AFTER DIGITAL

Computation as Done by Brains and Machines

JAMES A. ANDERSON

OXFORD

AFTER DIGITAL

Computation as Done by Brains and Machines

James A. Anderson

OXFORD
UNIVERSITY PRESS

CONTENTS

PREFACE

Gray, my friend, is all theory, but green life's golden tree.

—Goethe (1749–1832), Faust, Part I

This book derives from a course I taught at Brown for several years called "Computation as Done by Brains and Machines." We all own, tolerate, and sometimes even love our quirky and excessively literal digital computers and use them for chores at home and work. But computers based on digital electronics—our familiar computer companions—are only one form of computing hardware. Digital computers arise from a particular line of intellectual descent, from abstract binary logic in the 19th century, leading to powerful and elegant mathematical results in the 20th to the widespread construction of simple, cheap, fast, versatile, and powerful logic-based hardware in the present.

But if we view a "computer" as an aide to cognition rather than a specific class of hardware, there are other possibilities. We explore some of them in this book.

The major theme of this book is that the design of the basic computing hardware makes a huge difference in what computers can do and how effectively they can do it.

Digital computers are built from simple interconnected elements. The elements can be in one of two states. These states are variously interpreted as high-voltage or low-voltage, on or off, or even TRUE or FALSE, as in logic. Digital computer hardware works by performing "logic" functions very rapidly in sequences that can be as long as millions or even billions of simple operations. Usefulness comes not from the simple hardware but from the vastly complex set of instructions required to get the simple hardware to do anything useful.

There are other ways to make a "computer" that use very different hardware, and different hardware leads to a different spectrum of practical applications. Not so long ago, a major competitor for digital hardware was the "analog" computer. Analog computers work directly with quantities like voltages in an electrical circuit or the positions of gears in a mechanical device. Inputs and outputs are often continuous quantities. They are not two-state devices; hence, there is not a single type of analog computer, but many different types depending on what needs to be computed. The hardware is designed to match a specific application.

Forty years ago, analog computers were considered to be viable competitors to digital computers. At that time, they were more convenient to use and much faster for solving many

important problems than that era's digital machinery. However, as digital computers became cheaper and faster, analog computers lost the battle for survival and now even their name is unfamiliar, lost in the graveyard of past technologies.

Another alternative way to build computer hardware, one less studied, is found in the nervous system of animals, most importantly, that of humans. We still do not understand how the brain works in detail, but it is similar to analog computers in that many of the components—nerve cells, groups of nerve cells—work directly with analog quantities.

A problem with the "brain-computer" is that the basic elements that comprise it are slow and somewhat unreliable. Neural elements are at least a million times slower than the logic gates used in digital computers. But even though neurons are slow, there are billions of them. Somehow, the proper use of billions of elementary computing elements has produced a system more capable of performing some specific important cognitive tasks—perception, reasoning, intuition—than a digital computer, even though the digital computer is constructed from far faster and more reliable basic elements. And the brain does it with the energy consumption of a small lightbulb. It would be of both intellectual and practical importance to know how this feat is done.

Much of the interest in the brain as a computer—and our interest in this book—surrounds ability to perform the complex cognition that is the core of our being as humans. The structure that does these functions in humans is primarily the cerebral cortex. The high performance of our cerebral cortex is the major specialization of the human species, just as long fangs were a specialization of a saber toothed tiger or bad smells are for a skunk.

The basic conclusion? Hardware matters. Because analog, digital, and brain-like computers use such different hardware, it is not surprising that, in practice, they do best very different things. Agreeably, from our human point of view, the strengths of one form of computing are often complementary to the weaknesses of other forms. A goal of this book is to introduce different kinds of computation, appreciate their strengths and weaknesses, and see how they might ultimately work together.

Optimistically, one can foresee a future in which biological computers, analog computers, and digital computers work together in happy symbiosis, perhaps as tightly coupled to each other as a eukaryotic cell is to its resident mitochondria or a termite is to its gut bacteria that metabolize cellulose.

AFTER DIGITAL

OXFORD
UNIVERSITY PRESS

Oxford University Press is a department of the University of Oxford. It furthers the University's objective of excellence in research, scholarship, and education by publishing worldwide. Oxford is a registered trade mark of Oxford University Press in the UK and certain other countries.

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America.

Library of Congress Cataloging-in-Publication Data
Names: Anderson, James A., author.
Title: After digital : computation as done by brains and machines / James A. Anderson.
Description: New York : Oxford University Press, 2017. | Includes bibliographical references and index.
Identifiers: LCCN 2016042404| ISBN 9780199357789 (hardcover : alk. paper) | ISBN 9780199357802 (epub)
Subjects: LCSH: Computers—Psychological aspects. | Computational intelligence. | Computational neuroscience. | Neuropsychology.
Classification: LCC QA76.9.P75 .A53 2017 | DDC 006.3—dc23
LC record available at https://lccn.loc.gov/2016042404

1 3 5 7 9 8 6 4 2

Printed by Sheridan Books, Inc., United States of America

CHAPTER 1

THE PAST OF THE FUTURE OF COMPUTATION

What I cannot create I do not understand.

— Richard Feynman (On his blackboard at the time of his death, 1988)

Everyone knows what a computer is, and almost everyone owns one or two or three or more. Everyone agrees that computers have gotten faster and more powerful. Computers "compute," but "computation" is not so easy to define since "computation" comes in multiple flavors.

This book discusses three flavors of computation: analog, digital, and something that lies between the two and has aspects of both—computation as done by an important biological structure, our brain.

We will see that even though these flavors all "compute," to become practical devices they must work in very different ways even when they carry out the same operation. Telephones made with analog circuitry can be as simple as two tin cans and a string, or a microphone, earpiece, and the electric wires connecting them. In contrast, even the simplest digital telephone is a device of baroque complexity.

But what is "computation," now that we now have more of it than we had previously? What is it that we have more of? Definitions of computing can get complex and abstract. But perhaps we can instead define computers by what they do for their users, rather than how they do it or from what hardware they are constructed.

We could define a computer and what it does as a cognitive tool. Tool construction and use arises in a different, older path of human cognitive evolution than language and the associated abstractions of language, like logic, that are so important for digital computing. This definition of computation—as a cognitive tool—is messy and imprecise but shifts emphasis toward function and use and not to the intimate details of a particular class of hardware.

Such a functional definition also allows connection with the long history of complex tool construction in our species and its precursors. Tools are found in species much older than *Homo sapiens* (e.g., *Homo erectus*), and tools appeared as early as 2 million years ago in the form of stone hand axes, carefully shaped to enhance their function. *H. erectus* had a somewhat smaller

brain size than modern humans, perhaps 700–900 cc as opposed to 1,100–1,500 cc. It is not known if they had precursors to language, but it is very unlikely that they had language with anywhere near the abstract complexity of modern human language.

The hand axes that *H. erectus* made were remarkably well built. It takes a good deal of skill and a cultural tradition to make tools of such quality. PhDs graduating from the University of California, Berkeley, Anthropology Department were required at one time to learn to chip stones to construct a workable tool. It is not easy.

Even more complex tool construction and use has been found several hundred thousand years ago, still before modern *Homo sapiens* appeared. Several well-designed and carefully built javelin-like spears 6 to 7 feet long have been found in a soft coal mine in Germany, carbon dated to around 300,000–400,000 years ago, again before the emergence of modern humans. It is not easy to make an accurate spear that will remain stable for thrusting or in flight and hit the target with enough force to do damage. It requires cultural experience and manufacturing skill to give a piece of wood the proper tapered, hardened, sharpened shape and then use it effectively. This spear seems to be a device to allow organized hunting of big game, and it was not constructed by the hands of modern language-using man.[1]

We don't know exactly when fully human intelligence with complex language appeared. Our species, with a large brain and other necessary specializations for language in the form of vocal tract configuration, precise control of breath, and enhanced audition, is perhaps 200,000 years old, probably less.

The archaeological record suggests that unmistakable human-like cognition—indicated by use of symbols, abstractions, and art—is first found less than 100,000 years ago in South Africa, 50,000 years ago in Australia, and seemingly in fully developed form 35,000 years ago in cave paintings in Europe.

Return to the proposed definition of a computer as a tool, not an abstract engine. Tools like hand axes, javelins, and projectile points were designed to extend human ability to deal with the physical world beyond unaided human physical capabilities. Human cognitive abilities also need assistance. Therefore, we suggest that the computer is best understood as a tool arising from a long tool-making tradition, a device, designed to extend our cognitive powers beyond what our original biological equipment can do. How it is constructed is critical for the maker of the cognitive tool, not so much for the user. One other aspect of this definition is that it allows for the essential cultural "software tools" used to extend our raw cognitive abilities. Obvious examples for *H. sapiens* would be language and arithmetic.

COMPUTER HARDWARE EVOLUTION

If we think of the evolution of computers of whatever kind as the development of a cognitive tool, the change in the most common hardware for the familiar computing tool we call a digital computer has been remarkable over the past century.

In 1945, the first American digital computer, the ENIAC, used 17,000 vacuum tubes, 1,500 relays, and took up 1,800 square feet of floor space, the size of an average house. It consumed 150,000 watts of power and weighed 30 tons. It was programmed using switches and plug boards.

But computer hardware has changed. Consider the computers contained in the ubiquitous cell phone. The ENIAC weighed 30 tons; a cell phone weighs less than a pound. ENIAC used 150,000 watts of power; a cell phone uses perhaps a watt. Vacuum tubes were unreliable, so even with exceedingly good engineering ENIAC was out of commission roughly half the time due to equipment failure; cell phone computers almost never break. It is hard to compare exactly, but

the computer chip in the cell phone is many thousands of times more powerful by almost any measure: speed, memory, reliability. That's progress. But what is it that got smaller, better, and faster? Was a price in complexity paid for this increased capacity?

TELEPHONES AS AN EXAMPLE OF TWO "COMPUTING" TECHNOLOGIES

As a familiar example of two different technological approaches to the same human task, consider telephones. The essential human function of a telephone is to let members of our very social species communicate with each other using speech beyond the range of an unassisted voice. The telephone network in 1945, the time of the ENIAC, was well developed, widely available, and allowed worldwide voice communication. At that time, most telephones used wires to connect users to each other although there were a few radio links.

A modern cell phone lets teenagers talk and text to their friends from almost anywhere and—please note—also plays video games, applications inconceivable in 1945. In the 21st century, instead of wires, cell phones use low-powered radio transmitters and sensitive receivers to connect to a very complex switching network.

Classic wired telephones and cell phones both convey speech over a distance, but do so very differently. Spoken speech gives rise to pressure waves in the air. The vocal tract of the speaker constructs the speech signal by using air from the lungs streaming past the vocal cords and modified by the shape of the mouth and throat. Pressure waves in the air move from the mouth of the speaker to the ear and brain of the listener where they are converted into neural signals and, ultimately, understood as language.

FROM SPEAKER TO LISTENER

In wired telephones, voltages are produced from the speech pressure waves hitting a microphone. In 1945, these voltages were then sent from the transmitter (microphone) to the receiver (earpiece) over copper wires. Instead of copper wires, it is also possible to relay speech with two tin cans and a string. Words spoken into one tin can are reproduced in the other, transmitted by the vibrations in the string. In both cases—wires and string—the pressure waves from the speaker directly produce corresponding electrical or mechanical vibrations. The size of the vibrations can be large or small, corresponding crudely to loud or soft. Such a direct conversion of one continuous quantity—pressure waves in air—into another—voltage on a wire or vibrations on a string—is an "analog" device that works by directly transforming one continuous quantity into another continuous quantity.

