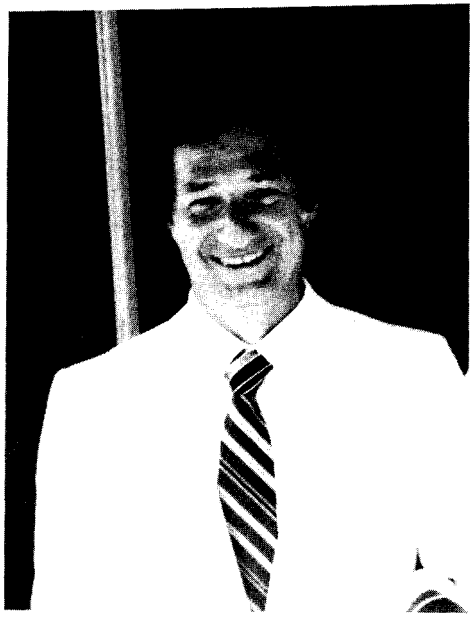
A PHYSICIST'S MODEL OF COMPUTATION

EDWARD FREDKIN
DEPARTMENT OF PHYSICS
BOSTON UNIVERSITY
BOSTON, MA, 02215, USA

This paper is an attempt to make a statement about what a computer is and how it works from the perspective of physics. The single observation that computation can be a reversible process allows for the same kind of insight into computing as was obtained by Carnot's discovery that heat engines could be modelled as reversible processes. It allows us to bring computation into the realm of physics, where the power of physics allows us to ask and answer questions that seemed intractable from the viewpoint of computer science. Strangely enough, this effort makes it clear why computers get cheaper every year.



Physics applied to Computers

Let us imagine ourselves physicists thrust back into the days before Galileo. We observe a heavy object that we hold in our hand. We note that it takes a steady force to hold the object and keep it from falling. We let it go, it falls to the ground and then comes to a stop with a thud. Back then, such observations could have led one to a set of laws of physics that are quite different from what we know today. For example, we might have concluded that a steady force is needed to keep an object still. We might have concluded that the energy or momentum (vis viva) associated with a massive object in motion simply disappears through friction or inelastic collisions.

There are certain principles that helped Galileo and Newton separate rough, common sense conclusions from the underlying laws of physics:

1. There are laws of essentially unlimited and clear applicability. A good example is conservation of momentum. The object is to discover and make use of such laws.
2. It is wise to formulate laws with regard to idealized models that don't suffer from the confusions of the real world.
3. We need to understand and make use of the concept of a closed system, where we can account completely for what is going on.
4. It is often easier to ignore boundary conditions; ie to analyze a dynamic interval as opposed to including the special circumstances about how it starts and finishes.

Today, everything mentioned above seems too obvious for words. Nevertheless, the kind of thinking that is second nature to physicists seems not to have been applied to computation. After a brief foray into how present day computer scientists think about computation, we will discuss computation from a physicist's perspective.

There are many different ways to model computation. The favorite of the computer scientists is the Turing Machine[1], which is a model that simplifies and makes clearer what a non-thinking clerk can accomplish (or cannot accomplish) with an unlimited supply of time, pencil and paper. What Turing did was to abstract the concept of the memory and notes of a mathematician from such messiness as notebooks and a brain, into a one dimensional tape of paper and a person acting as a computer whose mind is in one of only a small number of states. The only allowed operations involve writing one of a small number (e.g. 10) of symbols

onto a square space on the tape (erasing what was previously there), then moving the tape one square to the left or to the right, and finally changing the state from one of a small number of states to another state. The rules governing the operation of a Turing machine consists of a table of quintuples. The five items in each quintuple are:

1. The current state
2. The current symbol in the square in view
3. A new symbol to write in the square; replacing the old symbol
4. A new state
5. An instruction to move the tape one square either to the left or to the right.

The number of different quintuples is at most the product of the number of symbols times the number of states. It is easy to design Universal Turing Machines with a state-symbol product of less than 100. A Universal Turing machine can exactly mimic the behavior of any computer given two things: a long enough tape, and an initial segment of the tape that has a suitable program on it. To do one step of a computation, the computer (a person) finds the quintuple that starts out with items 1 and 2 matching the state (in his mind) and symbol in view on the tape. He then copies the symbol from item 3 onto the square in view (erasing the old symbol), memorizes the state in item 4, moves the tape according to item 5 and then does the same thing again. Amazingly, it is clear that such a simple machine can do exactly what any other computer can do; albeit very slowly.

The Turing machine is clearly a model thought up by a mathematician as opposed to a physicist. Ulam[2] and Von Neumann[3] thought up another computer model that is more like physics in many ways; the Cellular Automata[4]. Computer engineers prefer the Boolean Algebra[5] model of digital logic, mixed with a bit of Automata Theory[6]. System programmers have come up with their own models; a list of all models of computation would be quite a sight.

A physicist, on the other hand, might want a model of computation that is made up of elementary parts and simple, mathematical rules; clearly in concert with the laws of physics. A physicist ought to be able to relate the fundamental things a computer does to units such as Mass, Length and Time. It should be possible to speak about computation and physics in the same breath[7] "...energy, space-time, state, symmetries, translational invariance, particles, time-reversal...".

