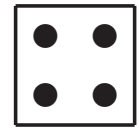


The Birth of Quantum Computing

In the 1940s England used its first electronic computers to crack enemy codes, while the US used its computers to perform computations for nuclear physics. Eight decades later, these same two applications are driving interest and investment in quantum computing. If the effort to build large-scale quantum computers is successful, these machines will surely be used to crack codes and model physics. But just as electronic computers eventually had many more applications than dreamed of in the 1940s, quantum computers will, in all likelihood, find work solving problems that are not even contemplated today.

This is the first of three chapters on quantum computing. We discuss this history in some depth in order to provide an intellectual foundation for understanding both how different quantum computers are from classical computers, and for helping readers to form an appreciation of just how early we are in the development of these machines. This appreciation will be relevant when we review policy issues in Chapter 8 and Chapter 9.

This chapter is based on both bibliographic research and interviews conducted with many quantum computing pioneers. Readers uninterested in this history can skip to Chapter 6, where we discuss the applications of quantum computing likely to be seen in the near future, the different kinds of quantum computers currently under development, the challenges facing the field, and the more distant future outlook for the technology.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

4.1 Why Quantum Computers?

Quantum computers are strikingly different from classical computers, and billions of dollars have been spent developing them today without any payoffs other than papers in prestigious scientific journals. To date, this aggressive research program seeks to realize three specific applications for quantum computers: simulating physics and chemistry, factoring numbers, and searching for optimal solutions to specific kinds of mathematical computations ("optimization").

4.1.1 Richard Feynman and Quantum Computing

In 1981, the American physicist Richard Feynman (1918–1988) proposed that the kinds of mathematical problems that quantum physicists need to solve might be more efficiently worked on using a computer based on quantum mechanics than one based on classical physics.[1] Feynman was speaking at a conference exploring the physics of computation co-sponsored by MIT and IBM. Held at MIT Endicott House Conference Center, a converted mansion built in the style of a French manor house in the Boston suburbs, the conference brought together an eclectic collection of roughly 50 renowned physicists and computer scientists. Feynman was the conference's big draw, and his proposal makes this conference the proper birthplace of quantum computing.

Of course, all present-day computers are based on quantum mechanics: computers use the flow of electrons, and electrons are the quantization of electronic charge. But computer engineers (the professionals who design the hardware of computers) go out of their way to make electrons behave as if they are classical objects – as if they were little balls traveling along wires, like water through a pipe. Indeed, in the 1970s, as the feature size of semiconductor lithography got smaller and smaller, some scientists were concerned that the walls between those pipes were getting so thin that electrons might seep (or "tunnel") from one pipe to another, causing an error. Specifically, the fear was that *quantum tunneling*, a consequence of the Heisenberg uncertainty principle, might slow or even halt the relentless march of Moore's Law (see Section 3.5, "Moore's Law, Exponential Growth, and Complexity Theory" (p. 98)). So Feynman's idea that computer engineers might actually want to embrace the uncertainty, nondeterminism and inherent randomness that comes with quantum phenomena was a radical proposal indeed.

[1] Feynman, "Simulating Physics with Computers" (1982).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Quantum Confusion

Some popular accounts of quantum computing present key concepts inaccurately. Here we set the story straight.

Quantum computers are parallel machines, but they do not solve hard problems by trying all possible solutions at once. Quantum computers run in parallel, inasmuch as a machine with 50 qubits uses all 50 at once.[a] If we build a quantum system with 10 million qubits, all of those qubits will compute in parallel. While some quantum speedup comes from this parallelism, it is thought that more comes from the ability of quantum computers to compute with quantum wave equations.

Qubits are a superposition of two possibilities, but this does not mean that two qubits simultaneously have four values (`00`, `01`, `10` and `11`). Qubits do not simultaneously have two values any more than Schrödinger's cat is both alive and dead at the same time (see p. 523). A qubit has a single, definite quantum state when it is measured: that state is either a `0` or `1`, and the probability that it will be in one state or the other depends on the quantum calculation.

Quantum computers cannot store an exponential amount of information. Google has built a quantum computer with 53 qubits, but it cannot store 2^{53} bits (8192 TiB) of information. Google's quantum computer has *no storage at all* in the conventional sense. Each time the computer solves a problem, it selects a single 53-bit result from 2^{53} possible answers.

Quantum computers use superposition and entanglement, but they do not simultaneously consider every possible variation of complex puzzles. That would require cycling repeatedly forwards and backwards, performing additional computations with every cycle. Reusing space and time in this manner would be powerful, but this is not how quantum computers work, and it's probably not possible in our universe.[b]

[a] Likewise, even the 8-bit microprocessor in the original Apple II was a parallel machine, in that it could add and subtract 8 bits *in parallel* at a given time. To find a true serial machine, you need to go back to the very first digital computers and their so-called *bit-serial architectures*. These machines added 8-bit numbers a single bit at a time.

[b] For a discussion of time-travel computing, see Aaronson, "Guest Column: NP-Complete Problems and Physical Reality" (2005).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Before the dawn of quantum computing, computer engineers had always tried their best to hide the uncertainty and inherent nondeterminism of the quantum realm in every circuit that they designed. Computers built using tubes in the 1950s and transistors ever since do this by using large ensembles of electrons to represent each `0` and `1` – and by strenuously avoiding the roll of the dice that is inherent in all things involving quantum mechanics. Instead of building computers that are governed by probability, computer engineers have traditionally built machines that they hoped would be *deterministic*. That is, they hoped that the computer would always generate the same output given the same input. When their computers didn't, they called such behavior a bug. Nowadays, we enjoy the successes of computer engineers pursuing determinism. One's computer can process billions of bits a second and run for years without crashing.

Deterministic machines are great for running spreadsheets and typesetting books, but they are poorly suited for analyzing quantum systems, such as a the chemistry of a molecule. That is because the complexity of a quantum system scales exponentially with the number of particles that the system contains: it might take 16 times longer to analyze a molecule with eight atoms compared to a molecule with four. A molecule with 10 atoms might take 64 times longer to study.

Feynman's key insight was realizing that the exponential scaling inherent in modeling quantum systems with classical computers might be avoided by using a computer built from the ground up on the math of quantum mechanics – that is, a computer designed to preserve and embrace the nondeterminism of quantum states. But to do that, quantum computers would have to do something that conventional computers can't readily do: they would have to be able to run backwards.

4.2 Reversibility

The idea that quantum processes could represent digital information and be used for computing emerged slowly in the 1970s. One of the first building blocks, largely worked out at IBM and MIT, was the idea of reversible computing.

Reversibility is a property of both classical and quantum physics, and it has profound implications. In classical physics, reversibility means that astronomers can take the equations used to predict the motion of the Sun, Moon, planets, and stars in the future and *run*

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

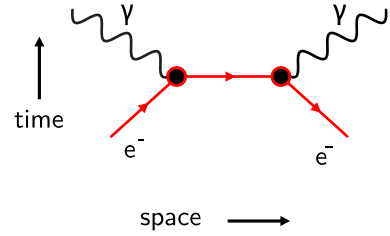


Figure 4.1. A Feynman diagram showing an electron–positron annihilation; rotate it 90° and you have an electron absorbing and then re-emitting a photon (γ). Note that time flows *up* in the diagram, along the vertical axis, while the three dimensions of space are represented as a projection along the horizontal axis.

the equations backwards to determine where those celestial bodies were located in the past. Indeed, taken from the vantage point of celestial mechanics, the direction of time is arbitrary.

In quantum physics, reversibility means that quantum processes can easily go forwards or backwards. In fact, at the quantum level, it is even possible to swap time with space.

4.2.1 The Arrow of Time

In 1948, Feynman, then a professor at Cornell University, came up with a visualization for describing how subatomic particles interact. The diagrams replaced the complex and hard-to-understand mathematics that physicists had previously used with pictures that can be understood even by a lay audience. They were so revolutionary and became so ubiquitous that today we call them *Feynman diagrams*.[2]

The Feynman diagram in Figure 4.1 depicts what happens when an electron (e^-) collides with a positron (e^+). Positrons are basically electrons that have a positive charge instead of a negative charge. Otherwise, electrons and positrons are identical. (When looking at the diagram, remember that time flows up from the bottom of the page to the top.)

[2]Feynman's diagrams were initially rejected by his peers, but gained popularity in the 1950s as Feynman successively refined his theory of how light and electrons interact. Feynman went on to share the 1965 Nobel Prize in Physics with Sin-Itiro Tomonaga and Julian Schwinger, "for their fundamental work in quantum electrodynamics, with deep-ploughing consequences for the physics of elementary particles." Flamboyant and commanding, today Feynman is also known for his ability to explain physics to lay audiences, for doing so with infectious enjoyment and captivating joviality, and for his work analyzing the NASA space shuttle *Challenger* disaster.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

The positron is called an *antiparticle* because when it interacts with an electron, the two particles annihilate each other, leaving two gamma particles[3] traveling away from each other each at the speed of light. This is the classic "matter–antimatter" reaction popularized in the 1960s television series *Star Trek* – one of the many bits of science at the heart of the series' science fiction.[4]

Recall that time in the Feynman diagram flows from the bottom of the page to the top, while the width of the page depicts separation in space. One of the curious aspects of quantum physics, however, is that the choice of time's direction is arbitrary. Swap the direction of time, and Figure 4.1 equally well describes two photons colliding to produce an electron–positron pair.[5]

Given that time appears reversible at both the cosmic and the quantum level, why then does time to us appear to flow in one direction – that is, *why is there an arrow of time* that appears to point from the past to the future? This is an open question in both physics and philosophy.

One possible explanation is that time's arrow might be an illusion: perhaps time *does not* flow from the past to the future. Time's arrow might simply be a trick of consciousness. Perhaps time *is* consciousness, and all events in the past and future are already fixed in four-dimensional space. If true, this explains the pesky riddle of quantum entanglement – Einstein's spooky action at a distance – but it also closes the door on the possibility of free will. That is, the future might be fixed, but we simply aren't aware of how it will unfold. If the future is fixed, then everything that will happen has already happened, and we have already made all of our choices that we will ever make – we just don't know it yet. Although some people reject this explanation out-of-hand, anyone who has ever been surprised by the ending of a novel or a movie has experienced this effect first-hand.

4.2.2 The Second Law of Thermodynamics

Instead of resorting to metaphysical or religious explanations, physicists typically cite the Second Law of Thermodynamics as the expla-

[3] Gamma particles are highly energetic photons.

[4] *Star Trek* also featured the concept of teleportation – the Star Trek *transporter* – which we will revisit in Chapter 7, as well as one of the first popular depictions of computer forensics.

[5] Such reactions have never been observed, but there have been proposals for creating "gamma–gamma" colliders that would do just this.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

***Burnt Norton* (Excerpt)**

Time present and time past
Are both perhaps present in time future,
And time future contained in time past.
If all time is eternally present
All time is unredeemable.
What might have been is an abstraction
Remaining a perpetual possibility
Only in a world of speculation.

– T.S. Eliot (1936)

nation of time's arrow. The Second Law holds that the entropy of a closed system tends to increase with the passage of time. But this is a bit self-referential, since what we call the "Laws of Thermodynamics" aren't really laws at all – they are observations that physicists have made regarding how energy appears to move through the world around us.

The so-called Laws of Thermodynamics were worked out between 1850 and 1920 to explain the behavior of heat. They are "laws," not theories, because they describe *what* the scientists observed; they didn't try to explain the *why* behind the observations. And they aren't laws, because there is no penalty for violating them.[6]

The First Law of Thermodynamics holds that the energy of a closed system remains constant. The Second Law says when two objects touch, heat naturally flows from the warm object to the cold object and not the other way around. By the early twentieth century physicists had learned how to construct devices like heat pumps and refrigerators that use mechanical energy to move heat "uphill" – that is, to suck the heat out of cold objects to make them colder, dumping the energy someplace else, making that second place warmer. These devices don't actually violate the Second Law, however, when you take into account the entire system consisting of the object being cooled, the object being heated, the heat pump, and the energy source.

