? years?).

Theoretical computer scientists consider an algorithm *efficient* if its running time is a polynomial function of the size n of its input. (n^2, n^3, n^4, etc.)

The class of problems solvable with polynomial-time algorithms is called P

This is a reasonable compromise between theory and practice.

For the definition of P (polynomial-time solvable problems) to be meaningful, you need to know that it doesn’t depend on the exact type of computer you use.

Universality of computers II.

This led various computer scientists to propose the

Quantitative Church's Thesis

A Turing machine can perform *efficiently* any computation that any (physical) device can perform efficiently.

(ca. 1970).

If quantum computers can be built, this would imply this "folk thesis" is probably not true.

Misconceptions about Quantum Computers

False: Quantum computers would be able to speed up all computations.

Quantum computers are not just faster versions of classical computers, but use a different paradigm for computation.
They would speed up some problems by large factors and other problems not at all.

What do we know quantum computers are good for?

- Simulating/exploring quantum mechanical systems efficiently. [Richard Feynman/Yuri Manin]

- Finding periodicity.
  Simon's problem [Dan Simon]
  Factoring large integers and finding discrete logarithms efficiently [PWS]
  Pell's equation [Sean Hallgren].

- Searching large solution spaces more efficiently [Lov Grover]
  Amplifying the success probably of (quantum) algorithms with small success probabilities.

Searching

Quantum computers give a quadratic speedup for exhaustive search problems (Lov Grover). Looking through N possibilities takes

- time N on a classical computer.
- time $\pi/4\ \sqrt{N}$ on a quantum computer.

Factoring an L-bit number

Best classical method is the number field sieve (Pollard)
time: $\exp(cL^{1/3}(\log L)^{2/3})$.

Quantum factoring (PWS): theoretical asymptotic time bound $cL^2(\log L)(\log \log L)$

Practical implications

Security on the Internet is based on public key cryptography.

The most widely used (and most trusted) public key cryptosystems are based on the difficulty of factoring and of finding discrete logarithms.

Both of these are vulnerable to attacks by a quantum computer.

The Superposition Principle:

If a quantum system can be in one of two mutually distinguishable states $|\,A\rangle$ and $|\,B\rangle$, it can be both these states at once. Namely, it can be in the *superposition* of states

$$\alpha\,|\,A\rangle + \beta\,|\,B\rangle$$

where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$.

If you look at the system, the chance of seeing it in state $|\,A\rangle$ is $|\alpha|^2$ and in state $|\,B\rangle$ is $|\beta|^2$.

The Superposition Principle (in mathematics)

Quantum states are represented by unit vectors in a complex vector space.

Multiplying a quantum states by a unit complex phase does not change the essential quantum state.

Two quantum states are *distinguishable* if they are represented by orthogonal vectors.

A *qubit* is a quantum system with 2 distinguishable states, i.e., a 2-dimensional state space.

If you have a polarized photon, there can only be two distinguishable states, for example, vertical $|\updownarrow\rangle$ and horizontal $|\leftrightarrow\rangle$ polarizations.

All other states can be made from these two.

$$|\nearrow\rangle = \frac{1}{\sqrt{2}}|\leftrightarrow\rangle + \frac{1}{\sqrt{2}}|\updownarrow\rangle$$

$$|\searrow\rangle = \frac{1}{\sqrt{2}}|\leftrightarrow\rangle - \frac{1}{\sqrt{2}}|\updownarrow\rangle$$

$$\left|\circlearrowright\right\rangle = \frac{1}{\sqrt{2}}|\leftrightarrow\rangle + \frac{i}{\sqrt{2}}|\updownarrow\rangle$$

$$\left|\circlearrowleft\right\rangle = \frac{1}{\sqrt{2}}|\leftrightarrow\rangle - \frac{i}{\sqrt{2}}|\updownarrow\rangle$$

If you have two qubits, they can be in any superposition of the four states

$$|00\rangle \quad |01\rangle \quad |10\rangle \quad |11\rangle$$

This includes states such as an EPR (Einstein-Podolsky-Rosen) pair:

$$\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

where neither qubit alone has a definite state.