Cell phones perform exactly the same transmission of a pressure wave from a speaker to a listener. However, the way they do it is totally different from a wired phone and is far more complex. A radio transmitter in the speaker's cell phone sends radio signals to a local antenna and receiver. These antennas are often visible on towers or high buildings. The received signal connects to the regular telephone system, which nowadays usually is based on fiber optics rather than copper wire. The signal is processed and sent over the telephone network to a transmitting antenna near to the cell phone of the listener, where it is transmitted and then received by the listener's handset.

The signal transmission and reception outside of the first and last stages—the microphone and the handset—is entirely "digital." "Digital" means information that, even about continuous quantities such as sound, is composed entirely of signals whose components can be interpreted as either "one" or "zero." A continuous pressure wave in air is converted into a set of some number of binary values, each composed of ones and zeros, that represents the actual speech signal.

A continuous sound pressure wave can take on many values of loudness, essentially every possible intermediate value between a highest or lowest limit. Transmitting these values continuously as they change with time is no problem for a voltage on a copper wire because the voltage can take any intermediate value. However, a major problem arises in a device using groups of distinct binary words to represent loudness. For example, a binary word composed of eight ones and zeros—eight bits—can only represent 256 different values of loudness. We could assign the value zero to no sound and 255 to the loudest sound, but a voltage corresponding to an intermediate value can be represented by one of only 254 possible values. When faced with a loudness corresponding to, say, 135.4, the word can only respond with the binary word meaning "135." Fine detail has been lost. The signal has been corrupted. Worse, a range of different values of loudness—from 134.6 to 135.5—will be described by the same binary word for 135. Humans could probably hear the difference in loudness between 134.6 and 135.5. This phenomenon is often called *quantization noise* and haunts digital systems. It can degrade the fidelity of the signal substantially.

In addition, the value of loudness cannot be transmitted for every instant of time—there are too many "instants." The resulting signal has a limited number of values of loudness at a series of particular times.

Samples are taken at regular intervals called the *sampling rate*. The basic telephone samples the pressure wave amplitude 8,000 times a second; that is, it converts a second's worth of speech into 8,000 binary words each containing eight bits to represent different levels of loudness. The resulting sound quality is poor but understandable.

The reason such a poor representation of speech is used is because the telephone system was originally designed in the 19th century. At that time, microphones were based on the compression of carbon granules, and earpieces were poor. However, even now, the current North American Public Switched Telephone Network (PSTN) standard for frequency response is 300–3,400 Hz, roughly from middle C on a piano keyboard to the piano key A7, four keys from the right end of an 88-key piano keyboard. The low-frequency loss is even more serious. Middle C on the piano is about 260 Hz and is the fortieth key on the piano, so most of the keys on a piano keyboard are weakened to a significant degree. Worst of all, the average fundamental frequency of a male voice is 150 Hz, well below the PSTN standard frequency band.

An obvious question is why do standard telephones sound as good as they do? A major reason is a bit of high-quality neural processing in the human auditory system, not a technical fix by the phone company. The phenomenon is called the *missing fundamental* or the *phantom fundamental*. Even if the fundamental frequency of a male voice is largely removed by the telephone circuitry, the brain of the listener can reconstruct what it would have been by looking at the overtones of the missing fundamental. The reconstructed pitch is what is heard by the listener. This is a good example of a common brain-like computational strategy. If something is missing that past experience strongly suggests should be there, then put it there. Is this useful hallucination a bug or a feature? We will see other examples of this strategy in action in the visual system and in cognition.

In a remarkable bit of technological conservatism, this limited fidelity telephone speech signal has changed little since the time of Alexander Graham Bell, although it would now

be possible to do much better. For example, music on single channel of a compact disk is sampled 44,100 times a second and has more than 64,000 amplitude levels possible for each sample; that is, the loudness of the signal is represented by a 16-bit binary word. Compact disk audio sounds far more natural than telephone speech. It would be easily possible to generate much higher quality speech for basic telephone service, but there is no incentive to do so.

This multiple stage process of (1) a voltage → (2) converted to a digital word → (3) transmitted → (4) converted back into a voltage is far more complex than a voltage simply sent down a wire. It is hard to say exactly how much more complex, but from essentially no distinct operations in a wired telephone, even a limited-fidelity digital system requires several million distinct operations per second. This estimate does not even mention the complexity of the system required to switch the signal between transmitters and receivers at different locations on the telephone network (see Figure 1.1).

All engineering students run across the abbreviation "KISS," which stands for the directive, "Keep it Simple, Stupid." These are words to live by in practical technology. But clearly a cell phone violates it. If we are keeping score, then, if simple is good the winner is:

Simplicity: *Advantage Analog*

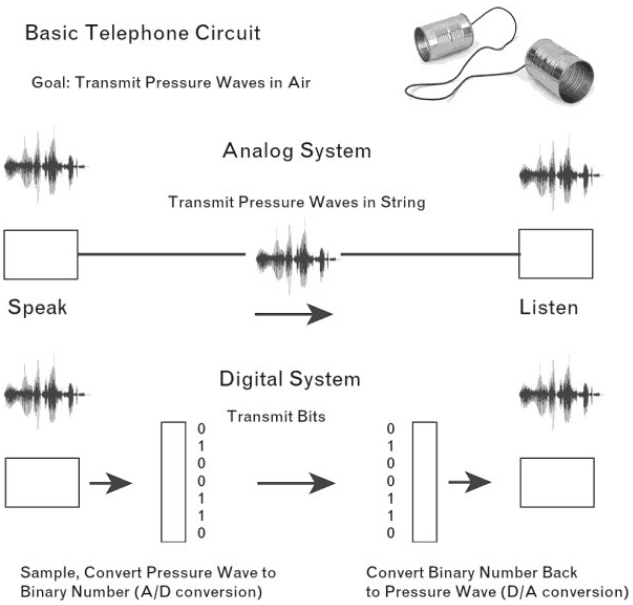

FIGURE 1.1: Cartoon showing differences in processing between digital and analog telephone systems. The analog system transmits a pressure wave down a string or a voltage down a wire. The digital system converts the pressure wave into a binary word (A/D; *analog-to-digital conversion*)—in this case a word of 8 bits—8,000 times a second, and transmits the resulting word to a receiver over some (unspecified) network and then converts the word back into a pressure wave (D/A; *digital-to-analog conversion*). There is no doubt which method is simpler, but there is also no doubt as to which process ultimately won in terms of overall commercial success. Figure by James A. Anderson

FLEXIBILITY

Digital speech transmission seems like an excessively complex way of working with limited-fidelity speech. It dramatically violates the KISS principle. But something remarkable is provided in return for the dramatic increase in complexity: flexibility.

Essentially all a classic telephone can do is send information about a pressure wave from one end of a wire to the other. Modern cell phones let teenagers talk with their friends from almost anywhere, performing very well the basic communication function of the telephone. But digital processing is so intrinsically flexible that the same processors that convey speech can also transmit text messages and photographs, display maps, watch movies and TV shows, rate restaurants, and serve a whole universe of vastly different applications. Because the essential speech aspect of the telephone is handled by a very powerful digital computer, it becomes possible to run many different applications on a cell phone by repurposing this computer.

It is this amazing flexibility that gives digital computers their power and usefulness. What a digital system loses in simplicity is more than made up for by flexibility.

Flexibility: *Advantage Digital.*

UP, UP, AND AWAY: INCREASING COMPUTER POWER WITH MOORE'S LAW

Every recent discussion of the future of computing starts with a graph showing the past growth of computer hardware is projected into the future. Over the years, digital computers have gotten faster and more capable in every way, no matter how capability is defined. But as their computing capabilities have become bigger and bigger physically, their physical electronic components have become smaller and smaller.

Description of this impressive, constantly increasing performance is based directly or indirectly on "Moore's Law." Moore's Law, is named after Gordon Moore, a co-founder of the semiconductor giant Intel. Remarkably, Moore's Law has been faithfully followed for more than 50 years, since Moore proposed it in 1965. Moore's Law is not a description of computing power, but lithography. It predicts that the number of electronic devices that can be crammed onto a little chip of silicon will double roughly every 1–2 years. Small size and the number of devices map directly into computer capability because more and smaller devices are, through the physics involved, faster and more powerful devices.

The semiconductor chip industry is now manufacturing chips with individual electronic component sizes far under a wavelength of light. An adenovirus (90 nanometers), responsible for many human respiratory infections, is roughly the size of the 2003 VLSI chip geometry. Current VLSI geometries are now well below 30 nanometers, under the size of many viruses. Recent improvements in technology predict that transistors as small as 7 nanometers may enter production as soon as 2017. The width of a strand of DNA is 2.5 nanometers.[2]

However, it is now widely believed that the increased computer power described by the Moore's Law increases in device density will be coming to an end "soon." Devices cannot become much smaller, and the reasons for this are due to fundamental laws of physics: smaller devices beyond a certain point—which we are now approaching—become unreliable due to quantum mechanics, the science of the extremely small.

One reason for this is a quantum mechanical effect called *tunneling*. Ordinary experience tells us that that an object cannot pass over a wall if it does not have the energy to reach the top of the wall. But for very small objects there is a finite probability they will "tunnel" through the barrier even if they do not have the energy to get over it. The classic example used in classes in elementary quantum mechanics is computing the probability that a truck will pass through a brick wall undamaged. As expected, the odds of this happening are very, very low and the probability depends on the mass of the truck and the strength and height of the wall. But it is not zero.

However, the insulators in the transistors in a chip keeping voltages apart from each other may be only a few tens of atoms thick. The chances of tunneling right through the insulating barrier are now not low. Tunneling can cause significant heating. Even worse, a logic element might spontaneously change state, say from ON to OFF, or 1 to 0. It would be hard to sell a computer that gave different answers, randomly, to the exact same problem. Such events would be a disaster for computer reliability. There are many ways of detecting whether or not a hardware error has occurred and then compensating for it. One well-known way is from Lewis Carroll in "The Hunting of the Snark," "*I have said it thrice: What I tell you three times is true*." Unfortunately, running a program three times and choosing the majority answer may negate any speed advantage from small device size.

PARALLELISM

Current technology has several generations of increased density yet to go. But whether Moore's Law fails next year or next decade does not matter: the end is in sight. And the way to maintain the familiar "natural" increase in power may not be easy.

One short-term response of the computer industry to this situation is to use multiple computers running simultaneously to provide more computer power to apply to a problem. This response can now be seen even in familiar home computers. By looking at the details of home computer ads, it can be seen that the speed of the computers themselves has not increased for several years. What has changed is the appearance of processors with multiple "cores." A "core" is an additional processing unit; that is, a separate computer.

It seems like a plausible argument that if one computer is fast, then two computers working together should be twice as fast and ten computers ten times as fast. This arrangement is called *parallelism* because many computers can be working on the same problem at the same time, in parallel. Unfortunately, this appealing idea does not work as well in practice as one might expect. It would work best if the task at hand was like digging a giant hole in the ground, where, in fact, many hands could indeed make light work. But computer applications of any significance are more like baking a cake, where a number of discrete steps have to be done in a precise sequence—serially—with the next step starting when the earlier step ends (i.e., mixing the ingredients before putting the cake in the oven). There are a few but very important applications where parallelism is possible, fast, and useful in science and engineering and in the advanced computer graphics seen in movies and video games.