[6]The discipline of quantum thermodynamics derives the modern laws of thermodynamics from quantum mechanics, but since the "laws" of quantum mechanics are also simply mathematical equations that happen to fit observations made by physicists of the physical world, even these "laws" are not really laws.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

The Third Law of Thermodynamics says that no matter how hard you work, you cannot cool an object to absolute zero Kelvin (−273.15 °C, or −459.67 °F). In fact, the colder a system gets, the more energy is required to cool it further.

There are many formulations for the Laws of Thermodynamics. Although most are mathematical, one is lyrical: *You can't win, you can't break even, and you can't get out of the game.*

Today the Second Law is widely understood in terms of *entropy*. A colloquial definition of entropy is that it is the amount of "disorder" that exists in a system – the more disorder, the more entropy. Another way of stating the Second Law is that the entropy of a closed system will tend to increase over time.

If you have ever made tea, you have experienced the Second Law. Take an empty teacup, drop in a tea bag, and fill the cup with boiling water. At first, the various organic molecules that make up the tea are all located inside the tea leaves.[7] The tea bag, its leaves, and the cup are all cold, the water is hot. This is a highly ordered system.

But as soon as the water and the tea mix, the organic molecules inside the tea leaves start to diffuse into the hot water, and within a few minutes the concentration of the molecules that we call "tea" dissolved in the water and still present in the tea leaves are roughly in equilibrium. Likewise, the temperature of the tea bag and the inside of the tea cup both rise, while the temperature of the water falls, until they too are roughly in equilibrium. If you wait long enough, the less agreeable molecules from the leaves will also migrate into the water, and the temperature of the water, the teacup, and the room will all come into equilibrium, and now you have ruined a perfectly good cup of tea in the service of science.

You may have also heard that there is a finite probability that all of the air molecules in a room will move into a corner, resulting in the asphyxiation of everyone in the room. In practice this never happens, because that finite probability is fantastically small. Likewise, there is a finite probability that the heat in the room will move back into the water, and that the bitter tea molecules will move back into the bag. But this is also very improbable – so improbable that you will never experience it, no matter how many cups of tea you forget on your kitchen counter.

[7] For a discussion of molecules that make up tea, see C.-T. Ho, Zheng, and Lib, "Tea Aroma Formation" (2015).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

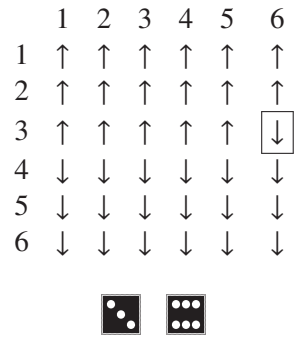


Figure 4.2. An exercise in entropy. The first roll rotates the arrow at row 3, column 6. Rotate enough arrows and the original pattern will be obscured.

You can also demonstrate the Second Law with 36 coins and a pair of dice. Place the coins in a 6-by-6 grid such that the top three rows show their heads pointed up and the bottom three rows show their heads pointed down. Once again, this is a highly ordered system – it's low entropy. Roll the dice and rotate the coin at the row specified by the first die and the column indicated by the second (see Figure 4.2). Repeat a hundred times, and you won't be able to see the original pattern. We have created a web-based version of this simulation that you can run at www.the-quantum-age.com/quantum-demos/.

Both tea diffusing into hot water and the coins rotating in accordance with dice rolls are randomized processes that are reversible in theory, but not in practice. This is probability at work. If you roll the dice twice and roll (3,6) followed by (3,6), you will end up with the original pattern. There is a 1 out of 36 (2.8 percent) chance that this will happen.[8] But if you role the dice *four* times, there are only 3888 sequences that will restore the original pattern, while there are 1 675 728 sequences of dice rolls that will not. The odds that a sequence of rolls will be such a restorative sequence grow *exponentially* worse with each additional pair of rolls. So while it is theoretically possible that you will one day see the initial checkboard pattern restored, the odds are vanishingly small. For example, the odds of restoring the board after six pairs of rolls is significantly

[8]Because we use the first die to represent the row and the second to specify the column, there are 36 distinct dice throws. For each of those 36 possible dice rolls, there is precisely one restorative sequence. Thus, there are a total of 36×36 dice rolls, of which 36 are restorative: $\frac{36}{36\times36} = \frac{1}{36} = 0.02\overline{7} = 2.7\%$.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Our Tools and Our Self-Conceptions

Einstein himself wrestled with the implications of quantum mechanics (see the sidebar "Man Plays Dice with Einstein's Words" on page 518), but for the average person these implications remain an abstraction in our day-to-day lives. As quantum technologies enter daily life, will we begin to see the world through the lens of quantum mechanics? After all, some ancients saw the universe as a geometric ballet, a reflection of the mathematics of the age. Clockwork and even steam technologies have served as metaphors to explain the celestial and our place in it. Consider this argument about our universe:

> The mechanism by which quantum mechanics injects an element of chance into the operation of the universe is called "decoherence." [...] Decoherence effectively creates new bits of information, bits which previously did not exist. In other words, quantum mechanics, via decoherence, is constantly injecting new bits of information into the world. Every detail that we see around us, every vein on a leaf, every whorl on a fingerprint, every star in the sky, can be traced back to some bit that quantum mechanics created. Quantum bits program the universe.[a]

How will we conceive of ourselves differently if the ideas in this book – the centrality of information and randomness – come to shape our worldview?

[a] Lloyd, "The Computational Universe" (2014).

less than the odds of winning any lottery on the planet. This is the Second Law at work, and it is all around us: time moves forward, eggs cannot be unscrambled, and people grow old. Feynman died of cancer, a disease caused by a random, uncorrected genetic mutation in a single cell. He was 69 years old.

4.2.3 Reversible Computation

There is a close relationship between the physics concept of entropy and the mathematical concept of information; in some formulations,

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

entropy and information are actually the same thing.[9] There is also a close relationship between the operation of conventional computers – classical computers – and entropy. Specifically, when classical computers operate, entropy increases. To give you some intuition as to why this might be the case, we will examine what happens in a classical computer when numbers are sorted.

Now we will explore a bit more of the hypothetical computer language that we explored in the last chapter. Recall that there are two kinds of information stored in the computer's memory: variables and code. Each variable has a name and an initial value. The code is executed one line at a time. Code can store and retrieve numbers from locations in its memory (as specified by variable names).

Let's see what happens when our hypothetical computer executes this pseudocode program:

```
PROGRAM SORT_NUMBERS:
BEGIN VARIABLES
A: 3
B: 2
C: 1

BEGIN CODE
1: IF A > B THEN SWAP( A , B)
2: IF B > C THEN SWAP( B , C)
3: IF A > B THEN SWAP( A , B)
4: PRINT-VARIABLES
5: HALT
```

To run this program, we load it into the computer. That sets the initial values of the variables and then runs the program one line at a time. When the computer stops, the output looks like this:

```
SORT_NUMBERS OUTPUT:
A = 1
B = 2
C = 3
```

This program sorts the variables `A`, `B`, and `C` and prints the result.[10] These variables are each a physical place inside the computer's

[9] Frank, "The Physical Limits of Computing" (2002).

[10] The program implements a simple sort algorithm called bubble sort. Although generally bubble sort is viewed as an inefficient sorting algorithm, it's fine here.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Step	PC	A	B	C
start	0000 0000	0000 0011	0000 0010	0000 0001
1	0000 0001	0000 0010	0000 0011	0000 0001
2	0000 0010	0000 0010	0000 0001	0000 0011
3	0000 0011	0000 0001	0000 0010	0000 0011
4	0000 0100	0000 0001	0000 0010	0000 0011

Figure 4.3. The value of each variable as the program SORT_NUMBERS runs

memory that can store a number. Our computer is a classical digital computer, so A is actually a set of bits. (See the footnote on page 85 and page 86 for a discussion of bits.) In our computer, A has 8 bits, and we encode them as an *unsigned 8-bit integer* (see p. 88).

It turns out that there's another variable in this computer program that we haven't mentioned yet. This variable is called the *program counter* (PC): it keeps track of the current line that the computer is executing. The PC starts at the first line (0000 0001), and ends on the fifth line (0000 0101). Figure 4.3 shows the value of each of the registers at the completion of each line of the program.

Bits are not abstract things: there is a physicality to each bit inside a computer. In the case of our hypothetical computer, each bit is built from a little bucket that can hold electrons. Each 1 corresponds to a small electronic charge and each 0 represents the absence of charge. In the case of this specific hypothetical computer, each bucket can hold between 0 and 400 electrons (see Figure 4.4).[11]

The bucket controls a switch that the computer uses to determine if the number of electrons in the bucket represents a 1 or a 0. If the bucket has no electrons, the switch is closed and the computer treats the bit as a 0. If there are more than 200, the switch engages, and the computer treats the bit as a 1. Every time the computer reads the bit, it then drains the bucket. If the computer reads a 1, it reloads the bucket back to its full capacity of 400 electrons. This read combined with a write is called a *refresh* operation, and forcing each bit to be either a 0 or a 1 is called the *digital discipline*, which

[11] The buckets actually hold *excess electrons*, since the bucket itself is made out of atoms, and each of those atoms also have their own electrons. However, it is easier to ignore the electrons that are part of the register's walls and just think about the excess elections.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

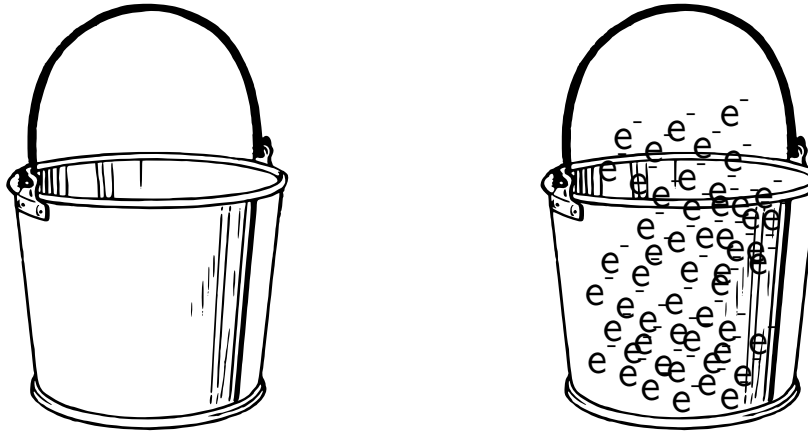


Figure 4.4. A bit in a computer's storage can be thought of as a bucket that can hold excess electrons. The bucket on the left holds no excess electrons and represents a 0. The bucket on the right holds 400 excess electrons and represents a 1.

we discuss in Section 3.3 (p. 84).[12]

From the First Law of Thermodynamics, we know that energy cannot be destroyed. When the computer starts up, the variables `A`, `B`, `C`, and `PC` are all `0000 0000`. The computer needs 4 bits (1600 electrons) to initialize `A`, `B`, and `C`. These electrons come from a massive reservoir of electrons called the computer's *ground*, which is drawn like this: ⏚.[13] Pulling those 400 electrons from the ground and dropping them into the buckets takes *work*. This work is performed using *energy* from the computer's power supply.[14]

As the computer program runs, electrons are being constantly sent from the memory back to the ground, and pulled back from ground into memory locations. For example, when the `PC` gets in-

[12]Readers with a background in electronics may realize that each bucket is actually a random access memory (DRAM) cell.

[13]On some computers, the computer's ground is actually connected to the third prong on of the electrical outlet – the ground prong – which connects to a green wire that eventually goes to the earth, hence the name *ground* on these computers is actually *the ground*! However, many computers these days don't have a wire connected to the earth. Instead, they have a *floating ground*, which is typically the negative terminal of a rechargeable battery.