If you have n qubits, their joint state is described by a 2^n dimensional vector.

The basis states of this vector space are:

$$|\,000\ldots00\rangle \qquad |\,000\ldots01\rangle \qquad \cdots \qquad |\,111\ldots11\rangle$$

The high dimensionality of this space is one of the places where quantum computing obtains its power.

To compute, we need to

- Put the input into the computer.
- Change the state of the computer.
- Get the output out of the computer.

Input

Start the computer in the state corresponding to the input in binary, e.g.

$$|\,100101101\rangle\,.$$

We may need extra workspace for the algorithm. We then need to add 0s to the starting configuration.

$$|\,100101101\rangle \otimes |\,0000000000\rangle\,.$$

Output

At the end of the computation, the computer is in some state

$$\sum_{i=0}^{2^k-1} \alpha_i \,|\, i\rangle$$

We can NOT measure the state completely, because of the Heisenberg uncertainty principle.

We assume that we measure in the canonical basis. We observe the output i with probability $|\alpha_i|^2$.

Output

When we observe the computer, we get a sample from a probability distribution.

Because of quantum mechanics, this is inherently a probabilistic process. We say that the computer computes a function correctly if we are able to get output that gives us the right answer with high probability.

The Linearity Principle

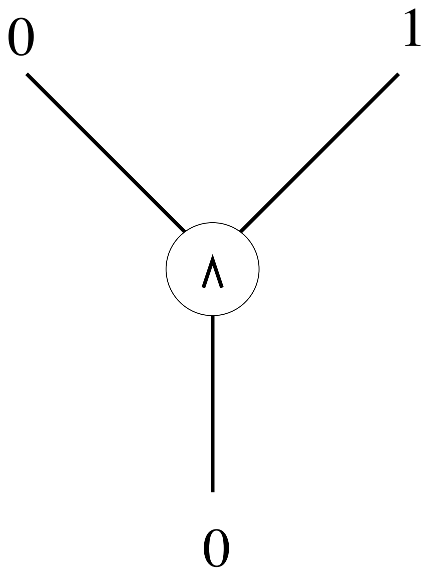
The evolution of an *isolated* quantum system is linear.

Evolution of pure states in an isolated quantum system can be described by a matrix operating on the state space.

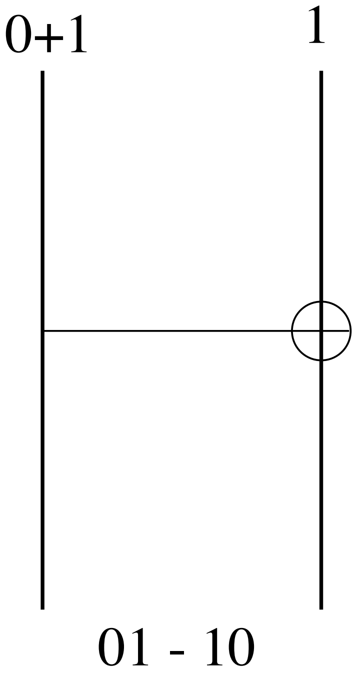
To preserve probabilities, the matrices must be unitary.

Computation

Apply transformations to qubits two at a time.



Classical Gate



Quantum Gate

By linearity of quantum mechanics, a one-qubit gate is a 2×2 unitary matrix and a two-qubit gate is a 4×4 unitary matrix.

A quantum gate is thus a linear transformation on a 2-dimensional (1-qubit) or 4-dimensional (2-qubit) vector space.

It is thus a 2×2 or 4×4 matrix.

In order to preserve probabilities, it must take unit vectors to unit vectors. This means the matrix is *unitary*. That is, if G is the gate, $G^\dagger = G^{-1}$.

A computation (program) is a sequence of quantum gates applied to one or two qubits at a time.

Why two?

Three doesn't give any more power, and seems more complicated experimentally.

Arbitrary many-qubit gates are hopeless, and theoretically realizable ones don't seem to add any extra computational power.

Idea behind fast quantum computer algorithms:

Arrange the algorithm to make all the computational paths that produce the wrong answer destructively interfere, and the computational paths that produce the right answer constructively interfere, so as to greatly increase the probability of obtaining the right answer.