But for general problems with many computers working in parallel, performance has not lived up to initially hopeful, naïve expectations. A user of the Dell website will note that the number of cores seems to have stabilized at four to eight. One reason for this is that it is very hard to write general-purpose software for parallel computers.

One feature of the biological brain that has interested many in the computer industry for years is how powerful the biological brain is as a computing device in spite of the fact that it is

built from a great many very slow, somewhat unreliable nerve cells. But with even such inferior hardware, when arranged in parallel with all nerve cells working together, brains can show great power and flexibility. Brains somehow do parallelism "right." It would be useful to know how they do it so computers could be built that way. A theme that will recur in later chapters is where the computer industry might find ideas to let it continue into the future the doublings of power of the past and what this new power might be used for.

LET'S BUILD AN ARTIFICIAL BRAIN!

Moore's Law deals with the density of devices for digital computer hardware (see Figure 1.2). In practice, smaller devices mean faster devices. Faster devices mean more operations can be performed in a second, thus giving more "power." In popular computer literature, a graph of Moore's Law is often followed by a graph of computer "power." Sometimes these graphs of "power" are illustrated with little pictures of animals whose nervous system is being equaled by that level of computer power. Thus we move up from bugs, to mice, to cats, to humans. A glance

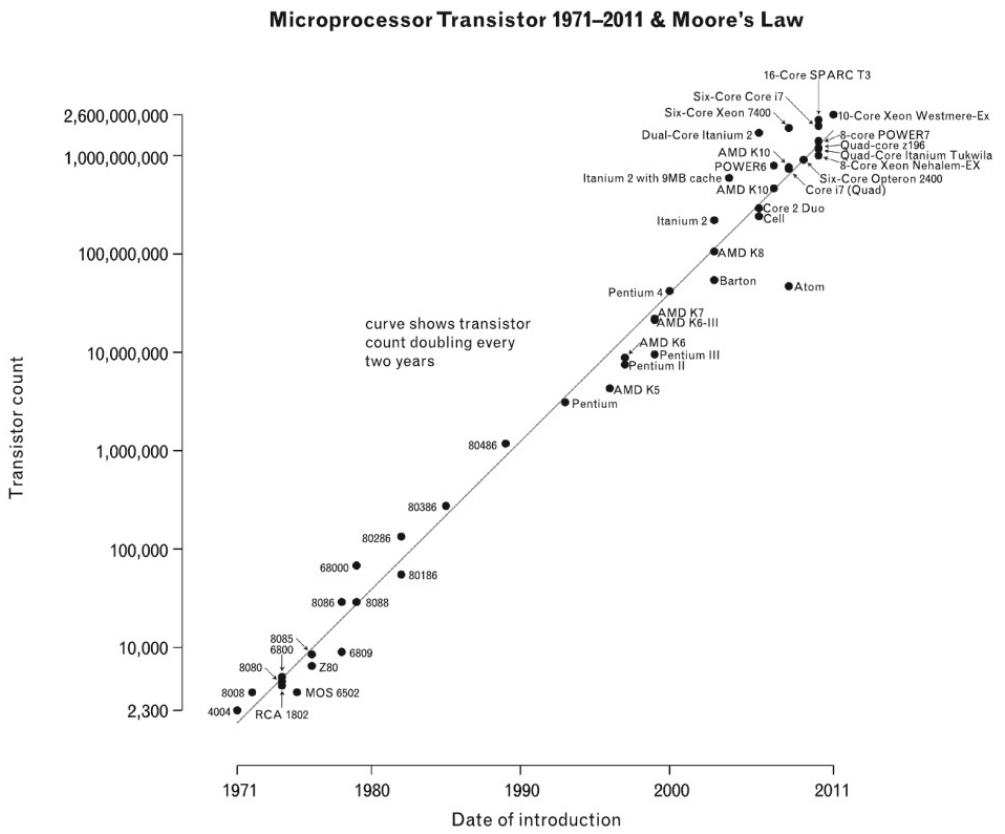

FIGURE 1.2: Transistor counts for integrated circuits plotted against their dates of introduction. The curve shows Moore's law—the doubling of transistor counts every 2 years. The y-axis is logarithmic, so the line corresponds to exponential growth. Figure from Wikipedia entry on "Moore's Law"; author "Wgsimon"

at the vertical axis of such graphs suggests that human brain power will be exceeded in the year 2030, 2050, 2075, 2100 . . . pick one.

One conclusion of this book is that equating brain power with computer power is not easy. A facile equating of "calculations per second" with brain power is meaningless. The basis for comparing computer speed and nervous system operation is unclear. The most likely source, probably buried deep in collective memory with subsequent wishful thinking, is based on a 1943 landmark model of brain computation by Warren McCulloch and Walter Pitts. This model, its genesis, and its influence will be discussed in detail in Chapter 7. The argument of McCulloch and Pitts was that nerve cells in the brain were tiny binary elements computing logic functions. Since computer components do the same thing, it then became reasonable to compare the size and speed of a nervous system with computer power.

Unfortunately, this splendid idea was not true. A great deal of the actual work of the nervous system in operation is based on "analog" processes, like many tiny tin cans connected by microscopic strings. Millions of operations can be required for a digital system to match the performance of a simple analog system for a particular function, as shown by the telephone.

PROBLEMS WITH EXPONENTIAL INCREASES

No tree grows to heaven.

— Wall Street Aphorism

If computer power increases by a factor of 2 every year, it gets big very fast. In 5 years, "power" will increase by a factor of 32, or 2^5; that is, roughly going from brisk walking to the high speed lane on an expressway. Many other phenomena show periods of exponential growth, but claims about the long-term results of unending exponential growth have a questionable history.

If a colony of *Escherichia coli* bacteria doubles in number every 20 minutes, as they can in a Petri dish, it is easy to show that, after a week, bacteria will be up to our knees and, after another week, will submerge Mt. Everest. Observation suggests this prediction seems not to be true. However, equivalent claims have been made for bugs, mice, rats, or humans whenever an alarming political point needs to be made.

In one familiar, politically charged example, human exponential population growth, the "Population Explosion," of great concern 30 years ago, does not currently exist outside of a few of the world's poorest countries, mostly in Equatorial Africa. At this time, most developed countries do not produce enough children to keep their population constant in the long term. Based on current reproduction rates, there will be few Italians, Russians, Japanese, or Koreans left in a millennium if these countries' birth rates do not increase.

Data for doubters: Fertility is measured in the average number of children produced per female. A rate of a little over 2, the "replacement rate" corresponds to a stable population in the long term. As of 2015, data from the CIA's *World Factbook* gives the estimated reproduction rate per female as 1.61 in Russia, 1.43 in Italy, 1.4 in Japan, and 1.25 in South Korea, values far below those required for stability. If this trend continues, the population of these countries will vanish in few thousand years. Places where a genuine long-term population explosion exists at present include Niger, 6.16; Uganda, 5.89; Nigeria, 5.19; and Tanzania, 4.89. In the short term, even in the developed world, population will still increase for a while as the death rate drops and children enter the reproducing population.[3]

Sometimes exponential growth abruptly stops for physical reasons. Passenger air travel has not increased in speed after a brief exponential phase since the introduction of jet aircraft in the 1950s and, in fact, has dropped since the Concord was taken out of service in 2003.

Reasons for the termination of exponential growth vary. In the case of bacteria, there are limits on available nutrients. In the case of airplanes, the economic and environmental problems of supersonic flight restrict commercial aircraft to subsonic speeds. In the case of human reproduction, the reasons for the end to an exponential increase in population are not certain, but the actual effect seems to be universal as economic development proceeds, and this even has a name: the *demographic transition*.

What will stop the simple form of Moore's Law is not clear, but past history suggests it will be something and not too long from now.

EVOLUTION OF INTELLIGENCE IN MAN AND MACHINE

Our telephone example showed the complexity and number of the digital operations that are required to match the performance of even a primitive analog system. But the flexibility allowed by digital systems more than makes up for it. Modern humans seem to have been around for between 100,000 and 200,000 years. What we think of as genuine "human" behavior, for example, the advanced symbolic behavior demonstrated in cave paintings, is much more recent, a few tens of thousands of years. Agriculture is less than 10,000 years old.

So, if our brain is so advanced, why did it take us so long to get smart?

Perhaps we can accomplish more and live better than our paleolithic predecessors because, with time, some powerful cognitive software has been developed, slowly, that allows genuine computational power and flexibility to be implemented through learning, culture, and personal experience in the human brain.

This observation is why we would like to call such behavioral software "computation." It is a powerful tool for making brains work better. The process of human cognitive evolution, allowing for the development of such fine software, is largely unknown. Other animals—mammals and the primates in particular—have many of the biological and behavioral components of human intelligence, but the whole package that gives rise to flexible and complex cognitive behavior is only there in *H. sapiens*. The story of human cognitive evolution is, sadly, invisible.

Therefore, consider human cognition as if it was a new set of computer software. "Cognitive" software is recently evolved, and it is full of bugs. For example, words are ambiguous, precise definitions of almost anything seems to be impossible, and there are multiple languages with arbitrary grammars and different vocabularies. The Tower of Babel does not correspond to good communication engineering design principles.

Overall, our cognitive software seems to be equivalent to the "alpha release" of a software product; that is, the earliest working version that ordinarily never leaves the software developers because it is so unreliable. However, even at such an early stage of development, it has proved of great value as the biological success of our species shows. We have gone from a few thousand furtive primates in East Africa a hundred thousand years ago to billions of humans worldwide now.

As mentioned earlier, physical tool-making "software" seems to be substantially older and is much better debugged. It is significant that modern tool use of all kinds is highly standardized and does not have the huge variety of mutually incomprehensible languages that plague human culture. Any human can learn to use a screwdriver, drill press, or lathe with minimal instruction. The language of technology seems to be about as close to a rich, universal human language as we currently have.

Many other human cognitive abilities are superb. The best examples come from the "software" and specialized hardware analyzing our senses: vision, audition, touch, balance, and muscle control. "Making sense" of raw sensory data is called *perception*. These abilities have been developed over hundreds of millions of years and are now highly optimized—so highly optimized that we often do not appreciate how good they really are because we, as their possessors, find them transparent and correct in operation.

Therefore, instead of being the alpha release, sensory and perceptual "software" corresponds to a late release with a high version number, well debugged, fully developed, and highly reliable. However, it took hundreds of millions of years to get to this degree of performance. In consequence, much of human cognition, even in the purest of pure mathematics, can be shown to rest firmly on highly developed sensation and perception, as discussed in Chapter 15. The "cognitive" and the "perceptual" systems are not separable in practice—fortunately for us—since they cooperate effectively.

COMPUTER EVOLUTION

Computers are also subject to the forces of evolution, both economic and intellectual. How have digital computers evolved since they first saw the light of day during World War II?

In biology, evolutionary success is based on reproductive fitness. Organisms that are the most successful in propagating their genes into their descendants succeed. A similar measure for a computer is how many computers get sold and for how much.

The number of individual computers has increased from a handful to billions. Their hardware is now almost completely reliable; their software is much less so. They can retrieve and store huge and increasing amounts of data with aplomb. They do logic and arithmetic millions of times faster and more accurately than any human could.

Development of this particular cognitive skill set is not an accident. This large number of expensive machines would not have been sold if they were not useful. But, in the process, machine evolution has increased rather than decreased the differences between machine "intelligence" and human "intelligence." Economic forces drive this separation: computers do for humans what humans do badly.