[14]In a laptop or cell phone, the energy required to flip bits comes from a chemical reaction. In computers that are plugged into the wall, the energy might come from an electric dynamo powered by a wind turbine, or from photons sent to the Earth by the Sun.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

cremented from `0000 0000` to `0000 0001`, those electrons come from the ground. When it gets incremented from `0000 0011` to `0000 0100`, the extra electrons go back to the ground.

All of this work generates heat, which is why a laptop gets warm when it is worked hard. The heat comes both from chemical reactions in the laptop battery to make the electronic energy that's needed to move the electrons, and from the movement of the electrons through the computer circuits, which also generates heat because the electrical wires have resistance. Overcoming that resistance also takes work.

Aside from the program counter, the `SORT_NUMBERS` program is pretty efficient in terms of electrons. All of the data movements are done with the `SWAP` function, which swaps the values in the two variables.[15] Nevertheless, when this program runs, information is destroyed. We know this because after the program runs, we've lost the original values of the variables `A`, `B`, and `C`, and there's no way to get them back.

In fact, there are other hidden sources of energy loss going on. The swap itself requires no energy, but `IF` statements in lines 1, 2, and 3 all generate a bit of information (whether to swap or not to swap) and then destroy that bit. And we are ignoring all of the bits that are set and then cleared when the computer executes `PRINT-VARIABLES`. Conservatively, that program is probably destroying billions of bits every time it runs.

4.2.4 The Landauer Limit

In 1961, Rolf Landauer (1927–1999) at IBM Research considered the operation of computers at the information-theoretic level. Landauer concluded that practical computation required that information be destroyed, resulting in the inevitable increase in entropy. Landauer showed[16] that even in an ideal computer, every bit of information that is lost must generate a tiny amount of heat – at least 3×10^{-21} J at room temperature. Today this is called the *Landauer limit*.

This amount of heat was insignificant compared to the other processes running inside IBM's computers of the early 1960s; no IBM

[15] Most computers have such swap instructions, although in practice what they do is far more complicated and electronically expensive than simply exchanging electrons between two memory locations.

[16] R. Landauer, "Irreversibility and Heat Generation in The Computing Process" (1961).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

engineer had ever observed it. It's still insignificant today. A typical modern desktop computer consumes roughly 10×10^{-15} J to convert a `0` to a `1` – roughly a million times more than heat generated by the information loss.

Landauer became IBM's assistant director of research in 1965,[17] and became an IBM Fellow in 1969. In 1972, he recruited Charles H. Bennett (b. 1943) to join the research staff at Yorktown Heights. At the time, Bennett had been thinking about quantum information for nearly a decade (see the sidebar "The Birth of Quantum Cryptography" on page 137) and was working on a paper that challenged Landauer's fundamental finding – that is, he found a way around the Landauer limit. Bennett published that paper shortly after joining IBM.

Bennett's paper starts out by restating Landauer's conclusion:

> The usual digital computer program frequently performs operations that seem to throw away information about the computer's history, leaving the machine in a state whose immediate predecessor is ambiguous. Such operations include erasure or overwriting of data, and entry into a portion of the program addressed by several different transfer instructions. In other words, the typical computer is logically irreversible.[18]

But in the pages that follow, Bennett showed that Landauer had overlooked something: Landauer had assumed that computers necessarily had to destroy information when they operate. Bennett showed that this need not be the case: it is possible to compute entirely with *reversible* operations. Such a computer would be more complex than a computer built from conventional logic – computers like the ones that IBM was building in 1973 – but in theory could be just as powerful. That is, it would be a Turing machine, a generalized computer that can run any program and simulate any other computer. Bennett showed how to build a reversible Turing machine.

Bennett didn't actually *build* a reversible Turing machine, of course, any more than Alan Turing built a Turing machine when he published *On Computable Numbers*.[19] Bennett merely showed

[17] Physics Today, "Rolf Landauer" (2019).
[18] C. H. Bennett, "Logical Reversibility of Computation" (1973).
[19] Turing, "On Computable Numbers, with an Application to The Entscheidungsproblem" (1936).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

that it is *theoretically possible* to build such a machine. Bennett also showed that such a machine would be significantly more complicated to design, harder to program, and would typically take twice as many steps as a non-reversible Turing machine to solve the same problem. But a reversible Turing machine would have a significant advantage over today's non-reversible systems: it would be liberated from Landauer's limit, and be able to compute with essentially no lower bound on energy loss.

As will be shown later in this chapter, reversible computation is also the key to solving problems on quantum computers.

4.3 Cellular Automata and Conway's Life

Bennett was not the only person in the 1970s interested in reversible computing. Another was Tommaso Toffoli, who developed his approach for reversible computation using a different approach to computing called *cellular automata*.

A graduate student at the University of Michigan, Toffoli had studied physics in Italy before moving to the US as part of the Fulbright Foreign Student Program. He eventually met up with Arthur Burks (1915–2008), a mathematician who had worked on the design of the EDVAC with John von Neumann (see the sidebar "John von Neumann" on page 138). After von Neumann's death, Burks completed and edited von Neumann's final book, which introduced the idea of *cellular automata*.[20] and explores many of their theoretical capabilities.

With his background in physics, Toffoli was interested in taking the research of von Neumann and Burks in a different direction. Specifically, he wanted to know if it was possible to build a *reversible* cellular automata. Toffoli recalled in an interview for this book that Burks and others thought that it wouldn't be possible to create such cellular automata, but Toffoli showed that it was, and published the work as his PhD thesis, with Burks as his thesis advisor.

4.3.1 Computing with CPUs, GPUs, and CA(s)

To understand the significance of Toffoli's question, and of what he discovered, we are going to look deeper into how computation works in a conventional computer.

The "brain" of the contemporary computer is a small device called the *central processing unit* (CPU). Inside the CPU there is

[20] von Neumann and Burks, *Theory of Self-Reproducing Automata* (1966).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

The Birth of Quantum Cryptography

QIS pioneer Gilles Brassard (b. 1955) traces quantum cryptography's start to a friendship between Charles Bennett and Stephen Wiesner, who met while they were undergraduates at Brandeis University in the 1960s. Bennett went to Harvard to pursue his PhD, while Weisner went to Columbia University. Weisner came up with an idea he called "Conjugate Coding," which used a pair of entangled particles to do things like create electronic banknotes that would be impossible to counterfeit and create pairs of messages, of which only one could be read by the recipient. Wiesner submitted a paper on his thought experiment to *IEEE Transactions on Information Theory*, but the paper was rejected and Wiesner went on to other projects.[a]

The possibility of using entanglement for some kind of communication stuck with Bennett and he shared it from time to time with others. More than ten years later, Bennett and Brassard were at an IEEE conference in Puerto Rico, where Brassard was giving a talk that touched on quantum concepts. Bennett thought that Brassard might be interested in Weisner's idea of conjugate coding. Brassard was, and the two expanded the idea into the basic concept of "quantum cryptography," which they presented at the Crypto '82 conference. The following year, Bennett and Brassard presented their groundbreaking article, "Quantum Cryptography: Public Key Distribution and Coin Tossing"[b] (frequently called simply *BB84*). We will take up the story of quantum cryptography in the next chapter. And if you are interested in the original Conjugate Coding paper, you can read it too,[c] since the success of the BB84 convinced Wiesner to get his original paper published.

[a] G. Brassard, "Brief History of Quantum Cryptography: a Personal Perspective" (2005).
[b] C. H. Bennett and G. Brassard, "Quantum Cryptography: Public Key Distribution and Coin Tossing" (1984).
[c] Wiesner, "Conjugate Coding" (1983).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

John von Neumann

Born in Budapest in 1903, John von Neumann was one of the most gifted scientists of the twentieth century. At the age of eight von Neumann was familiar with calculus; fluent in five languages, he published his first groundbreaking mathematical paper at the age of 19. Eager to escape Europe, he was offered one of the first professorships at the Institute of Advanced Study in Princeton, NJ, which he joined in 1933.

Von Neumann worked out the complex nonlinear equations describing the physics of shock waves; this had direct application to the design of explosives. Based on this work, he was invited to join the Manhattan Project in 1943, where he worked on the explosive "lens" for the implosion bomb. He was successful: the first implosion "gadget" detonated at the Trinity test site on July 16, 1945; the second detonated over the city of Nagasaki, Japan, on August 9, 1945, killing as many as 80 000 people.

At a chance meeting at the Aberdeen train station in August 1944, army lieutenant Herman Goldstine told von Neumann of a research project at the University of Pennsylvania to create a device that would be able to compute artillery tables far faster than the human "computers" that had been hired for the task.[a] Von Neumann joined the group, hoping that the Electronic Numerical Integrator and Computer (ENIAC) under construction, along with its successor Electronic Discrete Variable Automatic Computer (EDVAC), would be able to speed the Los Alamos bomb computations.

Goldstine typed up the group's design notes and gave them to von Neumann for editing during a train ride to New Mexico. When the report was distributed later that summer, *First Draft of a Report on The EDVAC*[b] carried von Neumann's name alone on its cover. This mistake is memorialized in the term *von Neumann architecture*, which describes the EDVAC's approach of storing both data and code in the computer's main memory. Today von Neumann architectures dominate the computer landscape. Quantum computers do not have von Neumann architectures, but they are controlled by conventional computers that do.

[a] See Grier, *When Computers Were Human* (2007) and LeAnn Erickson's 2011 documentary *Top Secret Rosies: The Female Computers of WWII.*
[b] von Neumann, *First Draft of a Report on The EDVAC* (1945).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

a complex circuit where the actual computing – the addition, the subtraction, and so on – takes place. This circuit is literally the computer's processor, although on some computers it is called a *core*; until the early 2000s most home computers had a single core, whereas today most home computers have anywhere between two and twelve.[21] The rest of the computer exists to move data and code from the Internet into the computer's memory, and then from the computer's memory into CPU, and then to move the results back to the outside world.

Cellular automata take a different approach to computation. In these systems, computation takes place *in the memory itself*. Imagine a large rectangular grid of cells, like a massive checkerboard that extends to the horizon. Each square is a processor that has a small amount of memory and executes its small program in step with all of the other squares. Each square can also communicate with its neighbors. By itself, each square can't compute much, but the assemblage of all of the squares could be much faster than today's fastest computers, for the simple reason that more instructions are executing at any given moment. That is, whereas contemporary computers have between two and twelve cores, and whereas graphic processing units might have a few hundred or even a few thousand cores, a large system based on cellular automata principles might have millions or billions of cores.

The phrase *self-reproducing* in the title of von Neumann's last book asks not if it is possible to create a robotic factory that is programmed to produce robot factories, but if it is possible, using computation, to have an underlying mathematical pattern that can reproduce itself. Such a structure could be the core idea that empowered a robotic robot factory, but the underlying design pattern might show up in other systems as well.[22]

[21] Contemporary computers also typically have graphic processing units (GPUs), which can have dozens, hundreds, or even thousands of cores. These cores are less flexible than the cores in the CPU and are optimized for performing the kind of math necessary to render complex scenes. Each specialized GPU core is typically slower than a general purpose microprocessor core in the CPU, but the GPU has many more cores than the CPU, so the net result is that it runs much faster. Although, as their name implies, GPUs were originally created for graphical processing, another common use for GPUs today is performing the massive and repetitive mathematical algorithms required by contemporary artificial intelligence algorithms.

[22] Design patterns used in nature frequently show up in engineered systems, im-

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

That is, the self-reproducing automata that are the subject of von Neumann's book *could* be a factory of robots, placed in a complicated arrangement so that the factory of robots created new factories of robots. Alternatively, it might be a collection of math problems that, when solved, created a new set of *the same* math problems. This is fundamentally the advantage that von Neumann and Burks enjoyed by working with mathematical abstractions, rather than trying to actually build self-reproducing automata out of wires, relays, and engines: the abstract mathematical system allows the thinker to focus on the conceptually relevant part of the problem without worrying about the details. As theoreticians, they could consider their theoretical models and determine if the models would work (if they could possibly build the systems), or if the models wouldn't work (even if they spent their lifetimes trying to build the system perfectly). This interplay between theory and practice shows up again and again in the history of computing, and it is the reason why theoreticians believed that quantum computers would be so powerful even before the first quantum computer was ever constructed.

