Quantum Parallelism — Where Quantum Computers Get Their Mojo From

How quantum computers harness quantum superposition to execute many computational paths simultaneously.

Viraj Kulkarni Jul 4, 2020 · 6 min read ★

Image by MonikaP from Pixabay

Quantum computers were proposed in the 1980s. Since then, physicists have been laboriously working to harness the power of nature to meet computing demands. There is no single best method of physically realising a quantum computer; the field is fragmented into several competing

approaches such as ion traps, optical lattices, photon quantum bits, nuclear magnetic resonance etc. But these different approaches all work towards implementing the same model of quantum computation in hardware. As long as we work with the model correctly, we need not concern ourselves with how it is implemented in hardware. To be fair, there are more than one models of quantum computation as well, but we shall only consider the standard quantum circuit model in this article.

We break this article into four parts. In the first part, we will understand what a single qubit is. Then, we shall see how multiple qubits can be represented. In the third part, we introduce transformations that act on qubits. And finally, we put all these together to explain how quantum computers can execute multiple computational paths in parallel.

Single Qubit

Classical computers operate on bits where each bit can be in the state 0 or 1. Quantum computers operate on quantum bits or *qubits*. A qubit exists not as 0 or 1, but as a superposition of the two (remember Schrödinger's cat that is both dead and alive at the same time?).

Let's represent this mathematically. The state of a qubit ψ can be written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

The above notation is called the braket notation, but we won't cover it here. This qubit exists as both 0 and 1, but if it is measured, we will get *either* 0 or 1 but not both. The probability that we get 0 is given by $|\alpha|^2$, and the probability we get 1 is given by $|\beta|^2$. Once measurement is performed, the qubit loses its superposition and continues to exist in its observed state as either 0 or 1. The braket notation is quite useful for many reasons, but the same equations can also be conveniently written using matrices and vectors as:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

$$|\psi\rangle = \alpha \begin{bmatrix}1\\0\end{bmatrix} + \beta \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

Disclaimer: *The article uses loose notation which is not technically fully correct. For instance, the above states 0 and 1 should be written as basis vectors $|0>$ and $|1>$. The reason for adopting the loose notation is the unfortunate*

fact that medium does not permit using mathematical symbols in text, and converting every instance into an image makes the article awkward to read.

Multiple Qubits

The state of two *unentangled* qubits can be represented as a tensor product of their individual states. If the first qubit has amplitudes a and b, and the second qubit has amplitudes c and d, their combined state can be written as:

$$|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

Some two-qubit states cannot be decomposed into tensor products of two individual qubits. Such qubits are called *entangled* qubits. Entanglement plays a central role in many quantum algorithms especially in the field of quantum cryptography. There is no counterpart to quantum entanglement in classical physics. We will not be reviewing entanglement in this article.

Quantum Transformations

There's no fun if we cannot do anything with qubits. But we can — with the help of transformations. Classical computers operate on bits with the help of logic gates such as NOT, AND, OR, NAND, NOR, XOR etc. Likewise, quantum computers operate on qubits using quantum gates. Owing to the posulates of quantum mechanics, all operations performed on qubits must be linear and reversible. Consequently, all quantum gates need to be linear and reversible.

The NOT gate is the simplest of them. It simply inverts 0 and 1.

$$\alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{\text{NOT}} \alpha\,|1\rangle + \beta\,|0\rangle$$

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

One gate that pervades quantum computing is the Hadamard gate. It transforms a qubit existing only as 0 into an equal superposition of 0 and 1 ie. if we measure the qubit, we will get a 0 with probability 50%, and we will get 1 with probability 50%. This, as we shall see, serves as a great starting point for many quantum algorithms.

$$|0\rangle \xrightarrow{\text{H}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