At this point in the development of machine intelligence, humans and computers are developing a cooperative, perhaps eventually even a symbiotic relationship. Each does for the other what the other does not do well; that is, it is different skills that make cooperation valuable. Such a situation is typical of biological symbiosis. The resulting combination can be more powerful than either alone. Computers are still far away from performing the cognitive skills that humans do so well: perception, association, memory-based intuition, knowledge integration. Developing human-like machine intelligence, however it is done, will be the next large step in computer evolution.

AN UNCERTAIN VIEW OF THE FUTURE OF COMPUTATION

In the shorter term future, everyone agrees on what will happen: computers will get faster, cheaper, and even more ubiquitous. Computer power and effective software will soon make almost all humans unemployed—even more, unemployable. The only jobs left for humans will be as politicians, artists, athletes, and lawyers. When intelligent machines ultimately establish a rational, efficient, and benevolent government, lawyers and politicians will also become unemployed. The near future may then become a pleasant and rewarding place to live.

This book is more concerned with the longer term, and its goal is to provide some perspective on how genuine machine intelligence might emerge, what it would look like, and why it might look that way. It is my feeling that to build a machine with true, human-like intelligence, it will be necessary to build into computers some of the cognitive functions and specialized hardware used by the one and only example of intelligence: us. Humans are not digital computers. They do not act much like them nor should they. We have computers to act like computers.

A new technology has emerged from essentially nowhere to change the world in under a century. Futurists, philosophers, and fiction writers all agree that remarkable events will emerge from the evolution of machine intelligence. However, there is no agreement as to whether it will lead to catastrophe or transcendence. Trying to understand which it might be and why is one of the tasks of this book.

CHAPTER 2

COMPUTING HARDWARE

Analog

DIGITAL SYSTEMS VERSUS ANALOG SYSTEMS

Anatomy is destiny.

—Sigmund Freud

The previous chapter made the suggestion that computation is better defined more by the task it performed than the hardware it is built from. Computation constructs aids to cognition. This definition is especially appropriate for the large class of devices called *analog computers*. Some analog "computers" are hardly recognizable as computational engines because they are so familiar, so old, and their function seems so simple. But they can do operations that unaided human perception and cognition cannot do as well and are computers by our definition. A central issue to this book is practicality.

At present, when "a computer" or "computation" is mentioned, most people think of a particular class of electronic machines working largely by logic and binary arithmetic. There are alternatives. One is the less familiar analog computer whose computations are based directly on the detailed physical components and construction of the device. Analog computers do some important operations simply and quickly. They have been around much longer than digital computers.

There are also hybrid intermediate forms, combining aspects of both analog and digital computation. Some of these hybrids are inspired by the hardware of the human brain and the kind of "cognitive software" that seems to run well on it. "Brain-like" computation will not necessarily supplant other forms but might sometimes prove a useful alternative, perhaps uniquely well-suited to dealing with problems of interest to humans. We will present some ideas in this area later.

MEET THE MAIN CHARACTERS

In 1969, the Hewlett-Packard Electronics Catalog was an expensively produced 672-page hardbound book covering the full range of HP's electronic products, from signal generators, to

voltmeters, to microwave hardware of all kinds. Most of the items described were electronic test equipment.

Pages 103–129 formed the section of the catalog describing HP's small line of digital minicomputers. This section is around 4% of the HP catalog, suggesting the relative importance of analog and digital components in HP's corporate consciousness in 1969.

Most striking from our current point of view, digital computers of the familiar kind were not the only option. From the beginning of the computer section:

> Computers may be divided into two main classes, "digital" and "analog". A digital computer is one that obtains the solution to a problem by operating on the information in the form of coded numbers, while the information processed by an analog machine is in the form of physical analogs such as voltages or shaft positions that represent numbers.[1]

MECHANICAL ANALOG COMPUTERS

Analog computers are far less familiar today than they were to the readers of the 1969 HP catalog. Consider mechanical analog computers, first for their intrinsic interest; second, for their long history; third, because they show how specialized complex mechanical analog computers are; and fourth, because they work successfully and reliably on important problems. Not so long ago, they were considered to be worthy competitors of the digital computers. In Wikipedia, there is a separate entry for analog computer but none for digital computer. As of 2015, an entry for "digital computer" in Wikipedia is redirected to "computer." Apparently it was held to be identical to the "computer" entry.

AN EARLY ANALOG COMPUTATION: BALANCES

> And this is the writing that was written, MENE, MENE, TEKEL, UPHARSIN. . . .
> TEKEL; Thou art weighed in the balance and art found wanting.
>
> —Book of Daniel (5:25 and 5:27; King James Version)

One of the earliest examples of an analog computer is the balance, familiar to all from the scales in the hands of statues of Blind Justice found in court houses, law offices, and on multiple TV shows.

A balance is simple: two identical pans are connected, usually with strings or small chains to the two ends of a beam that has a swivel and support at its center (see Figure 2.1). The pans move up and down as the beam tilts to one side or the other. If one pan goes up the other goes down. An item is placed in each of the pans. If one item is heavier than the other, the pan containing the heavier item will sink.

Therefore, the first thing learned from a balance is which of the two items is heavier. The shape, form, or composition of the items does not matter. The task is very general. The only parameter of concern is weight.

There is one additional useful bit of information often available from a balance. Some balance designs will stabilize with the pans not quite at the same level, providing a useful measure of small differences between the weights. The vigor with which one pan drops and

FIGURE 2.1: A classic balance. Balances are designed to tell which of two weights, one placed in one pan and one in the other, are heavier. Three thousand-year-old images of balances almost identical in construction and certainly identical to this in spirit have been found in Egypt. In one well-known example, the heart of the deceased is weighed against a feather to see if the good and bad parts balance. "If the heart did not balance with a feather the dead person was condemned to non-existence." Figure from Perhelion/Wikimedia

the other rises also provides qualitative information on how disparate the weights of the two items are.

Many variations of this simple balance are in daily use. If a set of standard weights is available, the weight of an item can be determined by finding the group of standard weights that most closely matches the new item. Doing this measurement accurately requires a set of calibrated weights. Members of standard sets of weights have been found in the archeological record from Egypt as early as 5,000 years ago.[2]

It is also possible to modify a simple balance to "multiply by a constant." Most pictures of balances show the beam pivoted in the center, as in Figure 2.1. If the pivot and support are moved to one side, say, one-fourth of the way from one end, the device will balance when the weight in the distant pan is one-third of the weight in the nearer pan. Further refinements lead to the balance scales with sliding weights and off-center pivots found in doctor's offices. Because of their intrinsic accuracy, reliability, and long-term stability, complex balances are still used for accurate weight measurements in laboratories and in high-precision physics experiments.

As an example, the journal *Nature* published "Tough Science. Five Experiments as Hard as Finding the Higgs."[3] The article was concerned with experiments addressing important issues that were difficult to do. Two of the five experiments used balances for their measurements. One is looking for the "wrapped up" extra dimensions of space, beyond our familiar three (or four, if you include time) predicted by the string theory of physics, the current best candidate for a "theory of everything." The other uses a simpler balance to allow redefinition of the world's weight standard, the standard kilogram in Paris, in terms of more fundamental physical constants instead of an actual object.

"PROGRAMMING" A BALANCE

One comment that is often made about "computers" (i.e., digital ones) is that they are programmable. But even a simple balance can be programmed over a narrow range of tasks by the way weights are added to the pans, just like different input data determines the results from a program.

Assume a set of standard weights, a balance, and an object to be weighed are at hand. One "software" strategy to determine the weight of the object is to test, one after another, all possible groupings of the standard weights to find the set that is closest to the weight of the object. This technique is slow since there can be many different combinations of weights. The weight of the object is approximately the sum of the weights of the best matching set.

A faster strategy would be to start with the largest weight that is lighter than the object to be weighed and add weights successively until the object in the other pan becomes lighter. The last weight added is then removed and a smaller one is used until the scale is in balance. The weight of the object is the sum of the weights left in the pan. This strategy depends on having an appropriate set of values for the weights.

If the two pans do not match exactly, it is possible to interpolate. If an added weight is too much and the next smaller weight is too little, the exact weight of the item must lie somewhere in between.

Another extension is to observe that *negative* weights can be formed by placing a weight in the same pan as the object to be weighed.

The point is that even an analog device as simple as a balance allows for a limited degree of "programming" in use.

Chapter 5 discusses the role of *analogy* when humans try to work with complex system. There can be complex social interpretations of the balance found in images and statues of Justice. Ideally, Blind Justice weighs only the totality of the evidence and the law in the case and chooses the winner to be the "heaviest" despite the importance of the individuals involved. Whatever "justice" might be in this analogy, it is not a physical weight, but even so the meaning of the analogy is clear and useful.

THE ANTIKYTHERA MECHANISM

A remarkable early example of a complex analog computational device is the *Antikythera mechanism*, a discovery that revolutionized our understanding of the state of information technology 2,000 years ago. The mechanism was discovered in 1901 by Greek sponge divers diving off the island of Antikythera between Crete and the Greek mainland; they found the remains of a Roman shipwreck in about 180 feet of water. The Antiythera shipwreck is conjectured to have occurred around 80 years BCE and seems to have involved the shipment of loot from Greece to Rome. The divers saw many statues and fragments of statues strewn on the sea bed. Amid the statues, jewelry, weapons, luxury glassware, furniture, and other treasures was a corroded bronze mass about 6 by 7 inches in size. It was taken to the Archaeological Museum in Athens. As the mass dried, it fell into several pieces, one fragment exposing an arrangement of several large interlocking gears (see Figure 2.2).

It was clear from the beginning that the mechanism was a very complex mechanical device. The more it was studied, the more remarkable it became. Recently, it has been the subject of a large research project using modern analysis techniques including high-intensity X-ray tomography and specialized digital optical imaging of its surfaces.[4]

FIGURE 2.2: Antikythera mechanism fragment (fragment A). The Antikythera mechanism consists of a complex system of 30 wheels and plates with inscriptions relating to signs of the zodiac, months, eclipses, and pan-Hellenic games. National Archaeological Museum, Athens. Figure from Wikipedia entry "The Antikythera Mechanism," author "Marsyas"

As the abstract of a 2006 *Nature* paper comments, "The mechanism is technically more complex than any known device for at least a millennium afterwards." Professor Michael Edmunds of Cardiff University, the leader of the team commented that, "in terms of historic and scarcity value, I have to regard this mechanism as being more valuable than the Mona Lisa."[5]

The bronze mechanism was originally housed in a wooden framed case about 12 by 7 by 4 inches. The gears were probably driven by a hand crank, now lost. Turning the hand crank would cause all interlocked gears within the mechanism to rotate, resulting in the calculation of future positions of the sun and moon and other astronomical events, such as phases of the moon and the times of eclipses (see Figure 2.3).

This mechanism is remarkable for many reasons. The parts are accurately made. There were 31 recovered gears, but probably at least 37 were in the intact device, perhaps more. Gears were hand-cut. Two had more than 220 triangular teeth. It must have been extremely difficult to cut gears of such size and precision with available tools in the first century BCE. The device contained its own documentation, and, using modern optical techniques, it was possible to recover some of the "instruction manual" that had been engraved on various flat metal surfaces.