4.3.2 Life (The Game)

Probably the best known cellular automata is *Life*, invented by the British mathematician John Horton Conway FRS (1937–2020). Life is *not* reversible, but its influence is great to this day, so we use Life here to present the concept of cellular automata, which will then give us a tool for thinking about quantum computers.

Conway designed the rules of Life through trial-and-error; we present the rules in Figure 4.5. Conway's goal was to create a simple set of rules that nonetheless produced successive generations with unexpected complexity. Below we will look at a few simple examples that do not have such complexity, followed by two examples that remain fascinating to this day.

plying that the underlying requirements for both natural and engineered systems may share fundamental commonalities. For example, both bacterial and computer programs called *quines* are self-reproducing automata that are structured in two parts: the first part is the genetic material or information that describes the machinery necessary to reproduce, and the second part is the machinery itself, which reads the information and reproduces both the information and the machinery. A factory of computers that built computers would probably be based on similar principles. See also Bratley and Millo, "Computer Recreations: Self-Reproducing Programs" (1972).

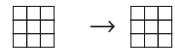
https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

The rules of Life:

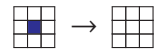
1. Gameplay is on a square grid of cells, such that each cell has eight neighbors.
2. Each cell can either be empty or alive.
3. There is a global clock. Each time the clock ticks, every empty cell that is surrounded by exactly three live cells transitions from empty to alive. (A "birth.")
4. Alive cells that have two or three live neighbors remain alive.
5. Alive cells with less than two alive neighbors become empty. (They die of "loneliness.") Alive cells with four or more alive neighbors become empty. (They die of "overpopulation.")

Figure 4.5. Rules for John Conway's "Life"

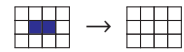
A grid with no live cells remains eternally empty:

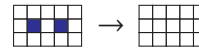


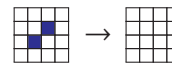
A single live cell also becomes empty and remains that way forever:



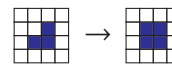
The three possible arrangements of two live cells also die out:





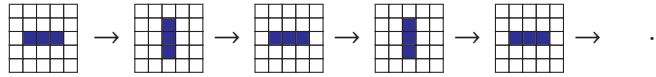


With three live cells there are three possibilities. A triangle of three live cells becomes a 2-by-2 square, which is eternally stable:

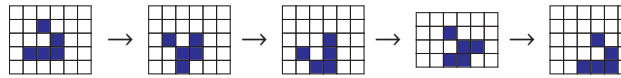


Three cells arranged in a diagonal will take two generations to die out. More exciting are three cells arranged in a horizontal row: they became a vertical row, which then became a horizontal row again. This repeating pattern is called a *blinker*:

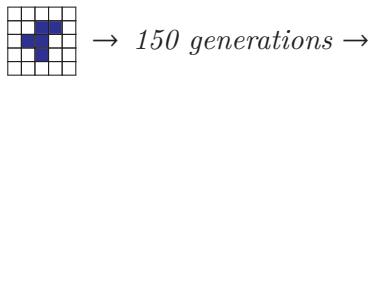
https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press



Start with five live cells and things get complicated fast. For example, there is the *glider*, which moves one cell to the right and one cell down every four generations, as demonstrated by this progression:



A slightly different collection of five cells called R-pentomino produces a staggering amount of complexity. The initial pattern runs without a repeat for 1103 generations and ends up producing eight 2x2 blocks, six gliders, four six-celled "beehives," four blinkers, and a collection of other objects. It must be watched on a computer screen to see this in all its glory. Below are the results at 150 generations using the web-based LifeViewer.[23] Look carefully and you can see that three of the pattern's gliders have already been launched and are sailing off to infinity:



Conway invented Life in 1969 and sent a typewritten letter about it to Martin Gardner (1952–2000), editor of the popular "Mathematical Games" column in *Scientific American*. Gardner featured the game a few months later,[24] igniting an interest in both Life and cel-

[23] See www.conwaylife.com/wiki/R-pentomino

[24] Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'" (1970).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

lular automata that continues to this day. Indeed, when Conway died in 2020 (one of the early notable deaths in the COVID-19 pandemic), the obituary in the *New York Times* quoted Gardner stating that at the peak of its popularity, "one quarter of the world's computers were playing [Life]."[25] The obituary also quoted musician Brian Eno, who said "Conway's LIFE changed mine … Conway himself thought it rather trivial, but for a nonmathematician like me, it was a shock to the intuition, a shattering revelation – to watch glorious complexity emerging from staid simplicity."

Life Is Turing Complete

Although this fact wasn't discovered for many years after its invention, the rules of Life have sufficient complexity that they are *Turing complete*. That is, with clever programming, the rules of Life and a starting configuration of sufficient complexity can implement the central processing unit of a computer that can read, execute, and modify its own program. This basic idea was created by Alan Turing, another English mathematician, in the 1930s.[26] Turing's great discovery was that a mechanical calculating device can compute *any computable function in all of mathematics* if it 1) can read instructions from a tape; 2) write new instructions back to the tape; 3) move the tape forwards or backwards; and 4) has logic for executing the instructions. This means that you could use a large grid running Conway's Life to compute the mathematical constant π (pi) to a million places if you wanted to. You could even use a grid running Life to simulate a top-of-the-line Intel microprocessor, which means that you could use it to run the Windows or Macintosh operating system, provided that you had a grid that was large enough.[27] (We discuss computing and what it means to be Turing complete in Section 3.4 (p. 91).)

Like all Turing Machines, the Life Turing Machine (LTM) has control logic, memory cells, and the ability to read and write to a massive "tape." One of the repeated patterns used by the Life Turing Machine is the *glider gun*, first developed by famed MIT hacker Bill Gosper (b. 1943), which repeatedly "shoots" gliders across the grid.

[25]Roberts, "John Horton Conway, a 'Magical Genius' in Math, Dies at 82" (2020).
[26]Turing, "On Computable Numbers, with an Application to The Entscheidungsproblem" (1936).
[27]Rendell, "A Universal Turing Machine in Conway's Game of Life" (2011).

The Life Turing Machine (LTM) uses the gliders to communicate between its various parts.

The LTM requires 11 040 Life generations for one Turing Machine cycle. Whether or not that is "slow" depends on how fast the underlying cellular automata runs: in a browser at 18 generations a second, it's slow from the point of view of a human watching the screen; on some kind of theoretical stringy fabric that can crunch 600 trillion generations per second (600 THz), it would be considerably faster than any computer in existence today. Likewise, if a cell in the Life array is the size of the array we show above, a LTM large enough to run a web browser would probably be larger than our planet. But if each cell in the Life array were on the order of 10^{-35} m – that is, a distance on the scale of the smallest quantum effects (see Appendix A) – then the entire computer would likely fit into the space of a single hydrogen atom.

Turning a massive, parallelized, conceptually clean cellular automata into an ornately complex contraption built from glider guns and mathematical tape may seem itself more like a mathematical diversion than a practical exercise in computing. The point of the exercise is to demonstrate that the underlying computational medium of Life's cellular automata is universal: it can therefore compute anything that is computable. Building a computer with glider guns and tapes is no more strange than building one with relays, tubes, or semiconductor transistors.

Where could one go with these observations? Recall Toffoli's interest in recasting physics as computation. Conway's Life is one of an infinite number of possible cellular automata systems, each with its own set of rules. A cellular automata could have rules that just consider each cell's north, south, east and west neighbors, for example. Cells could have a third state, *young*, which would prevent them from counting towards a birth. Cells could eventually die from old age. The game could be played on a hexagonal grid, or a three-dimensional grid, or even a five-dimensional grid. The key thing that makes it a cellular automata is that every cell follows a set of rules – typically the same rules – and that each runs more-or-less independently. Beyond that, everything is up for grabs.

Conway's Life demonstrates that even simple underlying rules can produce complex and unforeseen outcomes. Could our own reality be described by the rules of a cellular automata? What if the

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

fundamental stuff of the universe, deep down, actually *is* a cellular automata?

4.4 Digital Physics

The idea that reality itself might be nothing more than a program running on some cosmic computer was *not yet* a common idea among academics and science fiction authors in the 1970s. Nevertheless, it was increasingly clear that there was something fundamental about computation and information – not just at the societal level, but in the underlying fabric of biology and physics.

For example, there was the matter of life itself. In 1953 Watson and Crick published the structure of deoxyribonucleic acid (DNA),[28] the molecular basis of heredity, and started to unravel the entire process by which information encoded in DNA is synthesized into proteins. By 1970, scientists were increasingly comfortable with the idea that most (if not all) biological processes were based on the movement of *information* carried by molecules.[29]

Likewise, by the 1970s the philosophical implications of quantum mechanics – for example, whether Schrödinger's cat could be both alive and dead at the same time (see p. 523) – were increasingly being discussed and accepted outside the rarefied world of theoretical physics. In 1974, Stephen Hawking showed that quantum uncertainty causes black holes to radiate small amounts of energy – now called *Hawking radiation* – setting off what American theoretical physicist Leonard Susskind called "The Black Hole War"[30] over the question of whether or not information was destroyed by black holes or conserved in Hawking radiation.[31] So the idea that reality itself might be fundamentally based on information – that reality might *be* in-

[28] Watson and Crick, "Molecular Structure of Nucleic Acids: a Structure for Deoxyribose Nucleic Acid" (1953).

[29] The role of information in shaping the form of physical reality easily dates back to ancient Greece, where Heraclitus posed the question of whether or not the ship that Theseus had used to sail from Crete to Athens was the same ship after centuries afloat in Athenian harbor, despite the fact that all of its oars and timbers having been incrementally replaced over the years.

[30] Susskind, *The Black Hole War: My Battle with Stephen Hawking to Make The World Safe for Quantum Mechanics* (2008).

[31] Meanwhile in the popular press, the bestselling books Capra, *Tao of Physics: an Exploration of The Parallels between Modern Physics and Eastern Mysticism* (1975) and Zukav, *The Dancing Wu Li Masters* (1979) both drew similarities between quantum mechanics and eastern mysticism.

formation – wasn't necessarily so far-fetched, at least to those who thought about it.

Even if the underlying fabric of the universe is not actually a cellular automata, being able to describe it as such might give scientists a powerful alternative formulation for quantum physics. But in order to do that, the cellular automata certainly wasn't going to be the kind described by the rules of Conway's Life. That is because the Game of Life is not reversible, but physics is.

4.4.1 Edward Fredkin and Project MAC

Project MAC was established at MIT in 1963 to develop interactive computer systems and explore applications for their use.[32] Roberto Mario Fano (1917–2016) was the founding director of Project MAC, followed by the legendary J. C. R. Licklider (1915–1990), an American psychologist and computer scientist, who ran Project MAC from 1968 until 1971. Edward Fredkin (b. 1934) was Project MAC's third director, from 1971 until 1974, when Fredkin moved to California to spend a year learning quantum mechanics from Richard Feynman, with Fredkin teaching Feynman about computers in return. Fredkin's tenure as director was unlike the others, in that it was the only time that Project MAC (or its successors) had been run by a wealthy, ex-military, college drop-out who had made his fortune when his AI startup went public.[33].