Idea Behind All Fast Factoring Algorithms

To factor a large number N, Find numbers a and b so that

$$a^2 = b^2 \quad \text{mod N}$$

$$a \neq \pm b \quad \text{mod N}$$

Then

$$a^2 - b^2 = (a + b)(a - b) = cN$$

We now extract one factor from $a + b$ and another from $a - b$.

Example: Factoring 33

Take the numbers $a = 7$ and $b = 4$. Then 49 divided by 33 has remainder 16, so

$$7^2 = 4^2 \mod 33$$

Then

$$7^2 - 4^2 = (7 + 4)(7 - 4) = 33$$

and we find $33 = 3 * 11$.

Quantum Factoring Idea

To factor a large number N:

Find the smallest $r > 0$ such that $x^r \equiv 1 \pmod{N}$.

$$(x^{r/2} + 1)(x^{r/2} - 1) \equiv 0 \pmod{N}.$$

We now get two factors by taking the greatest common divisors

$$\gcd(x^{r/2} + 1,\ N)$$

$$\gcd(x^{r/2} - 1,\ N)$$

We can show this gives a non-trivial factor for at least half of the residues $x \pmod{N}$.

How do we find r with

$$x^r \equiv 1 \pmod{N}?$$

Find the period r of the sequence $x^a \pmod{N}$.

Need to find the period of x^a (mod N).

Idea: Use the Fourier transform

Problem: The sequence has an exponentially long period

Solution: Use the exponentially large state space of a quantum computer to take an exponentially large Fourier transform efficiently.

Factoring L-bit numbers

We will work with quantum superpositions of two registers

Register 1	Register 2
$2L$ bits	L bits

We will not give the fine details of the algorithms.

These involve more workspace
($3L$ is easy, $o(L)$ is possible).

Reversible Computation

We can do classical computations on a quantum computer as long as we can do these classical computations *reversibly*. That is, with gates each of whose possible outputs maps uniquely back to the inputs.

Any classical computation can be made reversible as long as we keep the input around.

The 3-bit *Toffoli gate* is a universal gate for reversible computation

$$(x, y, z) \rightarrow (x, y, z \oplus (x \wedge y))$$

This Toffoli gate can be implemented as a sequence of 2-qubit quantum gates.

Quantum Fourier Transform over Z_{2^k}

Have k qubits

$$|\,x\rangle \rightarrow \frac{1}{2^{k/2}} \sum_{y=0}^{2^k-1} \text{exp}(\frac{2\pi ixy}{2^k})\,|\,y\rangle$$

This is easily seen to be a unitary transformation on the 2^k-dimensional space of k qubits.

In order to implement this on a quantum computer, we need to break this into a series of 2-qubit gates.

The Cooley-Tukey FFT easily adapts to give $O(k^2)$ steps.

$$|\,0\rangle\,|\,0\rangle$$

$$\downarrow \quad \approx L \text{ steps}$$

$$\frac{1}{2^L}\sum_{a=0}^{2^L}|\,a\rangle\,|\,0\rangle$$

$$\downarrow \quad \approx L^2 \log L \log\log L \text{ steps}$$

$$\frac{1}{2^L}\sum_{a=0}^{2^L}|\,a\rangle\,|\,x^a \pmod{N}\rangle$$

$$\downarrow \quad \approx L^2 \text{ steps}$$

$$\frac{1}{2^L}\sum_{a=0}^{2^L}\sum_{c=0}^{2^L}|\,c\rangle\,|\,x^a \pmod{N}\rangle\;\; e^{2\pi iac/2^{2L}}$$

Observe computer.

We need to find the probability amplitude on

$$|\, c\rangle \,|\, x^a \text{ (mod } N)\rangle$$

in the superposition

$$\frac{1}{2^L} \sum_{a=0}^{2^L} \sum_{c=0}^{2^L} |\, c\rangle \,|\, x^a \text{ (mod } N)\rangle \;\; e^{2\pi iac/2^{2L}}$$

Many different values of a give the same value of x^a (mod N).