The complexity of the calculations embedded in the gearing was remarkable. Figure 2.3 provides some idea. Only one example: the Greeks assumed that orbits of heavenly bodies must be circular because that was a perfect geometrical form even though the observed motions did not

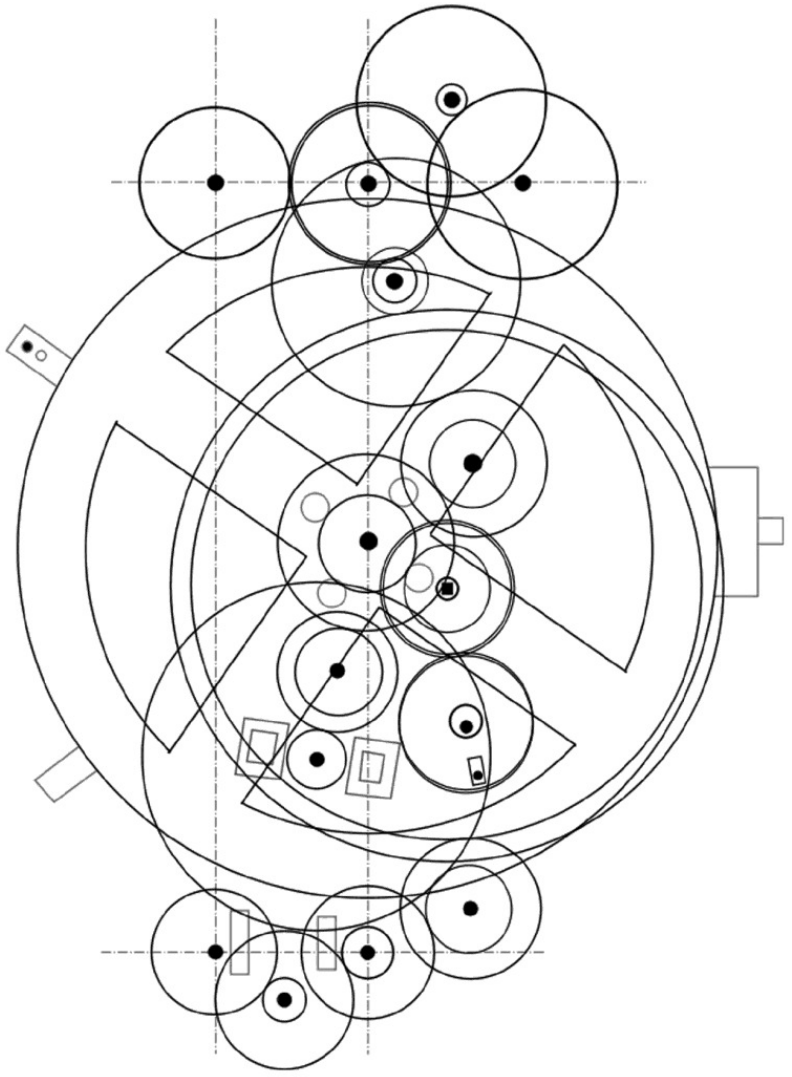

FIGURE 2.3: Reconstruction of the Antikythera mechanism's internal gearings, arrived at by high-intensity X-rays and inspection. Figure from Lead Holder/Wikimedia

agree with a circular orbit. A mechanism proposed by the astronomer Hipparchos in the second century BC explained the irregularities in the predicted position of the moon in the sky caused by its actual elliptical orbit. A whole series of complex planetary movements called *epicycles* (circles within circles) were developed to predict accurately the movements of the moon and planets. A complex set of gears including pin-in-slot gearing and epicyclic gearing was built into the Anitkythera mechanism to compensate for these observed astronomical complexities.

In summary, the device seems to have displayed

- Positions of the sun and moon over the year. The moon's position incorporated prediction of the moon's actual orbit.
- Phases of the moon, demonstrated by a rotating half-black, half white ball, as in some modern clocks.

- Date of eclipses of the sun and moon a generation into the future.
- Dates of Olympic and a number of other games held by different Greek cities.
- Rising and setting of several stars.
- Compensation for leap years.
- The device may also have predicted the locations of the planets, but many of the necessary gears are missing.

CONTRIBUTIONS TO CLASSICAL TECHNOLOGY? NONE!

There are hints in the Classic literature of such complex devices coming from the workshop of the great Greek scientist and engineer, Archimedes (287 BCE–212 BCE) in Syracuse (Sicily). The Roman orator, lawyer, and politician Cicero (106 BCE–43 BCE) in the first century BCE describes mechanical devices that predict the movements of the sun, moon, and planets. The devices were brought to Rome by the Roman general Marcellus as loot after Archimedes, their builder, was killed by accident in the siege of Syracuse in 212 BCE. One device was kept as a prized family heirloom and seems to have been demonstrated working for many years afterward.

Are there other devices? How many were there? What did they do? There are no signs of "engineering change orders" in the complex Antikythera mechanism itself—that is, the afterthoughts, corrections, and last-minute updates that almost any complex device accumulates with experience. Clearly, this device was a highly developed design that came from a long technological tradition but exactly where the factory was that made it is not clear. Suggested possibilities range from Syracuse, perhaps a continuation of the workshop of Archimedes, to Corinth, based on the lettering and vocabulary found in the "instruction manual."

It might be useful to stop being impressed with the amazing technology of the Antikythera mechanism and ask if these machines were ever actually used for anything. The Greeks developed them and the Romans treasured and respected them. Yet there seems to have been no attempts to develop them further or modify them to do other things. The Antikythera mechanism was never used for anything practical. Why should it be? The answers it gave projected eclipses and games for a generation into the future. Once it has been consulted—presumably by initializing it and turning its crank—the many answers it gave could be written down, and there was no more use for the machine for a while. It is a wonderful device with no obvious practical function, except perhaps for the phases of the moon or the positions of the moon and sun in the Zodiac. They were rich men's toys for enhancing the owner's social status. Nowhere do they seem to be held to be more than exotic curiosities.

There have been many theories about why this highly advanced technology became a dead end. Yet, in contrast, simple balances in one form or another have been in daily use all over the world for more than 4,000 years and are still in use today.

There is weak evidence that the Antikythera mechanism tradition continued in complex clockwork and astronomical devices in the Byzantine Empire and into the Islamic world as late as 1000 CE but, again, as rare, impractical, though valued devices. But suddenly, 1,500 years after the Antikythera mechanism, clocks calculating astronomical information of equivalent complexity, often coupled with other complex automata, started appearing throughout Medieval Europe. Whether there was a connection to ancient Greek technology through Byzantium and the Moslem Middle East is unknown and undocumented, but possible. But in Europe, unlike

in the Classical world, complex clocks spread rapidly and became commonplace and useful as a way of organizing events in a day, week, or month.

THE FIRST COMPUTER SIMULATION (!)

If Greek and Roman technology had taken a different route, the technology of the Antikythera mechanism could have developed quickly into:

- Accurate clocks
- Astronomical instruments
- Mechanical arithmetic calculators
- Surveying instruments
- Navigation equipment
- Or even fire control devices for catapults

However, there is one important application of the device that was culturally of great value in Classic times and that may have played a significant part in its initial construction and design: the Antikythera device is an early example of a *computer simulation*. It realizes in gears and wheels an abstract model of the movement of the heavenly bodies and predicts actual observations accurately. The owners and builders could look at the device, at the heavens, compare them, and say "Yes, we got it right." That was a major scientific accomplishment and must have been of great satisfaction at the time.

SLIDE RULES

A familiar and at one time ubiquitous analog device that computed in a quite literal sense was the slide rule. When I went to high school, every chemistry student had to buy, and sometimes even learn to operate, a 10-inch slide rule. At that time, they were sold in drug stores.

The slide rule was a simple device capable of rapidly providing low-accuracy solutions to multiplication and division problems. In high school chemistry, this usually meant working with the various "gas laws." As soon as classes were over, the students could sell their slide rules to the next generation or take them with them to college where, I suspect, they did not receive much use then either.

A simple rule had a central slide that could be moved back and forth and a number of printed numerical scales on both the fixed and moveable parts of the slide rule. A sliding hairline cursor was provided to give more accurate readouts of numbers and to allow reading the values from several scales.

Complex slide rules were things of beauty, with engraved ivory scales attached to dense wood cores. They could go far beyond simple multiplication and division. They could provide approximate values for most trigonometric functions and compute a number of more esoteric functions like hyperbolic sines and cosines, fractional powers, logarithms of logarithms, and the like.

Their major drawback was their limited accuracy. Values were represented as physical locations on a scale. Answers had to be read off the scale, using the cursor, and were good only to two or three decimal places. An obvious way to increase accuracy was to increase the length of

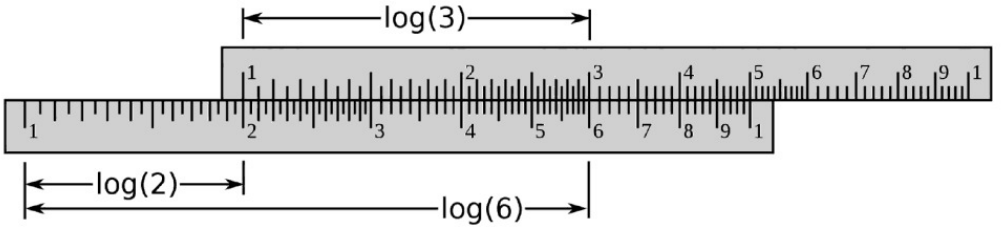

FIGURE 2.4: Operation of slide rule when multiplying 2 × 3. The start of the sliding upper scale (1) is aligned with 2 on the lower scale. Located under 3 on the upper scale is the correct product, 6. All multiples of 2 are computed at the same time at different locations of the upper scale relative to the lower scale. Located under 1.5 on the upper scale is the product of 2 × 1.5; that is, 3. 2 × 2 = 4 is located under 2 on the upper scale; 2 × 4 = 8 is located under 4 on the upper scale, and so on. Figure from Jakob. scholbach/Wikimedia

the slide rule from the common 10 inches (25 cm). Such longer rules were made for special purposes but were unwieldy. Circular slide rules were made in another attempt to increase accuracy, especially for simple multiplication and division, but, again, the devices were hard to carry, and the increase in accuracy was small.[6]

Slide rules have been around for several centuries. After John Napier (1550–1617), a Scottish mathematician, invented logarithms, William Oughtred (1575–1660) used two such scales sliding beside one another to perform direct multiplication and division. He is credited with inventing the slide rule in 1622.

Their most common operation is based on the properties of logarithms when two numbers are multiplied together. The logarithm of a product, say 2 × 3, is the sum of the logarithm of 2 plus the logarithm of 3. A multiplication, generally tedious to do by hand, has been changed into a much easier summation.

The slide rule did this summation mechanically (as shown in Figure 2.4). The two major scales of the slide rule, traditionally the "C" (sliding) and "D" (fixed) scales, were marked off with numbers whose distance from the left side of the scale was proportional to their logarithm. If someone wanted to multiply 2 × 3, one scale was moved relative to the other using the slider. As shown in Figure 2.4, the beginning of the slide (1) was moved to the location of the first number to be multiplied on the lower scale (i.e., 2). The other term in the multiplication, 3, was located on the slide, and the correct answer to the problem, 6, was read out as the value at the location beneath 3 on the bottom scale.

One valuable feature of the slide rule is that this operation worked for any pair of numbers, including decimals that were time-consuming to compute by hand.