Fredkin was born in southern California in 1934 into a family that once owned a chain of radio stores but lost them at the start of the

[32]"MAC" was an unstable acronym, variously standing for "Multiple Access Computer" (the project pioneered timesharing, allowing a computer to be accessed by more than one person at once), "Machine-Aided Cognition" (one of the project's original goals), "Man And Computer" (and later "Men Against Computers," because the project's members were overwhelmingly male and the computers somewhat buggy), and even "Minsky Against Corby" (recognizing the long-running feud between MIT professors Marvin Minsky and Fernando José Corbató – a stress that ultimately led Minsky's Artificial Intelligence Lab to break with Project MAC and go its own way). Following Fredkin's tenure, Project MAC was renamed the Laboratory for Computer Science and run by Michael Dertouzos from 1974 to 2001, and then by Victor W. Zue from 2001 to 2003, at which point the Laboratory for Computer Science and the AI Lab merged back together to form the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) MIT Institute Archives, "Laboratory for Computer Science (LCS)" (2011).

[33]Garfinkel (aut.) and Hal Abelson (ed.), *Architects of The Information Society* (1999).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Quantum Physics and Free Will

We know that many experts and organizational systems can embrace probabilities in a contingent world, but does that embrace have limits? Moving from the level of legal systems to the individual, quantum technologies could erode our assumptions about human morality.

Our assumptions about human morality are based in nondeterminism – one implication of which is that we have free will, that our choices are ours, along with the moral responsibility of them. Could more familiarity with quantum mechanics begin to alter our assumptions about determinism and ultimately, assumptions of free will?

Novelist Ted Chiang writes an exhilarating story that explores the moral responsibilities of a many-world universe in *Exhalation.*[a] In the story, Chiang imagines a version of the many worlds theory, one where the universe splits and is duplicated every time quantum decoherence occurs. In Chiang's world, people can consult an oracle that reveals how they acted in other worlds split from one's own by quantum decoherence. One character regrets an act, consults the oracle, and finds that other versions did not engage in the bad act. He thus concludes that his bad act in this world was an anomaly, one that does not stain his character too deeply, because in other worlds, he took a different set of actions. The philosophy of personal responsibility is woven together amongst these series of different worlds according to this character. Others however are crushed by the events in alternate worlds and regret the actions taken in their own world. If people begin to see their lives as deterministic, as one version of themselves in a reality of many versions, might they start to believe that they are not really responsible for their acts in this world?

[a] Anxiety is the Dizziness of Freedom in Chiang, *Exhalation* (2019).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Great Depression.[34] That is, they had known money, but now they were poor, and even though Fredkin hadn't experienced wealth himself, the family's loss nevertheless affected him deeply. Fredkin grew up experimenting with electricity and chemicals, got poor grades in high school, but got accepted to the California Institute of Technology on the strength of his entrance examinations. CalTech did not give Fredkin any financial aid, so Fredkin worked multiple jobs. It still was not enough money, and his grades were still terrible, so in his second year he dropped out of CalTech and volunteered to be an Air Force officer – it was better than the alternative of being drafted to serve in the Korean War.[35]

The Air Force first trained Fredkin to fly jets, then to be an intercept controller. "It's like air traffic control, except we're trying to get them to the same place at the same time," he later explained.[36] After that, the Air Force sent Fredkin to MIT's Lincoln Laboratory to learn about computers so he could oversee the testing of the newfangled computerized air defense systems that were then under construction. It turned out that Fredkin was quite good with computers: when the Air Force had trouble with a computer at MIT's Haystack Observatory that was tracking rocket launches, Fredkin was sent to figure out what was going wrong. (He found an overflow error in the computer's programming.) Fredkin also created one of the first computer assembler languages, and then taught a course at Lincoln on how to use it.[37]

Fredkin got along well with computers, but not with the Air Force. He left military service and took a civilian job at Lincoln Lab working with the same computers. But Fredkin had bigger plans. On his own, he placed an order for one of the world's first commercial computers, a Royal McBee Librascope General Purpose 30, which was a tube-based machine first manufactured in 1956 that had a retail price of $47 000 (equivalent to $465 000 in 2021). Fredkin recalls that he only had $500 to spare, but the computer had a long delivery time, so he figured that he would find the money before he needed to pay up. His plan was to offer programming courses at area companies so that they could then provide contract programmers to

[34]Wright, "Did The Universe Just Happen?" (1988).
[35]Fredkin, interviewed by Garfinkel in September 2020.
[36]E. F. Fredkin, "Oral History of Ed Fredkin" (2006).
[37]Walden, "Early Years of Basic Computer and Software Engineering" (2011), p. 52.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

the government, use the tuition money to pay for the computer, and make a profit in the process.

Fredkin made a list of his prospects; at the top of the alphabetized list was Bolt Beranek & Newman (BBN), a 10-year old MIT spin-off specializing in contract research. At BBN Fredkin met Licklider, who soon convinced Fredkin to drop his plan to be an itinerant teacher-with-a-computer and instead join BBN's research staff. Licklider then convinced BBN to assume Fredkin's purchase commitment for the LGP-30, at the reduced price of $30 000. BBN had no obvious need for the machine; Licklider pushed. "If BBN is going to be an important company in the future, it must be in computers," Licklider told Leo Beranek, one of the company's founders. Beranek agreed to the purchase, even though BBN had "never spent anything like that on a single research apparatus."[38,39]

BBN soon acquired a second computer, the PDP-1, which it leased from another MIT spin-off called Digital Equipment Corporation (alternatively shorted to Digital or DEC over the company's 41-year life).[40] Not a full-size computer like the ones sold by IBM, DEC called the PDP-1 a *mini-computer*. This was right around the time that Project MAC was getting started at MIT, and Fredkin was convinced that the PDP-1 could be logically partitioned into four *even smaller pieces* so that the single machine could serve multiple people at the same time, an approach called *time sharing*. At Fredkin's suggestion BBN brought in two MIT faculty as consultants: Marvin Minsky (1927–2016) and John McCarthy (1927–2011) – two of the computer scientists who had coined the phrase "artificial intelligence" just a few years earlier.[41] Working together, Fredkin

[38]Beranek, "Founding a Culture of Engineering Creativity" (2011).

[39]BBN *did* become an important company in the future. The company designed and produced the Interface Message Processors (IMPs) that routed packets on the ARPANET and early Internet. It also created and spun off Telenet, Inc., the company that built and sold service on the world's first public packet-switched network. BBN was variously publicly traded and private, and was ultimately acquired by Raytheon in 2009 for $350 million. A major player in quantum technologies, with scores of academic publications along with applied research into photonics, superconducting qubits, graphene, control systems, and cryogenic systems, BBN now holds over 20 patents in quantum technologies.

[40]DEC eventually created 53 PDP-1 computers; BBN got the first. Another was given to the MIT for students to use; in 1962 Steve Russell and others used it to create the video game SpaceWar!

[41]McCarthy et al., "A Proposal for The Dartmouth Summer Research Project on Artificial Intelligence" (1955).

and McCarthy successfully implemented time sharing on the PDP-1. Fredkin also experimented with cellular automata on the PDP-1's graphics screen.[42]

Still focused on getting rich, Fredkin left BBN in 1962 and founded Information International Incorporated, an early AI startup. Minsky and McCarthy joined the board as founders. Triple-I, as it was known, did early work with the LISP programming language and in robotics, but the company's ultimate success came after Fredkin designed and the company started selling the first high-resolution film scanner for motion picture film. Fredkin took the company public six years later, becoming rich in the process. (Triple-I was eventually acquired by Agfa-Gevaert in 2001.[43]) With his newfound wealth, Fredkin would ultimately purchase a mansion, an island in the British Virgin Islands, and a television station.

Licklider left BBN in 1962 to head the Information Processing Techniques Office (IPTO) at the US Department of Defense Advanced Research Projects Agency (ARPA, later renamed DARPA), where he put in place research projects that directly led to the creation of the Internet. He worked at IBM from 1964 to 1967, and rejoined the MIT faculty in 1968 as Director of Project MAC.

Fredkin rejoined MIT the same year as Licklider and also went to work for Project MAC, although the two events were not connected. Fredkin was recruited by Minsky, with whom he had formed an enduring friendship, to be the AI Lab's co-director. The idea was for Fredkin to help steady the lab, using his combination of technical skills and business acumen. In 1972 Fredkin became Project MAC's director, and was promoted to full professor (perhaps in an attempt to erase the embarrassment of having the lab run by a college dropout who didn't have a PhD).

Running Project MAC did not suit Fredkin. He soon hired his own replacement, then moved out to California for a year-long sabbatical, spending the 1974–1975 school year back at Caltech, this time as a Fairchild Distinguished Scholar at the invitation of Richard Feynman.[44] Upon returning to Boston, Fredkin resumed his profes-

[42] Wolfram, *A New Kind of Science* (2002), p. 876.

[43] Wright, "Did The Universe Just Happen?" (1988).

[44] Minsky introduced Fredkin to Feynman back in 1962 – three years before Feynman won the Nobel Prize, when Feynman was considerably less famous. "Feynman showed us a mathematical problem he had been working on. He had a notebook and the notebook had all these pages of mathematics," Fredkin recalled in

sorship at the Project MAC, which had been renamed the Laboratory for Computer Science, and continued working on the project he had started in California with Feynman: reversible computing.

4.5 Reversible Computing and Supercomputing

The basic idea of a reversible computer is that it is a computing machine that can go forwards or backwards in time for any sequence of computations. We discussed the idea of reversible computers earlier in this chapter while exploring Bennett's idea of a reversible Turing machine (p. 135), but it's not clear if Fredkin or Feynman were aware of Bennett's work at IBM.

4.5.1 A Most Successful Term Paper

Instead of building his reversible computer using the theoretical mathematical constructs of a Turing machine, Fredkin's reversible computer reflected his own practical orientation. His first approach was a model of a computing machine based entirely on billiard balls careening around a friction-less obstacle course and having perfectly elastic collisions. He called this the *billiard ball computer* and ultimately published the idea in 1982.[45] You can't actually build such a computer, of course, because we don't have frictionless billiard balls that undergo perfectly elastic collisions. That's why the computer

our interview. "He said, 'Look – this mathematical problem is something we need to solve. I tried to solve it, a graduate student also did it.' He showed us – he had a notebook with about 50 pages of dense mathematics in it, handwritten, and he kept circling great big expressions and giving them names. And he said, 'Look, I've done all the math here, and I get a final expression. Murray Gell-Mann has also done it, and a graduate student has done it, and all we know is that the three of us got three different results that are not compatible. So our conclusion is that no one can do this much mathematics without doing errors. Can you guys do something about it?' " (Murray Gell-Mann (1929–2019), was awarded the 1969 Nobel Prize in Physics "for his contributions and discoveries concerning the classification of elementary particles and their interactions.") When Minsky said that symbolic algebra was a problem that the lab was working on, Feynman added that he refused to type on a computer, so the symbolic algebra system *also* needed to be able to read his handwriting and convert it to computer notation. On the flight back from Los Angeles, Minsky said that he would have a graduate student work on the algebra, and Fredkin would work on the handwriting recognizer. In retrospect, the Minsky–Feynman–Fredkin meeting didn't result in any breakthroughs in handwriting recognition or symbolic math computation, but it did set the groundwork for the invention of quantum computing two decades later.

[45] E. F. Fredkin and Tommaso Toffoli, "Conservative Logic" (1982).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

is just a theoretical model: it's a way for thinking about building a reversible computer without actually having to build one.

The actual reversible computer that Fredkin proposed building would be built out of semiconductors. To do that, Fredkin needed a new set of basic circuits that themselves were reversible, and that could be used to build a reversible computer. Today we call Fredkin's basic circuit the *Fredkin gate*.

The Fredkin gate has three inputs (C, I_1, and I_2), and three outputs (C, O_1, and O_2). The fact that the number of inputs matches the number of outputs is not an accident: it is required by the basic rules of reversibility. That is, every input to the gate must have a unique output: this makes it possible to run the gate backwards for any output and learn its original input. Eight possible inputs with eight corresponding outputs is the smallest number of combinations that produces a device that is both reversible and universal.