We have to add the coefficients $e^{2\pi iac/2^{2L}}$ on all of them.

Let a_0 be the smallest non-negative integer such that

$$x^{a_0} \equiv x^a (\text{mod } N).$$

Then x^{a_0}, x^{a_0+r}, x^{a_0+2r}, ... are all equal (mod N).

Each contributes to the amplitude on

$$|\, c\rangle \,|\, x^a \text{ (mod } N)\rangle$$

with the coefficient $e^{2\pi i(a_0+br)c/2^{2L}}$.

We observe $|\, c\rangle$ with probability proporitional to

$$\left| \sum_{b=0}^{\approx 2^{2L/r}} e^{2\pi ibrc/2^{2L}} \right|^2$$

This is a geometric sum which is close to 0 unless

$$\frac{rc}{2^{2L}} = d + O(r/2^{2L})$$

for some integer d.

We know

$$\frac{rc}{2^{2L}} = d + O(r/2^{2L}).$$

Thus

$$\frac{c}{2^{2L}} = \frac{d}{r} + O\left(\frac{1}{N^2}\right).$$

with $r < N$.

$\frac{d}{r}$ will be one of the closest fractions to $\frac{c}{2^{2L}}$ with numerator and denominator less than N.

We can use continued fractions to find $\frac{d}{r}$, and then use r to factor N.

Example: Factoring 33

Take $x = 5$. Then (mod 33) we get

1	5	5^2	5^3	5^4	5^5	5^6	5^7	5^8	5^9	5^{10}	5^{11}
1	5	25	26	31	23	16	14	4	20	1	5

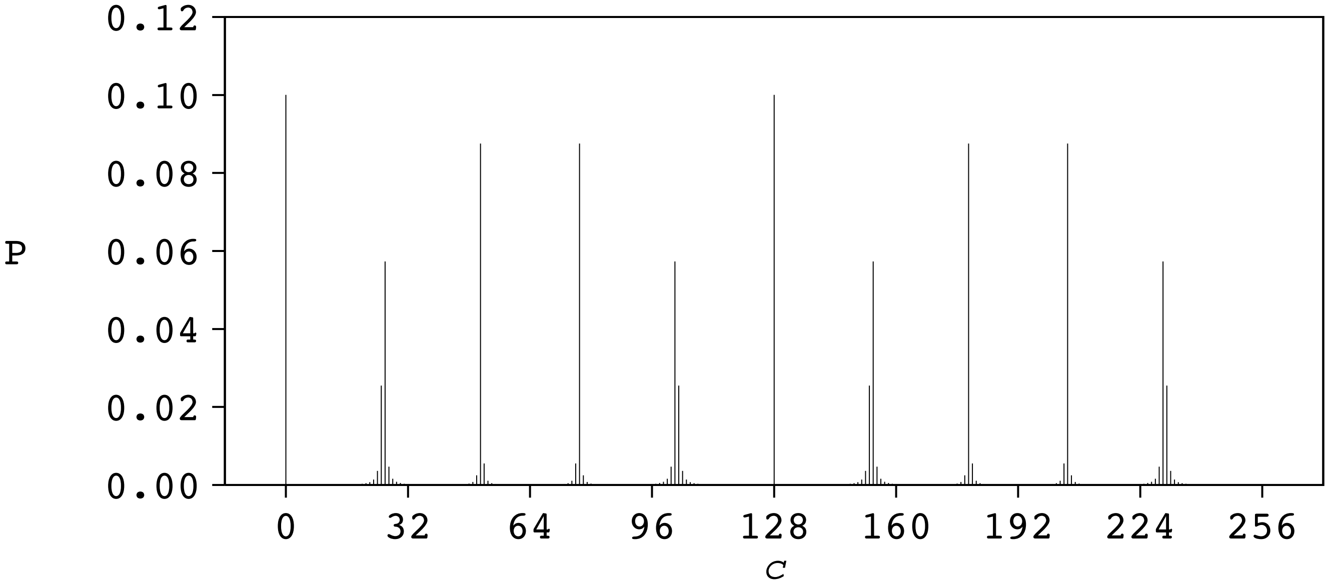
The period r is 10, and

$$x^{r/2} = 5^5 = 23 \quad \text{mod } 33.$$

Then

$$33 \text{ divides } (23 + 1)(23 - 1) = 24 * 22$$

Taking greatest common divisors, 24 gives us the factor 3, and 22 gives us the factor 11, and we find $33 = 3 * 11$.