A 20TH-CENTURY MECHANICAL ANALOG COMPUTER: NAVAL FIRE CONTROL

Probably the most complex electromechanical analog computers of all time were designed, built, and used extensively in the first half of the 20th century as fire control computers for the US Navy, used for accurate aiming of naval artillery.[7]

For more than 40 years, mechanical analog computers provided the US Navy with the world's most advanced and capable fire control systems for aiming large naval guns at either surface or air targets. This analog computing technology provided a significant advantage to

the US Navy in World War II. The Mark IA naval fire control computer of this era was an "an electro-mechanical analog ballistic computer." The "electro" part referred to internal motors, relays, servomechanisms, and gyroscopic stabilization. The actual computing operations were largely mechanical: accurate constant-speed motors, disk-ball-roller integrators, nonlinear cams, mechanical resolvers, and differential gears. Fire control analog computers were large: the Mark Ia was a box 62 inches long, 38 inches wide, and 45 inches high, and it weighed more than 3,100 pounds (see Figure 2.5).

Accurate naval gunnery presents a complex problem. The computer had to take into account the relative speeds and bearing of the ship and target; the wind, range, relative elevation of ship, and target for air and land targets; aerodynamic and ballistic characteristics of the shell; and the

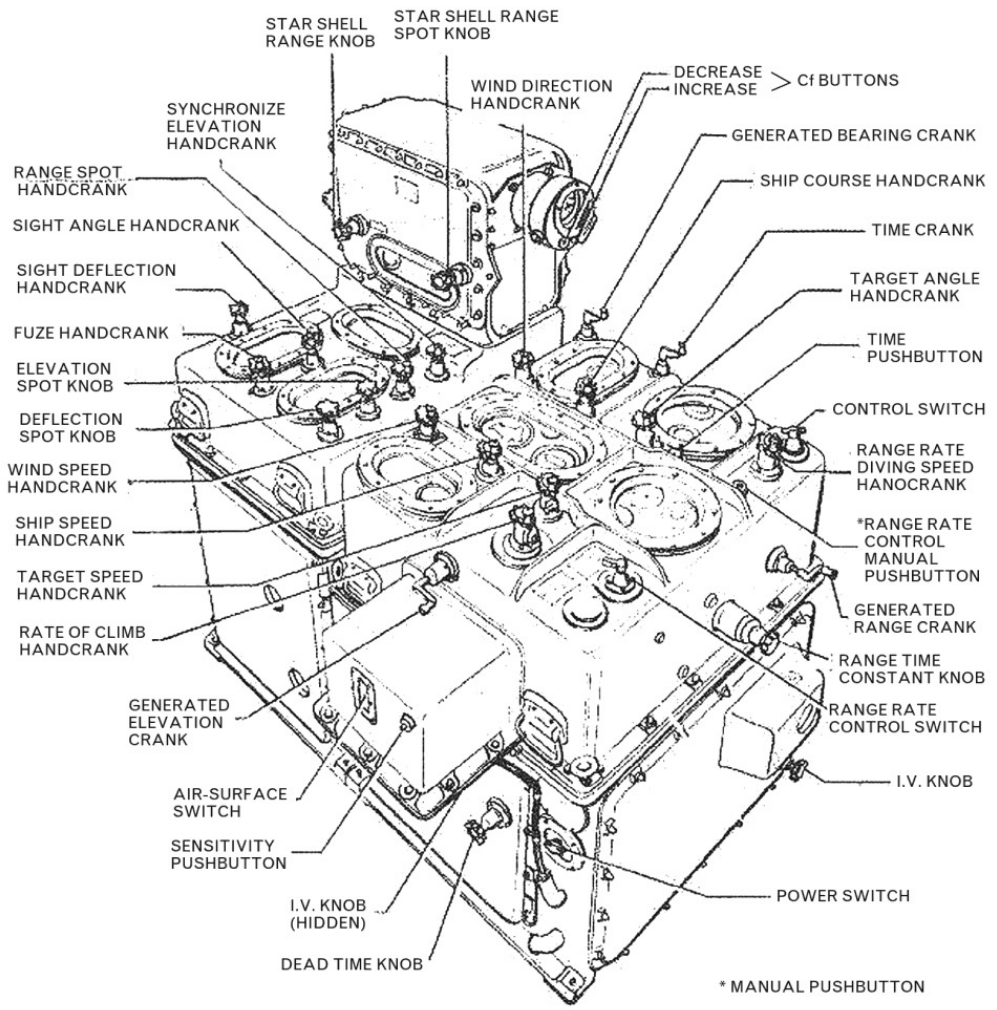

FIGURE 2.5: Description and Operation, 29 June, 1945, OP 1064, Prepared for the Department of the Navy, Bureau of Ordnance, by the Ford Instrument Company. NAVPERS 10798-A, NAVAL ORDNANCE AND GUNNERY, VOLUME 2, FIRE CONTROL. Figure from Jostikas, original uploaded by Hohum on en.Wikipedia; public domain

gun elevation angle, as well as time of flight to allow fuse setting. Worse, the ship itself rolled constantly back and forth.

After about 1940 and the installation of automatic control, the guns could fire with precise aiming at any time, releasing accurate aiming from the difficulties involved in synchronizing firing with the rolling of the ship. . . . The Bureau of Ordnance considered the Computer Mark 1 to be very successful.

High accuracy was possible with slow targets. For example, the battleship *Washington* is said to have achieved more than nine hits on the Japanese battleship *Kirishima* out of 75 rounds of 16-inch shells at 19,000 yards (nearly 11 miles) range in a night battle at Guadalcanal in 1942.[8]

Automatic fire control was a long way from the svelte Antikythera mechanism analog astronomical computer but it used many of the same basic mechanical components. These fire controllers were the ultimate form of mechanical analog computation. They were not replaced in the Navy with the now more familiar digital computers until the 1970s. They did their job exceedingly well.

It is mildly ironic that the US Navy developed and used impressive mechanical analog computers for accurate gunnery while at the same time the US Army sponsored the development of the first US digital computer, the ENIAC, designed to allow accurate firing of new land-based artillery pieces because current, largely mechanical, arithmetic computational technology was inadequate to do the job.

Some of the reasons for this observation are important. The basic physical properties of large naval rifles and shells did not change significantly for decades. All the important physical parameters involved in computing shell trajectories were well understood and tabulated and could be incorporated directly into the specialized machinery of the fire control system for the Navy.

However, the changing needs of war required rapid development of many kinds of new artillery pieces for the Army. Effective use of these new types of artillery required the construction of new "firing tables" to allow accurate aiming that provided some hope of hitting a distant target. Computing trajectories was a straightforward application of well-understood physics and mathematics. Detailed behavior of the flight of the shells was affected by many parameters—barrel elevation, amount of powder, shell type, air temperature, wind direction, relative heights of target and artillery, and more. Solving the required equations by mechanical calculators for all the important configurations of shell and artillery piece could take well over a year, an unacceptable delay during wartime.

Flexibility of computation, the ability to compute the answers to new problems, suddenly became as important as accurate performance on old problems. Because the software for a mechanical analog computer was realized in complex, precisely machined hardware, it was difficult to change it easily and quickly for a new application.

It seems initially highly unlikely that a mechanical device constructed to aim artillery pieces accurately would give rise, in only a few years, to a word processor, a video game, or a social network. Yet this is exactly what happened to flexible digital computers but not for inflexible analog computers.

The construction of the first digital computer (the ENIAC) was sponsored by the US Army for the specific problem of solving the equations of motion of artillery shells to provide the data for firing tables. But computers rapidly became capable of far more than this due to the genius of the group that worked on the ENIAC project, some of the greatest minds in mathematics and physics of the 20th century. This project also developed a surprisingly strong connection to early brain theory, as will be shown later.

ELECTRONIC ANALOG COMPUTATION

Until about 1930, analog computers were largely mechanical and, in consequence, were difficult to modify for new applications. Attempts to make more flexible analog systems were made for a few important classes of problems. The *differential analyzer*, a largely mechanical analog computer, was developed during the 1930s specifically to solve differential equations, and it used mechanical means to do it. Differential equations are ubiquitous in engineering and physics and describe the behavior of many important physical systems. The behavior of artillery shells is only one example. In practice, differential equations often must be solved numerically, making them a natural and critical application for a computer of any kind. Solutions of differential equations are required for fire control systems, and these solutions could be obtained by either analog (Navy) or digital (Army) technologies. But analog and digital computers work with these equations and give solutions in utterly different ways, as different as the digital and analog telephone systems discussed in Chapter 1.

As a relevant aside, one that connects to cognitive science, an early developer of the practical differential analyzer was Vannevar Bush, then on the MIT faculty, a founder of Raytheon, and a key scientific policy advisor to the US Government during the World War II era. After the war ended, in a remarkably far-sighted bit of computational speculation, Bush also proposed an associative linking system to be used for information storage and retrieval from large datasets (the "memex") that was a precursor of the architecture of the World Wide Web. Douglas Englebart and Ted Nelson, early pioneers in the human use of computers, were influenced by Bush's ideas in their development of information retrieval mechanisms such as hyperlinks and Nelson's hypertext. The ultimate realization of large, linked, associative networks is, of course, the Internet. Association, perhaps the key cognitive operation, is discussed in detail in Chapter 9.

The 1930s' largely mechanical differential analyzer was not easy to program or use. However, after World War II, a whole range of versatile, reliable, and inexpensive electronic devices that could be used to build analog computers became widely available.

Calculations difficult for mechanical devices to perform become simple using analog electronics. Consider an example: suppose the problem is to multiply 7×3. We have at our disposal a resistor, a voltage source, and meters to measure voltage and current. Set the resistor to have a value of 7 ohms. Adjust the voltage so that 3 amperes flows through the resistor. Then, using Ohm's Law from high school physics, the voltage across the resistor, measured with the voltmeter, is 21 volts, the product of 3 and 7 and the correct answer.

Another analog multiplier is the calibrated electronic amplifier, very similar to those used in stereos and iPods to drive loudspeakers or headphones. Set the gain of the amplifier—the volume control—to a gain of 7. Put 3 volts at the input to the amplifier. Then 21 volts appears at the output, again the product of the gain, 7, times the input voltage, 3 volts. (These are unrealistic examples, but the basic concept is correct.)

The amplifiers used in analog computers were initially specifically designed for analog computational applications and were called *operational amplifiers*, or, more commonly, *op-amps*. They are found nowadays in large quantities in virtually every home, buried anonymously in consumer electronic devices such as TVs, stereos, telephones, and answering machines. They cost pennies apiece.