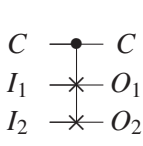
In addition to being reversible, The Fredkin gate is *universal*, in that any digital circuit can be built from a combination of Fredkin gates. (In today's computers, the NAND gate is sometimes used as a universal building-block, because any electronic circuit can be built using a combination of NAND gates. See Section 3.3.2 (p. 90).) Because it is a binary logic gate, each of the inputs and outputs can be either a `0` or a `1`. If C is `0`, the output bits are each the same as the corresponding input bits. If C is `1`, then the output bits are swapped. The Fredkin gate is thus also called a *controlled swap*, or CSWAP. It is shown in Figure 4.6.

Tommaso Toffoli completed his dissertation in 1976 and submitted a journal article proving that reversible automata could be constructed; the article was published the following year.[46] Toffoli recalls interviewing with Charles Bennett at IBM and with Fredkin at MIT and decided to become a Research Scientist in Fredkin's group, which he joined in 1977.

The following spring Fredkin taught an eclectic graduate course at MIT called Digital Physics (Figure 4.4). The course consisted of Fredkin sharing his intuition about the nature of reality with graduate students, and then trying to get students to develop formal mathematical proofs of these conjectures as their final projects. One of those projects was Bill Silver's term paper, "Conservative Logic,"

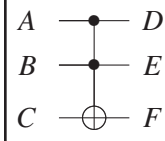
[46] Tommaso Toffoli (1977).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press



C	I_1	I_2	C	O_1	O_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

Figure 4.6. The Fredkin gate (CSWAP)



A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Figure 4.7. The Toffoli gate (CCNOT)

in which Silver worked out detailed proofs regarding the properties of Fredkin's gate.[47]

In 1980, Toffoli came up with an improvement to the Fredkin gate that is somewhat better suited for designing complex circuits. Today it's called the Toffoli gate. Whereas the Fredkin gate is called a controlled swap (CSWAP), the Toffoli gate is called a *controlled controlled NOT* (CCNOT).[48] This gate is shown in Figure 4.7.

Like the Fredkin gate, the Toffoli gate is also universal, meaning that it can be used to create any kind of digital electronics currently in use (or imaginable, for that matter). Both gates can also be generalized to more than three inputs. In practice Toffoli gates are used more often than Fredkin gates when discussing quantum circuits, perhaps because they offer more flexibility.

4.5.2 Reversible Computing Today

Heat was not a major concern for most computers in the 1980s, but it is today. Nevertheless, mainstream computer companies are not building their conventional systems with reversible logic. Here are some reasons why they aren't:

[47] Silver left MIT in 1981 to join his classmate Marilyn Matz and MIT Lecturer Dr. Robert J. Shillman in a startup venture called Cognex Corporation, which sought to develop and commercialize computer vision systems. Cognex went public in 1989 and is currently listed on the NASDAQ as CGNX with a market cap of $14B.

[48] In a controlled NOT gate, a control bit determines whether a data bit is inverted or not. In a controlled controlled not (CCNOT), *both* control bits need to be 1 in order for the data bit to be inverted.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

DIGITAL PHYSICS
Edward Fredkin
January 17, 1978

6.895 Digital Physics
(New)

Preq.: Permission of Instructor
Year: G(2)
3-0-9

An inquiry into the relationships between physics and computation. 6.895 is appropriate for both computer science and physics students. Models of computation based on systems that obey simple physical laws and digital models of basic physical phenomena. Tutorial on conventional digital logic. Information, communication, memory and computation. A formal model of computer circuitry, conservative logic, will be used to model computers at various levels of complexity from simple logic gates to processors, memory, conventional computers and Turing machines. Questions about reversibility and about the conservation of information during computation. Minimum energy requirements for a *unit* of computation. Generally reversible iterative processes. Tutorial on some areas of the quantum mechanics. Digital time and space. Universal cellular automata. Digital model of the zero-dimensional Schrodinger equation. Proof of the conservation of probability in the digital model. Three dimensional digital Schrodinger equations. Digital Newtonian mechanics. Digital determinism. The laws, physical constants and experimental tests of digital physics. Atomism. Questions of the ultimate nature of reality. Metaphysics and cosmogony.
E. Fredkin

Figure 4.8. Announcement for Fredkin's Digital Physics course.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

- The computer industry has nearly a hundred years' experience working with computer designs that are not reversible, while there has been comparatively little work done with reversible computing. The switching cost of moving from our current technology stack to a new one would be substantial, even if this other stack offers theoretical advantages. Similar switching costs are observed in other industries, such as the nuclear industry's failure to shift to a thorium-based fuel cycle, or the failure of the US to shift to the metric system.

- Although computers do convert electrical energy into heat when those `1`s are sent to ground, a significantly larger source of wasted energy is from semiconductor effects such as *resistance* (the fact that semiconductors do not perfectly pass electricity) and *leakage* (the movement of charge from one electronic device to another in a manner not aligned with the electronic circuit). What's more, leakage gets worse as transistors get smaller, placing a limit of just how small silicon electronics can get. Another limiting factor is the wires that carry signals between semiconductor devices: they have both resistance and capacitance, which again limits how energy-efficient, and how fast, signals can be carried between devices.

- Reversible computing requires more than reversible gates: it requires replacing large chunks of the technology stack. For example, there is a need to develop efficient reversible algorithms, presumably written in new computer languages that support reversible computing.

- Reversible computers require more transistors than traditional computers because they need to retain all of the information necessary to reverse the computation.

- Given that the computing industry hasn't hit the limits of non-reversible technology, there has been no reason to pursue reversible computing. Instead, the industry has exploited other approaches – most notably parallel computing – to achieve the significant speedups we have experienced over the past four decades. Whereas in the 1990s it was common for desktops and laptops to have a single CPU, today systems typically have between four, eight or even more.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

- An even bigger speedup has taken place on the other side of the Internet, in the "cloud" that delivers web pages to a desktop computer or information to applications running on a smart phone. Cloud computing has made it possible for each query to use hundreds or thousands of computers for an instant, getting a tremendous speedup.[49]

While reversible computing doesn't currently make sense for electronic computers, it is an area of active research. Meanwhile, reversibility is a basic requirement of computing on a quantum computer. The reason has to do with entanglement and superposition: the quantum part of a quantum computation stops when the wave function collapses, which happens the moment a non-reversible action takes place and a measurement is performed. So a quantum computer that implements any sort of logic has to use reversible logic *by necessity*.

Today it is common for quantum computer engineers to express the complexity of their algorithms in terms of the number of Toffoli gates that their algorithm and problem require, just as electronic computer engineers describe the complexity of their systems in terms of the number of electronic NAND gates or transistors. For example, in 2019 Google released a paper describing an approach for factoring the large integers used in cryptography *in hours*, stating that such a machine would require twenty million state-of-the-art (e.g. "noisy") qubits, and "$0.3n^3 + 0.0005n^3 \lg n$ Toffolis."[50] With a standard encryption key size, $n = 2048$, this comes to roughly 2.6 billion Toffoli gates.

While that may seem like a lot of gates, in November 2020 the Apple M1 system-on-chip contained 16 billion transistors.[51] Although the two kinds of gates are fundamentally different, the comparison shows that it is within the realm of today's technology to build a device with billions of active components. We will return to Google's paper in the next chapter.

[49] For example, in 2010 a single search at Google used more than a hundred computers, but each for just two-tenths of a second. See Dean, "Building Software Systems At Google and Lessons Learned" (2010).

[50] Gidney and Ekerå, "How to Factor 2048 Bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits" (2019).

[51] Apple Computer, "Apple Unleashes M1" (2020).

4.5.3 Defense Money

What made all of this research possible was a spigot of money from the US Department of Defense flowing into MIT's various computing projects during the 1960s and 1970s. This is not a new story, of course. The first computers built in Germany, England, and the US were all built to help with the war effort. It was the awarding of the SAGE missile defense system to IBM that cemented the company's position as the dominant computer manufacturer in the world. In 1961 IBM built its first transistorized supercomputer, the IBM 7030 "Stretch," for the US National Security Agency, apparently to assist in some way in the business of code-cracking. By the 1970s investments in supercomputing were helping to make sophisticated stealth aircraft a reality and to make the mathematical modeling of nuclear explosions so accurate that the US was able to stop physically testing nuclear weapons.[52]

Even before the simulations and models became crazy accurate, conducting physics experiments inside a computer had many advantages that made them a strong complement to experiments conducted in the lab or in the deserts of Area 51. Three such advantages are speed, scalability, and repeatability:

- **Speed** is the most obvious advantage: in the world of a computer, setting up a new experiment typically means editing a few files and reserving time on the computer system. This makes it easy for scientists to try a wide range of different ideas.

- **Scalability** means that scientists can run more experiments in a period of time simply by buying more computers. Scalability is not so easy in the lab, where running multiple experiments at the same time means having more lab space, as well as having more flesh-and-blood researchers to conduct the experiments.

- **Repeatability** is an often-overlooked advantage of conducting experiments in simulation. With complex experiments in a physical lab it is often difficult to repeat the experiment and get nearly the same result. This is because the outside world is always intruding. A truck may drive by, causing the ground to

[52] The US signed the Comprehensive Nuclear Test Ban Treaty on September 27, 1996, in part because the computer modeling had become so powerful as to make testing itself obsolete.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

> vibrate; a solar flare may eject a shower of high-energy atoms, ions, and electrons into space, causing a light show in the northern sky and interfering with sensitive electronic instruments down here on Earth. All of this must be taken into account when conducting physical experiments. With computerized experiments, the only real risk is bugs in the software.

Realizing these goals requires machines that are easy to program, reliable, secure, and accessible – hence the government's interest in funding basic research into software design, operating systems, security, and networking. The world we live in today – the hardware and software that was used to write the book you are reading – are direct beneficiaries from these government funding decisions.

A key to enabling the creativity and productivity of this basic research was the way that the funding agencies gave the researchers flexibility to set their own agenda. At MIT, the Laboratory for Computer Science and the Artificial Intelligence Laboratory were funded in no small part by a series of master agreements with DARPA, such as Office of Naval Research contract N00014-75-C-0661, which moved millions of research dollars from Washington to Cambridge. The money was delivered as a block grant, with individual faculty members needing to simply write project proposals describing what each planned to do with their share of the pie. As long as the faculty projects advanced the overall goal of building computers that were faster, better at solving problems, or easier to program, funding was all but guaranteed.

In November 1978, Fredkin and Toffoli included in MIT's proposal to DARPA a 20-page project description titled "Design principles for achieving high-performance submicron digital technologies."[53] The proposal expanded the ideas of conservative logic, showing how it would be possible to use reversible gates to cheat the power loss associated with conventional digital electronics. It then proposed approaches for using even less power, such as using superconducting switches with Josephson Tunneling Logic (also called *Josephson junctions*). The only mention of cellular automata was a reference to Toffoli's 1977 journal article, and while the proposal mentions Landauer's work, it doesn't mention Bennett's. But it does

[53] Twenty-four years later, the proposal was finally published (E. F. Fredkin and Tommaso Toffoli, "Design Principles for Achieving High-Performance Submicron Digital Technologies" (2001)).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

cite Fredkin's unpublished lecture notes from 1975–1978, and Bill Silver's MIT term paper. If nothing else, the proposal shows scientific progress is not linear, and the mere fact that scientific work has been published is no guarantee that others working in the exact same field will see it (or at least take notice of it) in a timely manner.

Fredkin and Toffoli's proposal was funded (likely a foregone conclusion), marking the beginning of the group's support by DARPA.

4.6 The Conference on The Physics of Computation (1981)

In the 1930s H. Wendell Endicott (1880–1954), a successful industrialist and philanthropist,[54] built a French-style manor house on a hill crest of his 25-acre suburban estate overlooking the Charles River in Dedham, Massachusetts. Endicott's will stated that the house should be donated "to an educational, scientific or religious organization." The property was offered to MIT when Endicott died, and the Institute turned it into a luxurious conference center.