Example: Factoring 33

The period r is 10.

Difficulties of Quantum Computing

Quantum states are notoriously hard to manipulate.

To do 10^{10} steps on a quantum computer without error correction, and still come up with the right answer, you would need to perform each step with accuracy one part in 10^{10}.

The same objection was raised to scaling up classical computers in the 1950's.

Von Neumann showed that you could build reliable classical computers out of unreliable classical components.

Currently, we don't use many of these techniques because we have extremely reliable integrated circuits, so we don't need them.

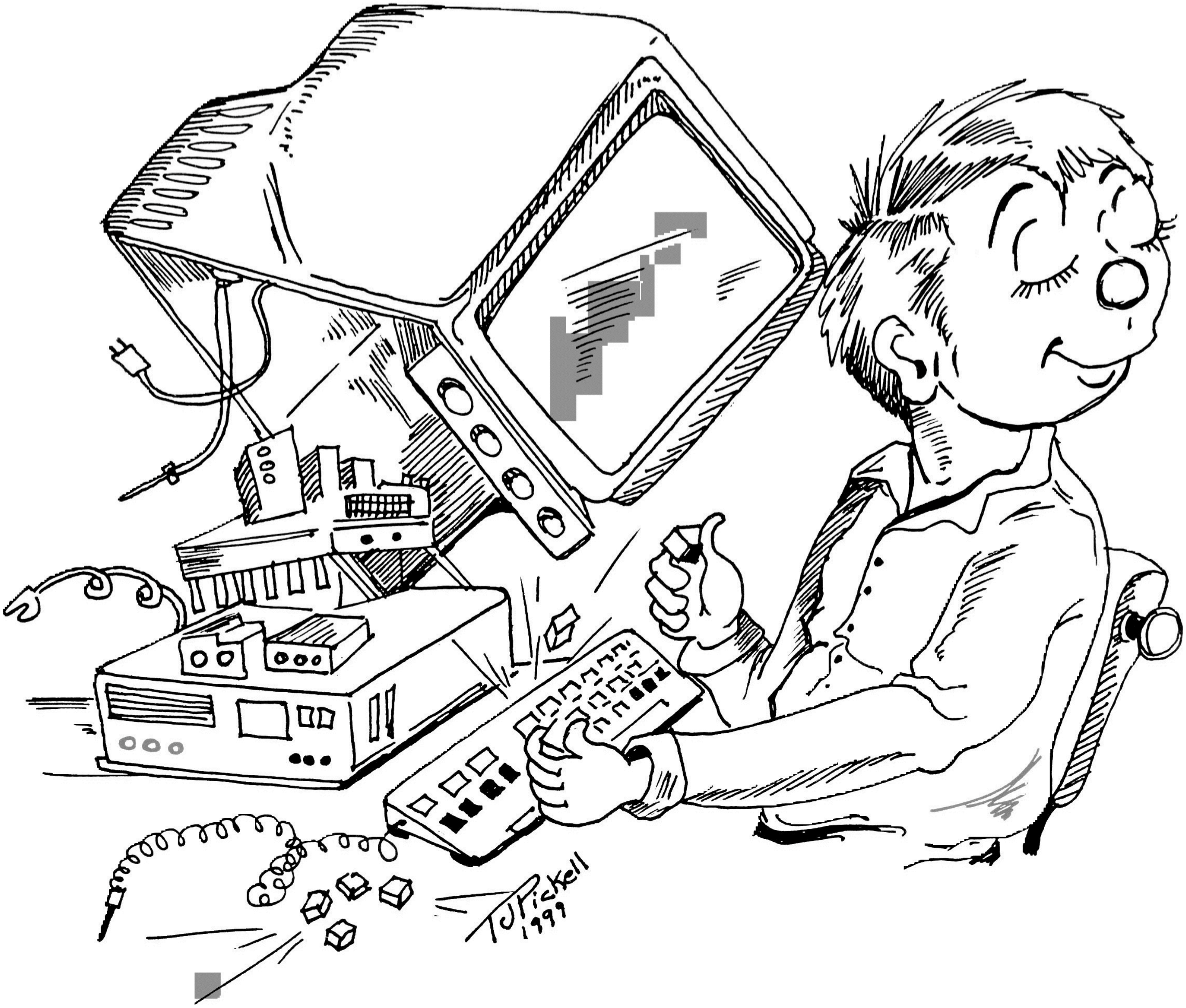
Quantum Computing Difficulties

Heisenberg Uncertainty Principle:

You cannot measure a quantum state without changing it.

No-Cloning Theorem:

You cannot duplicate an unknown quantum state.


TJ Pickell
1999

Main techniques for fault-tolerance
on classical computers.

- Consistency Checks
- Checkpointing
- Error-Correcting Codes
- Massive Redundancy

Quantum error correction

Quantum error correcting codes exist.