Other electronic components can be added to the mix—capacitors, inductors, diodes—that expand the range of possible basic analog computer functions. With a few components connected to the op-amps, they can perform a myriad of useful electronic functions: for example, stable, high-quality, programmable gain amplifiers, differentiators, integrators, and comparators. It now became possible to approximate the solutions of differential equations directly by

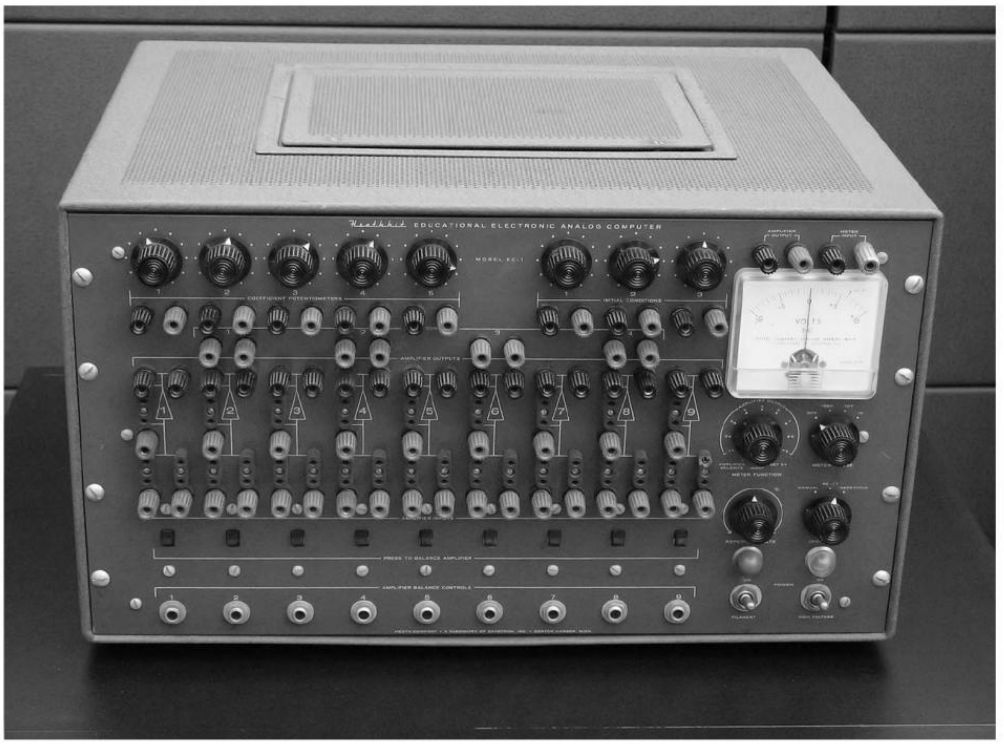

FIGURE 2.6: The Heathkit EC-1 Educational Analog Computer was introduced in 1960 and could be purchased assembled or as a kit for $400. The front panel provides sockets for the patch cords to connect to voltage sources, precision resistors and operational amplifiers. Amateur electronics experimenters before 1993 were familiar with the extensive Heathkit line of "do-it-yourself" electronics from the Heath company in Benton Harbor, Michigan. These kits were designed to be built at home.

electronic hardware simply by connecting together several op-amps and other devices in circuits designed to realize the specific equation parameters needed.

Soon, electronic analog computers (see Figure 2.6) were made by combining many op-amps and supporting hardware in a large cabinet. The components could be connected together by *patch cords*, wires with plugs at both ends. The plugs fitted into the various jacks on the components, and a proper set of connections realized the analog computer program. Such a system showed orders of magnitude more flexibility than a mechanical analog device ever could.[9]

In the 1960s and 1970s, an analog computer was much faster for many important problems than a digital computer, for example, solving most differential equations. An analog computer generated the answer values very quickly by direct electronic means, with no intermediate steps required. The electronic components, like the gears in the Antikythera device, were designed to "behave" the way they did because of the underlying laws of physics and the "programmer" who designed it.

A digital computer requires thousands or even millions of individual computer steps to solve even the simplest differential equation. Writing workable differential equation digital computer programs for real-world engineering problems can be difficult and are prone to some nasty and serious numerical malfunctions, as I found out.

During college, I had a summer job at a North American Aviation facility just south of the Los Angeles Airport (LAX). My job was to compute the heating of parts of the X-15 research airplane. The X-15 was designed to study the extreme heating that occurred when the aircraft

re-entered the atmosphere after a trip to the beginnings of outer space, 50 miles up. Having the X-15 wings fall off because they got too hot and weakened was considered undesirable. Predicting the heating of even a small wing segment required solving a set of nearly 30 differential equations numerically on a supercomputer of the day, an IBM 7090. It was necessary to numerically approximate the solutions. Unless the parameters of the approximation were chosen properly, the simulation "blew up," with values immediately going to plus or minus infinity. While getting my small part of this project to work I managed to waste several thousand dollars of the government's money and caused my boss some unhappiness as I discovered this bit of numerical pathology. However, I finally got it working. The wings of the X-15 remained firmly attached to the fuselage, as can be seen in the aircraft hanging from the ceiling of the Smithsonian Air and Space Museum. This display aircraft is still discolored in spots from the heat of its many re-entries, giving some idea of just how hot it got.

An analog computer requires essentially no individual computer operations at all. The reasons for the great difference in complexity between the two types are fundamental and unavoidable. For real-world applications that had to be fast—for example, in airplanes, spaceships, or nuclear reactors—analog computers in the early decades of computing were faster and much more reliable than digital computers. Since 1970, current digital computer software has not changed the number of basic operations required for a program to solve a problem. It still takes the same very large number of elementary digital numerical operations to compute an adequately accurate digital solution. However, the many orders of magnitude of increase in digital hardware speed conceal from the user the complexity of the programs that are required to solve even a simple differential equation on a digital computer.

CHAPTER 3

COMPUTING HARDWARE

Digital

'Tis the gift to be simple, 'tis the gift to be free
Tis the gift to come down where we ought to be,
And when we find ourselves in the place just right,
'Twill be in the valley of love and delight.

—Elder Joseph Brackett (Shaker Hymn, 1848)

Digital computers are built from hardware of stunning simplicity, so a word of warning is in order: this chapter on digital computer hardware is short and simple. The next chapter on digital computer software is long and complicated.

First, the essential thing you need to know about digital hardware is that it is based on devices with two states. Depending on your whim, the two states can be described as:

- ON or OFF
- 1 or 0
- High-voltage or low-voltage

Or, most portentous, and most significant for historical and expository reasons:

- TRUE or FALSE

The hardware development of the computer industry since 1940 has primarily been devoted to increasing the speed, reliability, packing density, and size of the physical devices realizing the two states.

Second, the other important aspect of digital hardware is connectivity. One two-state device has to be able to connect to other two-state devices. The connections have to allow the devices to influence each other's states in some manner. Practical issues involve how easy it is to connect one two-state device to another, how many devices can be connected together at one time, and how the devices can influence each other. Connections in real devices are almost always made

with physical connections; that is, wires, coarse or fine, or, at present, conductive traces on a silicon substrate narrower than a wavelength of light, a bacterium, or some viruses. This size allows many millions of binary devices to be placed on a small silicon computer chip.

The hardware is basically simple although beautifully engineered. All the complexity and, of course, the usefulness of this vast number of interconnected binary devices arise from the efforts of humans to tell the hardware what to do, as discussed in Chapter 4.

BINARY ELEMENTS THAT CAN BE USED TO BUILD COMPUTERS

The mainstream of two-state computer hardware evolution runs from relays, to vacuum tubes, to germanium transistors, to field effect transistors in the roughly seven decades from 1940 to the present. Although the hardware is vastly different, the basic function is the same.

RELAYS

Relays are electromechanical devices with a set of mechanical electrical contacts that can be opened or closed by the flow of current in a magnetic coil, an *electromagnet*. Because relays are On or Off devices, they were used to build completely general and functional digital computers in the early days of computing. The IBM Automatic Sequence Controlled Calculator (ASCC), called the Mark I by Harvard University, was a relay-based computer devised by Howard H. Aiken (1900–1973), built at IBM, and officially presented to Harvard in 1944.

A less well known early computer built largely from old telephone relays was constructed by Konrad Zuse (1910–1995) during World War II in Germany, completely independently of what we think of as the mainstream of computer development. Zuse is sometimes given credit for building the first programmable computer, although he was mostly concerned with numerical calculations.

Mechanical relays are very slow to switch state, switching in at best a few thousandths of a second—milliseconds—and were almost immediately superseded by much faster electronic devices that, even in their early forms, could switch between states in microseconds.

VACUUM TUBES

Vacuum tubes are called "valves" in England because they control the flow of electrons in the device just as a faucet controls the flow of water in a pipe. A simple "current flow–no current flow" pair of states in a valve, electronic or hydraulic, combined with the proper connections, is sufficient to build a computer. Vacuum tubes are not obsolete and are still widely used for specialized applications, most notably high-power amplification. The problems with vacuum tubes are those of incandescent light bulbs: they use a lot of electrical power, give off a lot of heat, and burn out frequently.

The first American computer, the ENIAC (1945), was built from 17,000 vacuum tubes. Keeping the ENIAC running for several days continuously using the unreliable vacuum tubes of the early 1940s was an impressive feat of practical electrical engineering. Essentially, all computers built until the mid-1950s used vacuum tubes as their binary element.

TRANSISTORS

Every reader of this book, unless they are a hermit, personally owns several million transistors of a type called *field effect transistors* (FETs) in their computers and cell phones, although probably most are not aware of it. Since readers own millions, perhaps billions of them, a brief description of how they work might be useful. The only other very small, useful things that most humans "own" in such large quantities are trillions of intestinal bacteria.

FETs were invented and patented in 1925 by Julius Lilienfeld (1882–1963), an Austro-Hungarian physicist who moved to the United States in the 1920s. It was not possible to build practical devices based on his design at the time he invented it, but his design was correct, workable, and patentable. These early patents caused Bell Labs legal trouble when they wanted to patent some of the technology that led up to the transistor.

To make an FET, a narrow channel is formed in an insulator. Suppose a voltage is applied to each end of the channel. In one kind of FET, electrons flows in the channel—a current—from the "source" of electrons, like the negative side of a battery, to the "drain" where electrons return to the positive side of the battery. Source and drain are vivid analogies that mean exactly what they seem to mean, except they describe electron flow and not water flow. Consider a hose with a source of water at the faucet and a drain at the other end. If the faucet is open, water will flow.

The electrical source and drain require a wire each, connecting to a voltage source or ground. Attached to the side of the channel is a third electrode called the "gate." Suppose electrons are flowing freely through the channel. The gate electrode is put next to and very close to the channel. The way this device works is a realization of the high school physics rule that "opposites repel." When a negative voltage (electrons) is placed on the gate, electrons flowing in the channel are repelled and the channel is effectively narrowed. If enough electrons are put on the gate, flow through the channel ceases entirely: the gate has closed the channel and turned the FET into a switch (i.e., on or off). This is exactly what is needed for the binary elements that comprise a computer.

One additional refinement is possible. The mere presence of electrons on the gate is what turns the current on and off in the channel; there need be no current flow from gate to channel. The electrons on the gate work by their electric field compressing the channel. If a very thin layer of an insulator is placed between the gate and channel, the electric field still exists, but there is no current flow from the gate to the channel. Glass is the insulator most commonly used. Such a device is called an *insulated gate FET* (IGFET) or a metal oxide semiconductor field-effect transistor (MOSFET), and this is the FET version commonly used in computers.

Essentially no current flows through the insulator. The result is that the overall power consumption of the device is very low, and it is possible to build integrated circuits with hundreds of thousands or millions of IGFETs.

For an even more efficient version, the technology almost universally now used in current computers is based on MOSFET technology—with an important twist. There are two kinds of entities that can flow through the channel (the hose) of an FET. The first entity is the familiar electron, carrying a negative charge. However, solid-state electronics can also be designed to use "positive" current carriers. These positive entities are sometimes called "holes" because they correspond to a vacancy due to the absence of an electron. As a homely analogy, consider the gas tank caps on a parking lot full of cars. If one cap is missing and the patrons of the lot are unscrupulous, when a cap is found to be gone, the owner of the car steals one from another car. The absent cap—the hole—can move around the lot just like a current does and can be analyzed in the same way.