When academics start developing a new field, it's common to hold some kind of meeting for early innovators to meet and exchange ideas. Always thinking big, in 1980 Fredkin decided to hold a conference at Endicott House and invite the biggest names he could get in physics and computing to discuss his up-and-coming ideas. Fredkin knew that he would need to have a big name to get the other big names to come, so he called up Richard Feynman, who agreed to give a keynote speech. Fredkin invited IBM Research to co-sponsor the conference. Rolf Landauer readily agreed, and both he and Charles Bennett agreed to attend.

Fredkin, Landauer, and Toffoli were the official organizers. Then came the invitations! Fredkin had earlier met Konrad Zuse, the German inventor who had built one of the world's first digital computers during World War II (see Chapter 3), so Zuse got an invite. The prominent physicists Freeman Dyson and John Wheeler were invited. Also invited were a number of up-and-coming researchers, including Paul Benioff (b. 1930), who went on to create the first mathematical model of a quantum computer; Hans Moravec (b. 1948), best known now for his work in robotics and artificial intelligence, and his writings as a futurist; and Danny Hillis (b. 1956), who went on to create the supercomputing company Thinking Machines, after which he became a Fellow at Walt Disney Imagineering. In total

[54] MIT Endicott House, "Our History" (2020).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

1 Freeman Dyson
2 Gregory Chaitin
3 James Crutchfield
4 Norman Packard
5 Panos Ligomenides
6 Jerome Rothstein
7 Carl Hewitt
8 Norman Hardy
9 Edward Fredkin
10 Tom Toffoli
11 Rolf Landauer
12 John Wheeler
13 Frederick Kantor
14 David Leinweber
15 Konrad Zuse
16 Bernard Zeigler
17 Carl Adam Petri
18 Anatol Holt
19 Roland Vollmar
20 Hans Bremerman
21 Donald Greenspan
22 Markus Buettiker
23 Otto Floberth
24 Robert Lewis
25 Robert Suaya
26 Stand Kugell
27 Bill Gosper
28 Lutz Priese
29 Madhu Gupta
30 Paul Benioff
31 Hans Moravec
32 Ian Richards
33 Marian Pour-El
34 Danny Hillis
35 Arthur Burks
36 John Cocke
37 George Michaels
38 Richard Feynman
39 Laurie Lingham
40 P. S. Thiagarajan
41 Marin Hassner
42 Gerald Vichnaic
43 Leonid Levin
44 Lev Levitin
45 Peter Gacs
46 Dan Greenberger

Photo courtesy Charles Bennett.

Figure 4.9. The Physics of Computation Conference, MIT Endicott House, May 6–8, 1981

roughly 60 researchers attended. Financial support for the conference was provided by the MIT Laboratory for Computer Science, the Army Research Office, IBM, the National Science Foundation, and the XEROX Corporation.[55] Norman Margolus, a PhD student in Fredkin's group, recorded and took notes of every lecture. These notes were then turned into articles and eventually published.

Before the conference, Feynman told Fredkin that he refused to focus his keynote on computers and physics, because computers and physics had nothing to do with each other. Physics is all about probability and randomness, Feynman said, whereas the whole goal of computing for the previous 50 years had been building machines that were reliable and predictable – the very opposite. Fredkin told Feynman that he could talk about anything he wanted, just come.

[55] E. Fredkin, Rolf Landauer, and Tom Toffoli, "Physics of Computation" (1982).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Fredkin recalls that when Feynman got up, the physicist started telling the story of how Fredkin had invited him to talk about computation and physics, and that he had refused to do so. "And I've changed my mind, and I'm going to talk about what he originally wanted," Feynman reportedly said in his matter-of-fact way.

Feynman's talk at the Endicott conference marks the birth of quantum computing, an idea that was unknowingly conceived by Feynman and Fredkin during Fredkin's year-long sabbatical at CalTech. It was a crazy idea. At roughly the same time that computer engineers were worrying that quantum mechanical effects in the form of quantum tunneling and uncertainty might pose real limits to computation by making machines act nondeterministically, Feynman proposed embracing the nondeterminism of quantum mechanics to build computers that could solve a problem that was simply too complicated to solve any other way – and that problem was quantum physics itself.

Feynman started his talk with a straightforward question: "What kind of computer are we going to use to simulate physics?"[56] After briefly suggesting that such a computer should have elements that are *locally connected* (like a cellular automata or a Thinking Machines' Connection Machine), he showed that the probabilistic nature of quantum physics means that quantum physics simulations necessarily have exponential complexity. The only way around this, Feynman said, was by using computing elements based on quantum mechanics itself, because the quantum wave equations would then match the systems that they were simulating. (Feynman says a lot of other things in his talk as well, but that's the gist of it.)

The rest of the conference was a fun mix of physics and computer science. Toffoli delivered a talk suggesting that physics might receive fresh insights from computing if computing is modeled with reversible computation.[57] Paul Benioff discussed and further developed his model of quantum mechanical Turing machines.[58] Fredkin and Toffoli significantly extended Bill Silver's MIT term paper and presented their ideas on Conservative Logic.[59] Danny Hillis presented his ideas on how to build massive computers using mesh networks

[56] Feynman, "Simulating Physics with Computers" (1982).
[57] Tommaso Toffoli, "Physics and Computation" (1982).
[58] Benioff, "Quantum Mechanical Hamiltonian Models of Discrete Processes That Erase Their Own Histories: Application to Turing Machines" (1982a).
[59] E. F. Fredkin and Tommaso Toffoli, "Conservative Logic" (1982).

with only local connectivity and routing – the basis of the Connection Machine that he was building.[60] Landauer discussed the impact of Heisenberg's Uncertainty Principle on the minimal energy requirements of a computer.[61] Marvin Minsky speculated that if the vacuum of the Universe is composed of discrete "cells, each knowing only what its nearest neighbors do," then "classical mechanics will break down … and strange phenomena will emerge" such as the phenomena described by both relativity and quantum mechanics,[62] possibly pointing the way towards a theory of quantum gravity. Other contributions included those by Donald Greenspan,[63] and John Wheeler,[64] all of which appeared in two successive issues of the *International Journal of Theoretical Physics*. It was not a top journal, but it was the best peer-reviewed journal that would take the collection.

4.7 Russia and Quantum Computing

Invention is rarely a straight line, and insight rarely comes in a single flash. It is common for good ideas to be invented and re-invented.

As we have seen, Toffoli and Fredkin were developing reversible logic at roughly the same time that Paul Benioff developed the idea of a quantum Turing machine.[65] These academics soon found each other, thanks to the milieu of papers, conferences, phone calls and email that American academics enjoyed in the 1970s and 1980s.

What about on the other side of the Iron Curtain?

Historians of quantum computing frequently point out that in Russia, R. P. Poplavskii wrote a 1975 Russian-language article, "Thermodynamical Models of Information Processing"[66] in which it was observed that classical computers would be insufficient for simulating quantum systems that do not have a simple solution: "The quantum-mechanical computation of one molecule of methane requires 10^{42} grid points. Assuming that at each point we have to perform only

[60] W. D. Hillis, "New Computer Architectures and Their Relationship to Physics or Why Computer Science Is No Good" (1982).
[61] Rolf Landauer, "Physics and Computation" (1982).
[62] Minsky, "Cellular Vacuum" (1982).
[63] Greenspan, "Deterministic Computer Physics" (1982).
[64] Wheeler, "The Computer and The Universe" (1982).
[65] Benioff, "The Computer As a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers As Represented by Turing Machines" (1980); Benioff, "Quantum Mechanical Models of Turing Machines That Dissipate No Energy" (1982b).
[66] Poplavskii, "Thermodynamical Models of Information Processing" (1975).

10 elementary operations, and that the computation is performed at the extremely low temperature $T = 3 \times 10^{-3}$ K, we would still have to use all the energy produced on Earth during the last century."[67] In 1980, Yuri Manin wrote *Vychislimoe i nevychislimoe* (*Computable and Uncomputable*), which further explored such ideas. The language barrier, combined with the very real travel barrier imposed by the Soviet Union, prevented these works from being influential in the West.

Today we can read excerpts of Manin's 1980 article in English, thanks to his leaving Russia and publishing an English-language 2007 edition of his essays. "We need a mathematical theory of quantum automata," Manin wrote. "Such a theory would provide us with mathematical models of deterministic processes with quite unusual properties. One reason for this is that the quantum state space has far greater capacity then the classical one: for a classical system with N states, its quantum version allowing superposition (entanglement) accommodates e^N states."[68]

Some journalists and historians of science cite these articles by Poplavskii and Manin as evidence for the idea that quantum computing arose on both sides of the Iron Curtain. However, these articles do not appear to have spawned conferences or investment in Russia, as their counterparts did in the United States (see discussion of nation-state investment in quantum information science in Section 9.2, "Industrial Policy" (p. 380)).

We believe that these publications are similar to Feynman's 1959 talk, in which Feynman posits that at the atomic scale computation can be performed not with circuits, "but some system involving the quantized energy levels, or the interactions of quantized spins."[69] Such a statement is a long way from Feynman's detailed proposals for quantum computing that would come two decades later, and there is no intellectual approach for drawing a line from Feynman's 1959 talk to modern-day quantum computing (or to modern-day nanotechnology, for that matter), because that line points back to Fredkin and Toffoli, and then to Burks and von Neumann. While Poplavskii and Manin were certainly walking down intellectually intriguing paths,

[67] As quoted in Manin, "Classical Computing, Quantum Computing, and Shor's Factoring Algorithm" (1999).

[68] Manin, *Mathematics As Metaphor: Selected Essays of Yuri I. Manin* (2007).

[69] Feynman, "There's Plenty of Room at The Bottom: An Invitation to Enter a New Field of Physics" (1959).

the historical record implies that their paths were never explored beyond the first few steps.

4.8 Aftermath: The Quantum Computing Baby

Feynman returned to California, where he delivered several more lectures on the promise of quantum computing. He published an article about the idea in a special publication marking the 40th anniversary of the Los Alamos laboratory;[70] a revised version appeared in Optics News.[71] Another version of the article appeared in Foundation of Physics the following year.[72,73]

4.8.1 Growing Academic Interest

Three years after the MIT conference, the British physicist David Deutsch wrote an article discussing the relationship between computing, physics, and the possibility of quantum computing for the *Proceedings of the Royal Society of London*, one of the world's oldest and most prestigious scientific journals. "Computing machines resembling the universal quantum computer could, in principle, be built and would have many remarkably properties not reproducible by any Turing machine,"[74] Deutsch hypothesized. The statement is literally true, because quantum computers as he proposed them would have access to both a source of perfect randomness and the ability to create entangled states. Such a machine *would* be able to model quantum physics and quantum chemistry to any arbitrary precision (discussed in Chapter 5), and create unbreakable cryptographic codes (discussed in Chapter 7). This article helped to legitimize the idea of quantum computing and present it to a broader scientific and technical community that had not previously encountered it. "To view the Church–Turing hypothesis as a physical principle does not merely

[70] Feynman, "Tiny Computers Obeying Quantum Mechanical Laws" (1985b).
[71] Feynman, "Quantum Mechanical Computers" (1985a).
[72] Feynman (1986).
[73] Feynman's son, Carl Feynman, was an MIT classmate of Danny Hillis. Feynman learned of Thinking Machines when the company was being formed and offered to spend the summer helping out. He was hired as a consultant shortly after the company was founded, becoming its first employee. Feynman soon found that the Connection Machine's mesh architecture was surprisingly well-suited to performing the complex computations required for simulating quantum mechanics and other kinds of physical systems, paving the way for the company's early sales. See W. D. Hillis, "Richard Feynman and The Connection Machine" (1989).
[74] Deutsch, "Quantum Theory, The Church–Turing Principle and The Universal Quantum Computer" (1985).

make computer science into a branch of physics. It also makes part of experimental physics into a branch of computer science."