They can be used to make quantum computers fault-tolerant. so that you only need to perform each step with accuracy approximately one part in 10^4.

How do quantum error-correcting codes get around the no-cloning theorem Heisenberg uncertainty principle?

Measuring one of the qubits gives NO information about the encoded state, so the remaining qubits can retain all the information about the encoded state without violating the non-cloning theorem.

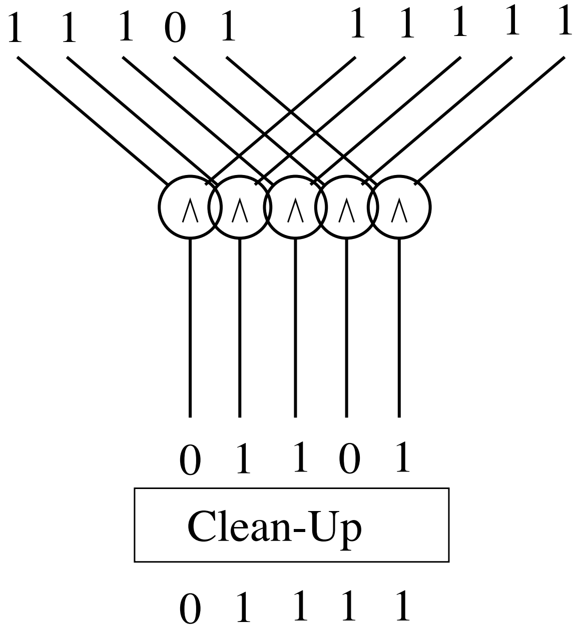
We design the codes so that we can measure the error without measuring (or disturbing) the encoded state.

This means that all likely errors are *orthogonal* to the encoded data

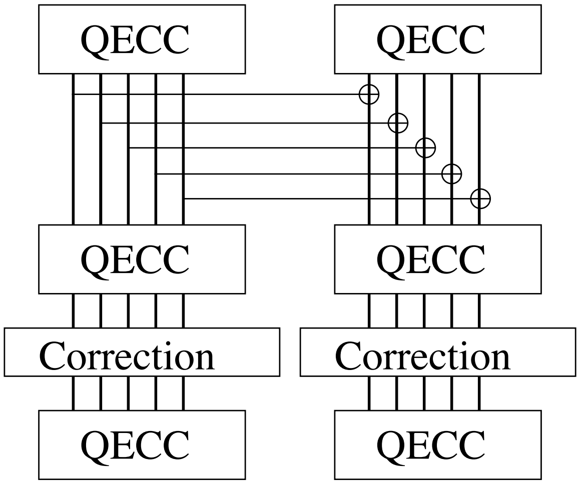
We can then fix the error without destroying the encoded data.

Fault Tolerant Computing

Classically:



Quantum Mechanically:



Best current results

If the quantum gates are accurate to around 1 part in 10^4, you can make fault-tolerant quantum circuits.

(This estimate uses some heuristic arguments; the best rigorous result is 1 part in 10^6.)

The best upper bound is around 1 part in 5.

Open question: What is the right threshold?