In the case of FETs, these two carriers of current in the channels—negative electrons and positive holes—give rise to *p-channel* and *n-channel* MOSFETs. They form a "complementary" pair of devices. With proper design, the pairs can be used together to form binary devices of extraordinarily low energy consumption called a *complementary metal oxide semiconductor* (CMOS). Their extremely low energy consumption is why millions of them can be used in a laptop computer that can run for hours on a small battery.

However, these CMOS transistors are now so small that they are running into fundamental physical limitations, as mentioned in Chapter 1. The insulation between gate and channel may be only tens of atoms thick. Structures this small can allow quantum *tunneling* between gate and channel; that is, leakage, through the insulators, reducing reliability and increasing device heating. It is fundamental physical problems like this one that have caused many in the computer industry to say that the constant past increase in device speed will soon end and that the industry should start to look for alternate computer architectures. One major topic of this book is what an alternate, brain-like computer design might look like and what useful things it might be able to do.

BINARY COMPUTER HARDWARE THAT NEVER SEEMS TO HAVE MADE IT TO PRIME TIME

It is always interesting to consider devices that originally showed great promise as computer components but that were never practically implemented at large scale.

FLUIDIC LOGIC

Fluidic logic takes the hydraulic analogy and makes it real. It uses the motion of a fluid through cleverly designed fluid-filled channels to perform digital operations similar to those performed with electron flow in electronics. The 1960s saw the application of "fluidics" to sophisticated control systems, largely military. It was useful in unusual environments where electronic digital logic would be unreliable, as in systems exposed to high levels of electromagnetic interference, ionizing radiation, or great accelerations, for example, in ballistic missiles.

OPTICAL COMPUTING

For decades, substantial government funding, mostly military, has studied an always promising—but, so far, never delivering—future technology using optical systems to realize the logic functions required to build a computer. Military funding drove the field because an all or largely optical computer could be exceedingly rugged, very small, and immune to electromagnetic interference. It is straightforward to make optical devices talk to electronic devices and vice versa. There have been a few limited practical successes for optical computing but nothing that threatens the dominance of standard electronic technology.

DNA COMPUTING

Since it was first suggested by Leonard Adelman in 1994, there has been interest in computing with biomolecules, most notably DNA. Very simple logical functions have been implemented

and even small programs written using DNA-based computing systems. Through the efforts of cell biologists and biochemists, there is a rich set of enzymes and other biological techniques available to construct, copy, and control DNA molecules. The programming problem can be implemented in DNA using cleverly designed DNA segments that either bind to other segments or don't bind, depending on the structure of the problem. DNA computing can use many different molecules of DNA to try different solutions at the same time. Mundane issues such as reliability, computation time, and programmability remain to be solved.

More recent interest in computers and DNA has centered on the extraordinary potential capacity of molecular storage. Enthusiasts have claimed that all of the 1.8 zettabits of data (a whole lot) in the world in 2012 could be stored in a teaspoon of DNA and that the data would be stored accurately for possibly millions of years with proper storage precautions and well-known error-correcting techniques.[1]

GAME OF LIFE

One of the most unusual ways to realize a universal computer is British mathematician John Conway's "Game of Life." Famous physicist, mathematician, and renaissance intellect, John von Neumann (1903–1957) in the 1940s attempted to find an abstract machine that could build copies of itself, just as animals reproduce themselves. The Game of Life was designed by Conway to realize and simplify von Neumann's ideas. For theorists, the Game of Life is interesting because it can be shown to have the power of a universal computer; that is, anything that can be computed by a digital computer program in general can be computed within Conway's Game of Life. However, the way the Game of Life computes is spectacularly slow and impractical. But it could work.

In the early days of personal computers, simple graphics programs implementing the Game of Life were ubiquitous. An initial pattern was chosen and then marvelous patterns would grow, thrive, collide, or die based on the rules of the game. The Game of Life provided hours of entertainment for paleo-geeks. But the actual rationale for development of the game was more interesting and important.

TINKER TOYS

Two MIT undergraduates, Danny Hillis and Brian Silverman, constructed a tic-tac-toe playing computer from Tinkertoys in 1978. It was built as an assembly of standard Tinkertoy rods and wheels in a 3×3 array of units. The designers commented, "The machine plays tic-tac-toe with the human player giving the first move. It never loses. . . . It could have been built by any six-year old with 500 boxes of Tinker Toys and a PDP-10 [supercomputer]."[2]

NEURONS

Single brain cells—neurons—and ensembles of neurons show some aspects of "on–off" behavior. They are not binary in the sense of digital computer elements, although some people in years past thought they were. There is extensive discussion of the consequences of assuming that nerve cells are binary devices in the sense of a relay or an FET in later chapters.

All these different hardware realizations of digital computers are a good demonstration of the adage, "One computer is all computers." True, of course, only for digital computers.

QUANTUM COMPUTERS

> Now my own suspicion is that the Universe is not only queerer than we suppose, but queerer than we can suppose.
>
> —J. B. S. Haldane, *Possible Worlds and Other Papers* (1927), p. 286

The only fundamentally different digital computer hardware architecture currently on the horizon is quantum computation. Quantum computation is based on what is called *quantum superposition*. Ordinary binary physical devices must be in one state or the other, interpreted in computing machines as 0 or 1. However, remarkably and unreasonably, a quantum mechanical device can be in both states at once; that is, the sum of the two different states. It can be both 0 and 1 at the same time. Such unusual elements are called *qbits*.

Quantum computers can only solve the same set of problems that regular digital computers can. However, they can display huge increases in computer speed for a few very important, though specialized, problems. Quantum computation has been funded for years at a high level by a small set of three-letter US government agencies that want to solve a particular problem critical to their function: breaking codes.

Many current codes and ciphers used by governments, businesses, and financial organizations can be broken if a specific bit pattern—the key—can be determined. If the key is 128 bits long, then one way to break the code is by trying all 128 bit key sequences one after the other. But to do this for a long key takes astronomical amounts of computer time, in this case, on the order of 2^{128} distinct operations, a number somewhat larger than the number of atoms in the universe.

Quantum superposition lets many possible solutions be tried at the same time. Then, the simplest form of key solution—brute force; that is, try them all—can be shown to have a quantum computer solution time proportional to the square root of the number of inputs, very much less than the 2^{128} operations otherwise required. Properly implemented, execution times for some problems can be changed from years to seconds. At present, no one has built a large quantum computer—or at least told us that they have done so. But if "they" can use a quantum computer for code-breaking, encrypted digital data will be readable at will by those using the right hardware.

A small quantum computer is now available from a Canadian company, D-Wave, in British Columbia (http://www.dwavesys.com). This novel device has excited great interest, but it is currently too early to say what its potential may be or even know for sure whether it actually is a quantum computer. Google bought one of these machines, and a team from Google recently reported that it is indeed working as a quantum computer and shows up to 100 million times speedup over a single CPU on problems designed to test the quantum functioning of the device.[3]

GROUPS OF DEVICES WIRED TOGETHER

The second requirement for a computer is that the individual binary elements have to be connected together in groups to work. For many decades, most of the low-level computer wiring

The meaning of the logic function AND is close to its normal language meaning: the function AND is TRUE only if inputs A AND B are both TRUE.

Another basic logic function is

Logic Function Inclusive OR

Input		Function Output
A FALSE	B FALSE	> FALSE
A TRUE	B FALSE	> TRUE
A FALSE	B TRUE	> TRUE
A TRUE	B TRUE	> TRUE

This function is sometimes called Inclusive OR because it is true if A OR B OR both A and B are TRUE. These simple lists of TRUE and FALSE values for different inputs are called *truth tables*. Many common logic function can be described in familiar language terms, for example, AND. However, there are some logic functions that are not so familiar.

In particular, consider NAND or "NOT-AND." The truth table for NAND is

Logic Function NAND

Input		Function Output
A FALSE	B FALSE	> TRUE
A TRUE	B FALSE	> TRUE
A FALSE	B TRUE	> TRUE
A TRUE	B TRUE	> FALSE

The output is FALSE only if A and B are both TRUE.

Another is logic function NOR or "NOT-OR":

Logic Function NOR

Input		Function Output
A FALSE	B FALSE	> TRUE
A TRUE	B FALSE	> FALSE
A FALSE	B TRUE	> FALSE
A TRUE	B TRUE	> FALSE

If there are four possible input patterns of TRUE and FALSE, each pattern associated with a TRUE or FALSE value, then there are 16 possible truth tables (i.e., potential logic functions). Some are familiar and some are not.

One might consider, for example, the following truth table:

Input		Function Output
A FALSE	B FALSE	> FALSE
A TRUE	B FALSE	> FALSE
A FALSE	B TRUE	> FALSE
A TRUE	B TRUE	> FALSE

This function might be held to be a very cynical and depressed function: that is, "everything is false." But variants of it have important applications in computer operations, where it might

soon became a potent and useful analogy for the nervous system. Information flowed over wires, just like activation flowed over nerves. It used electricity and batteries and simple mechanical devices. As technologically obsessed adolescents grew up, so did the phone system. As the telephone system got more complex, it became necessary to find more efficient ways to handle this complexity, suggesting even more useful analogies to brain function beyond mere connections.

THE INVENTION OF THE CENTRAL EXCHANGE

The telephone analogy with brains can be surprisingly rich and productive. Consider a basic problem arising as the network grows. Users must be connected together by wires. Initially, every user can be connected directly to every other user because there are not very many users. But practical problems appear as the number of users grows.

Suppose there is a small local group of six subscribers. Direct connection between all of them requires 15 lines. To add one additional subscriber directly connected to the previous set of six now requires six more lines for a total of 21 lines. One more subscriber adds seven more lines for a new total of 28 lines. The number of added telephone lines grows very rapidly, roughly as the square of the number of subscribers. This scaling relationship is practically and economically unsustainable (see Figure 5.2).

Therefore, early in the history of the telephone network, it became necessary to invent the *central exchange*. Subscriber lines were brought together at this location. A new subscriber simply connected to the exchange. The number of lines now grew directly as the number of subscribers grew and not as the square of the number of subscribers (see Figure 5.3).

However, to make it work, substantial processing complexity had to be added. To connect two subscribers together required a switchboard at the central exchange. But even here, a budding technologist could look at the earliest switchboards and understand them (see Figure 5.4).

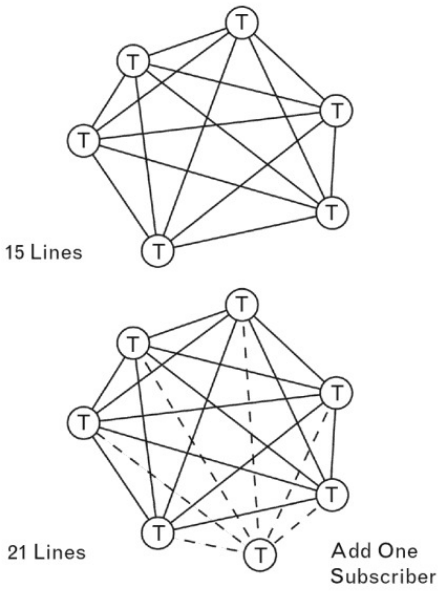

FIGURE 5.2: If every telephone subscriber has to connect to every other subscriber, the number of required connections increases very rapidly, roughly as the square of the number of subscribers. This very rapid increase is not sustainable. Figure by James A. Anderson