Reading Deutsch's article 35 years after its publication, a confusing aspect is the fact that he differentiates a "quantum computer" from something he calls a "Turing-type machine." The article conveys that a Turing-type machine is limited in that it can only execute steps sequentially, while Deutsch suggests that a quantum computer will be able to solve some problems faster because it will be able to consider many states at once, in part because it is based on quantum computing, and "quantum theory is a theory of parallel interfering universes." What is confusing about this today is that the Church–Turing hypothesis is not concerned with the *speed* with which a computation can be performed – it is only concerned with whether a computation can be performed *at all*.[75] In 1984 it was not immediately clear whether quantum computers would face the same limitations of Turing machines, or if they might implement a stronger, more powerful form of computation. Today computer scientists have shown that quantum computers may be more efficient at solving certain kinds of problems, but they cannot solve problems that are fundamentally different than Turing machines – or if they can, we haven't figured out how to express such power.[76] Surprisingly, even this perceived efficiency of quantum computers is a belief – it has not been mathematically proven, for reasons described in the following chapter.

In 1985 Asher Peres at Technion, the Israel Institute of Technology, published an article further exploring how a quantum computer might do something extremely simple: adding together 1-bit numbers. In working through his example, Peres showed that a quantum mechanical computer would necessarily require some kind of error

[75] For example, a sequential Turing machine with a clock speed of a billion cycles per second is likely faster at computing problems than a parallel Turing machine with a thousand processors all running with a clock speed of a 10 cycles per second, but both machines are universal. By *universal*, we mean that either of these machines, given enough memory and enough time, could compute what any other Turing machine can compute.

[76] Quantum cryptography is fundamentally different from quantum computing, in that today we know mathematically that systems that use quantum cryptography can do something that it is simply impossible to do with conventional cryptography, and that is exchange messages in a way that they cannot be intercepted without detection by an attacker. However, Quantum cryptography is not strictly solving a problem, and it doesn't use quantum computing, so it doesn't disprove the sentence referenced in the paragraph above.

correction. Ideally, Peres wrote, with such a system "it should be *impossible to keep a record* of the error,"[77] because errors would ideally cancel out each other. He ended the article by noting that quantum computers need not be digital computers: "Ultimately, a quantum computer making full use of a continuous logic may turn out to be more akin to an old-fashioned analog computer, rather than to a modern digital computer. This would be an ironic twist of fate." (The D-Wave quantum computer resembles an analog computer; we discuss traditional analog computers in Chapter 3.)

In October 1992, the Dallas IEEE Computer Society and Texas Instruments sponsored the Workshop on Physics and Computation. "This workshop was long overdue since the first major conference on the Physics of Computation was held at MIT over a decade ago," wrote Doug Matzke, the workshop's chair. Landauer was the keynote sponsor; Fredkin "gave a stimulating and entertaining talk" at the banquet.[78] A follow-up conference was scheduled for two years later, in 1994.

In June 1994, Peter Shor, then a researcher at AT&T Bell Labs, published a technical report at the Center for Discrete Mathematics & Theoretical Computer Science (DIMACS), at the time a joint research project between Bell Labs and Rutgers University. An "extended abstract" based on the technical report was presented at the Foundations of Computer Science (FOCS) 1994 conference, which took place between November 20–22 in Santa Fe, New Mexico. Shor's paper showed that if a certain kind of quantum circuit could be built on an as-yet non-existent quantum computer, then laws of quantum mechanics could be combined with number theory in such a way as to solve a particular math problem very efficiently. Solving *that* particular math problem would make it possible to efficiently factor large numbers.[79] And factoring large numbers would have a huge impact on the world, because the world's most sophisticated encryption systems at the time (and still today) depended upon the fact

[77] Peres, "Reversible Logic and Quantum Computers" (1985).

[78] Matzke, "Message From The Chairman" (1993).

[79] Shor uploaded "an expanded version" of his FOCS paper to arXiv on August 30, 1995, and updated that version in January 1996. The papers can be found at arxiv.org/abs/quant-ph/9508027. This version of the paper was published as Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer" (1997).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

that we are unable as a species, on Earth, today, to rapidly factor large numbers.

It is hard to overstate the significance of Shor's algorithm for the development of quantum computing. Before Shor's announcement and subsequent publication, quantum computers were nonexistent theoretical constructions that were largely a curiosity of the physics and theoretical computer science communities. Shor's algorithm showed that there would be serious, real-world implications for quantum computers that would directly impact national security. It was the starting gun of the quantum computing race. Charles Clark at the US National Institute of Standards and Technology organized the NIST Workshop on Quantum Computing and Communication, held in August 1994 at the agency's campus in Gaithersburg, Maryland.[80] Based on a discussion at the workshop, NIST had a working quantum circuit with two qubits based on trapped ions operational in July 1995.[81] (David J. Wineland, one of the paper's authors, would later share the 2012 Nobel Prize with Serge Haroche "for groundbreaking experimental methods that enable measuring and manipulation of individual quantum systems.")

Also in the summer of 1995, the MITRE Corporation's "JASON" summer study, funded by DARPA, focused on quantum computing. The report identified factoring and simulating quantum physics, but presented diagrams for how to create a quantum adder and multiplier, and discussed the importance of quantum error correction. The report had three main recommendations.

- "Establish a research program to investigate possibilities for quantum computing beyond Shor's algorithms."
- "Seed research in various communities for quantitative minimization of algorithmic complexity and optimum circuit."
- "Supplement ongoing experimental research related to the isolation and control of discrete quantum systems suitable for quantum logic."

[80] Gaithersburg, MD: National Institute of Standards and Technology, "NIST JumpStarts Quantum Information" (2018).

[81] C. Monroe et al., "Demonstration of a Fundamental Quantum Logic Gate" (1995).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

Also in 1995, Benjamin Schumacher coined the word *qubit* in his article "Quantum Coding."[82] In the article, Schumacher compares the information-theoretic differences between traditional bits of information and "Shannon entropy" and quantum bits, which had previously been called two-state quantum systems, and which Schumacher termed *qubit*. But whereas Shannon's seminal 1948 article[83] contemplated the information capacity of a noisy channel, Schumacher considered the information capacity of a noiseless quantum communications channel. He then considers the impact of entanglement between quantum states. In the article's acknowledgments, Schumacher notes: "The term 'qubit' was coined in jest during one of the author's many intriguing and valuable conversations with W. K. Wootters, and became the initial impetus for this work. The author is also grateful to C. H. Bennett and R. Jozsa for their helpful suggestions and numerous words of encouragement."

4.8.2 The First Quantum Computers

Three years after NIST created the first quantum circuit, two separate teams of researchers proposed, developed and published similar approaches for using nuclear magnetic resonance (NMR) in liquids as the medium for quantum computation.[84] "Although NMR computers will be limited by current technology to exhaustive searches over only 15 to 20 bits, searches over as much as 50 bits are in principle possible, and more advanced algorithms could greatly extend the range of applicability of such machines," observed Cory et al.

The challenge with NMR-based quantum computers is that the NMR spectrum increases in both complexity and density with each additional qubit. At some point the spectrum becomes too complex, and too noisy, to make sense of the computation's result. But these computing systems demonstrated that the theoretical ideas first proposed by Feynman and later refined by Shor actually worked: in 1998 the first algorithm was run on an NMR-based quantum computing system (see Section 5.3 (p. 210)), and in 2001 Shor's algorithm was run for the first time on an actual quantum computer, an NMR system with 7 qubits, successfully factoring the number 15 to get

[82] Schumacher, "Quantum Coding" (1995).

[83] Shannon, "A Mathematical Theory of Communication" (1948).

[84] Gershenfeld and Chuang, "Bulk Spin-Resonance Quantum Computation" (1997); Cory, Fahmy, and Havel, "Ensemble Quantum Computing by NMR Spectroscopy" (1997).

its prime factors, 3 and 5.[85] We will further discuss Shor's breakthrough and the race for quantum factoring in the next chapter (see Section 5.2, p. 188).

4.8.3 Coda

In the past 25 years, the world has seen quantum computers go from theoretical constructs to working machines that can solve real problems. But progress on quantum computers has been much slower than progress during the first 25 years of classical electronic computers.

The London Mathematical Society published Alan Turing's model for computation in 1936. By March 1940 Turing had built the first code-breaking Bombe at Bletchley Park. Together with the Colossus machines, Bletchley Park was able to decrypt thousands of messages a day, and had a significant impact on the war effort. In fact, the impact was so significant that the existence of these machines was kept secret for decades. Meanwhile, by the end of World War II there were stored program computers in various states of design, operation, and construction in Germany (where the effort was largely ignored by the Nazi military), the United Kingdom, and the United States. Early electronic computers used a variety of different technologies for computing and storage, including relays, tubes, and spinning magnetic drums, but the industry was profitable from the very start. By 1965 the industry had firmly settled upon transistorized logic. IBM manufactured the first hard drive in 1956, and in 1970 Intel publicly released the first commercial DRAM (dynamic random access memory) chip. Governments and corporations bought these computers to solve problems that required organizing information and performing computations.

Quantum computing, in contrast, was first proposed in the 1970s. It wasn't until 1994 that there was a clearly articulated reason for creating such a machine: not to simulate physics, but to crack codes. Unlike the first electromechanical and electronic computers, the first quantum computers could not crack any messages of any significance whatsoever: the most impressive mathematical feat that one of the machines accomplished was to factor the number 15 into the prime numbers 3 and 5. Unlike the work at Bletchley Park, the work on quantum computing has taken place in public, with multinational

[85] Vandersypen et al., "Experimental Realization of Shor's Quantum Factoring Algorithm Using Nuclear Magnetic Resonance" (2001).

Quantum Computers: Not Just Fancy Analog Devices

At the dawn of the computer age there was considerable interest in so-called *analog computers* for solving a variety of scientific problems. Some of these machines were mechanical, with rods, gears and curves milled into metal,[a] while others were electronic. Indeed, much of the recent success in artificial intelligence is based on a computing model that is essentially analog (and was first created with analog computers), and analog computers are making a comeback in some areas.[b]

But quantum computers are not simply a new take on analog computers:

- The physical things that represent information inside an analog computer are one-dimensional vectors, such as position (in mechanical analog computers) or voltage (in electronic analog computers). Quantum computers use two-dimensional vectors (the complex numbers used to compute quantum wave functions).

- Analog computers don't rely on superposition or entanglement, with the result that all of the information stored within an analog computer is not potentially interacting with all of the other information stored inside an analog computer. Put another way, the individual parts of a large analog computer appear to experience local causality and statistical independence; the lack of these makes quantum computing possible.

- As such, information can be copied out of an analog computer without destroying the information it contains. It is thus possible to covertly eavesdrop on an analog computer. Quantum computers and networks, in contrast, can detect eavesdropping because it destroys their computations.

- Analog computers can't efficiently run quantum algorithms such as Shor's algorithm or Grover's algorithm.

[a] Clymer, "The Mechanical Analog Computers of Hannibal Ford and William Newell" (1993).
[b] Tsividis, "Not Your Father's Analog Computer" (2017).

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

teams engaging in a friendly competition within the pages of scientific journals. Today, 23 years after the first successful quantum computation, there is still no agreement on what media should be used for quantum computation, and whether it is better to run machines in vats of liquid helium cooled close to absolute zero, or if they can be run at room temperature. Whereas technologies for storing digital information preceded Turing's paper by more than a century,[86] approaches for storing quantum information are still on the drawing board.

Unquestionably, computing with superposition and wave equations that we describe in Appendix B – what we call quantum computing – is much harder than computing with relays, tubes, and transistors – classical computing – that we describe in Chapter 3. Having recounted the history of quantum computing from 1961 through 1998, we explain in the next chapter why governments and corporations continue to pursue quantum computing. We discuss the kinds of devices being made, their intended uses, the competitive landscape, and the outlook for the technology.

[86] Joseph Marie Jacquard (1752–1834) patented his punch-card operated loom in 1804.

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press

https://doi.org/10.1017/9781108883719.008 Published online by Cambridge University Press