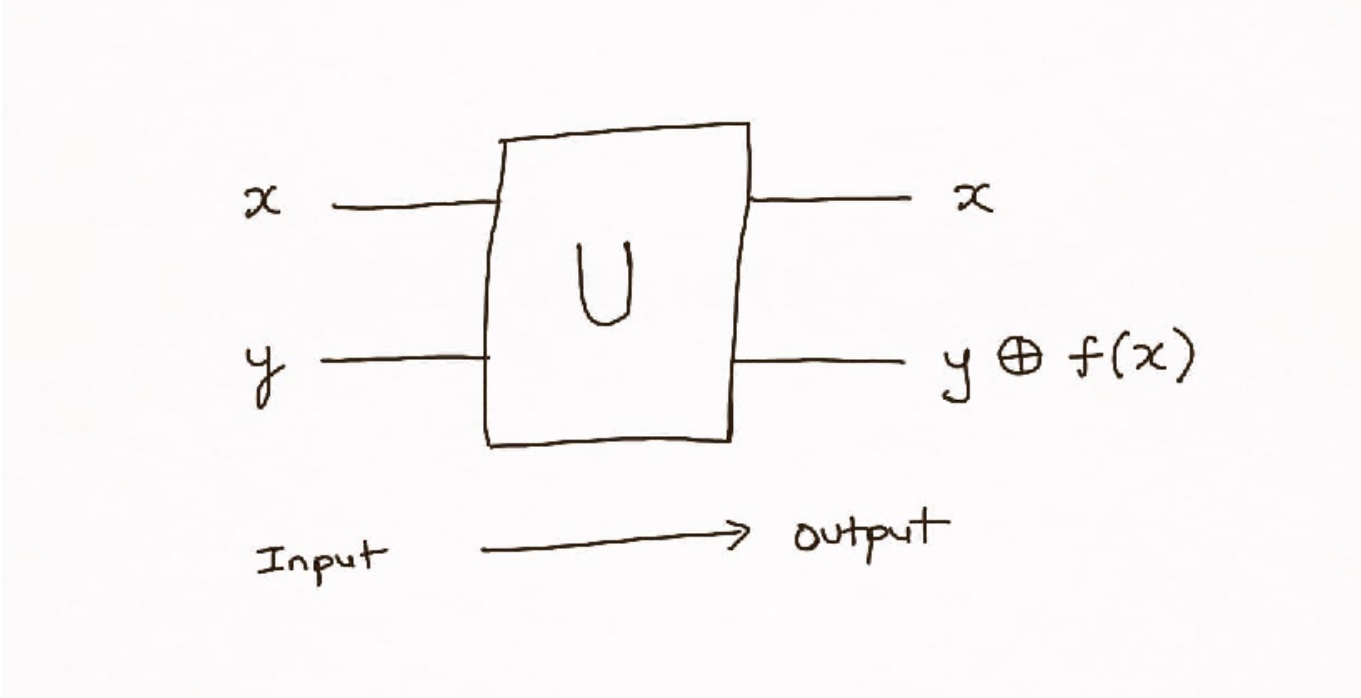
Quantum Parallelism

Now, armed with the above knowledge, let's unravel how quantum computers do their magic. We will do this using a simple example working on two bits. Suppose you are given a classical function *f* that takes two bits as input and returns one bit as output.

$$f(x) : \{0,1\}^2 \rightarrow \{0,1\}$$

To evaluate *f* on all four permutations of two bits, we will need to call *f* four times: *f*(0,0), *f*(0,1), *f*(1,0), *f*(1,1). Exploiting quantum parallelism allows us to evaluate all four inputs in a single call to *f*. Note, however, that our function *f* is not reversible, and all operations on qubits must be reversible. So we first define a reversible version of *f* as follows:

$$U\left|x\right\rangle\left|y\right\rangle \rightarrow \left|x\right\rangle\left|y \oplus f(x)\right\rangle$$



The plus symbol with the circle around it denotes the XOR operation. The quantum oracle *U* takes two inputs *x* and *y* and outputs two values. The first is *x* itself. The second is the value: *y XOR f(x)*. It is easy to see that when *y=0,* the second output equals *f(x)*.

We set up the input ϕ as an equal superposition of the four two-bit inputs 00, 01, 10, 11. To do this, we initialise two qubits as 0 and apply the

Hadamard gate on both of them. This is represented as:

$$|\phi\rangle = (H \otimes H)\,|00\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

Next, we apply our quantum function U on this state by setting $y=0$. This gives us:

$$\begin{aligned} U\,|\phi\rangle\,|0\rangle &= \frac{1}{2}U(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |0\rangle \\ &= \frac{1}{2}U(|00,0\rangle + |01,0\rangle + |10,0\rangle + |11,0\rangle) \\ &= \frac{1}{2}(|00,f(00)\rangle + |01,f(01)\rangle + |10,f(10)\rangle + |11,f(11)\rangle) \end{aligned}$$

If we separate out the two qubits, we see that the second output qubit contains the superposition of all four results we are interested in. Thus, we evaluated four inputs using a single application of the target function.

> For a more detailed but still gentle introduction to quantum computing, please see the review paper at https://arxiv.org/abs/2006.12025

Discussion

Quantum parallelism forms the heart of many quantum algorithms. There are however some important caveats we need to consider. Although the output contains a superposition of the results we are interested in, we cannot directly read them. Any direct form of measurement will give us a single result and the other three will be lost. Clever tricks therefore need to be used to read out the final answer we are interested in, and these tricks may not be computationally trivial. Secondly, for our example, preparing the input state was simple, but this may not be the case for other problems. Whether using quantum parallelism will eventually lead to speed-up over classical alternatives depends on these caveats.

Quantum computers are already here, and it's time we start taking quantum seriously. The more people from diverse fields embrace it, the faster it will become a reality.

If you liked the article, check out my other pieces on Medium, follow me on LinkedIn or Twitter, view my personal webpage, or email me at viraj@berkeley.edu.

Quantum Computing Quantum Physics Data Science