For a long time, the concept of a physics based model was commonly thought to necessarily rest on the laws of thermodynamics because it was thought that all microscopic acts of computation were necessarily dissipative. However nothing very interesting ever came of such thoughts. What follows is a physically correct exposition of some of the concepts of computation that is based on simple Newtonian mechanics. It is based on the same concepts as the kinetic theory, with the exception that we look at the model in microscopic detail; not statistically. Others have shown that similar models can be based on mechanisms consistent with the laws of quantum mechanics[8]. We believe that it is possible to model and understand microscopic aspects of computation better and more clearly from such a physical model, while all macroscopic aspects remain as what we know from other models of computation.

Ordinary Computers Like the Mac or Cray

Real computers have the distinction, amongst machines or systems, of corresponding most exactly to their abstract models. Most systems, say internal combustion engines that convert heat energy into mechanical energy, only crudely resemble their abstract models. Other systems match their models very well. A gearbox, whose output shaft turns at a rational multiple of the input shaft's RPM is quite good.

Ordinary computers are made up of millions or billions of parts, each of which might do millions or billions of operations per second, and each part does the exact thing expected of it every time. There is no approximation in the result. In contemporary computers, this is all due to the extreme quantization of signal and the existence of the appropriate transfer function. In other words, although noise is ever present at every microscopic action within a computer, the noise can nowheres ever exceed a very small threshold because it is converted to heat by every logic gate, while memories have sufficient hysteresis or signal thresholds to eliminate the possibility of being affected by noise. Of course this is all relative; today's technologies allow us to construct computers with essentially any probability p of not making an error, $0 \leq p < 1$, given that certain environmental conditions are maintained. Economics causes us to accept computer circuits where 10^7 gates operating 10^8 times per second will collectively drop a bit no more than every 10^8 seconds provided they are protected from heat, cold, lightning, cosmic rays, and other such trauma. That is about one error

every 10^{23} potential operations. If you wanted one error for every 10^{40} operations, the computer might be somewhat more expensive.

Thinking About Computation in a Closed System

It is very easy to get confused about what computation is unless computation is first modelled as a closed system. This means that we will ignore all input and output; essentially leaving it out of the model for the time being. Imagine a computer that consists of nothing but a memory (RAM) and a processor. Somehow, the program and the data are already in memory. The computer runs until the computation is finished, at which point the results of the computation are in memory. We will first make a model that describes that part of computation, and then we will show how to add input and output to the model. This is like considering the falling object between the time it is let go until some time before it hits the ground. We want to be able to answer questions about how much of the resources of nature (such as matter and energy or space and time) are needed to do such a computation. With regard to energy, there are two parts to the answer: how much energy is needed to set the computer in motion, and how much energy is dissipated in doing the computation.

Energy Requirements for Computation

If a most efficient supercomputer works all day to complete a weather simulation problem, what is the minimum amount of energy that must be dissipated according to the laws of physics? The answer is actually very simple to calculate, since it is unrelated to the amount of computation. The answer is always equal to zero. It is much like looking at a machine, such as a heat engine, and asking how much energy must be dissipated in converting heat into mechanical energy. The answer is, as so beautifully elucidated by Carnot, always equal to zero. This is because Carnot leapt over all other conditions and constraints by observing that a reversible process did not dissipate energy. Strangely enough, the necessary dissipation in a computer is only related to the amount of output data printed or displayed at the end of the computation, and unrelated to the amount of intermediate data generated during the computation. Thus the amount of energy that must be dissipated by a computation that answers the question "Is 3 an odd number?" is exactly the same as must be dissipated by a computation that answers the question "Is the 1,000,000,000th digit of

Pi an odd number?"; zero plus the dissipation required to display one bit, Log(2)kT, in both cases.

The Model

In order to model computation in a physically correct way, we will follow the lead of computer engineering by identifying the atomic parts of computers, and show that compositions of these parts can implement essentially any kind of computer. The difference will be that we will use parts that are physically correct rather than just logically correct.

When we say "computer" we mean only the central processing unit and its memories. In the abstract, ordinary commercial computers are built out of logic gates and wires. In real life, transistors, conductors, capacitors, insulators, power supplies, packaging and cooling constitute the essential kinds of elements found in computers. The normal abstract engineering model of a computer is a diagram consisting entirely of gates and wires along with a few other things that could be replaced by functionally similar things made of gates and wires; e.g. a magnetic hard disk could be replaced by a functionally similar device made of gates and wires. The rules are simple, each gate has inputs and an output. Each output can be connected to a few inputs.

In practical computer engineering, many different kinds of gates are used, yet the 2 input NAND gate is logically sufficient. All other gates can be synthesized out of combinational circuits of just one type of universal gate such as the NAND gate.

We will now develop a different abstract model that is based entirely on physically simple computational atoms and processes. At first it seems that we must account for three kinds of processes: computation, memory and communication. We will first explain why, in a formal, physically correct model of computation, memory and communication are the same process. Second, we will look at the atomic parts, which turn out to be two kinds of particles. For space, we will use 2 + 1 dimensional space-time. Two spatial dimensions are used mainly because of its correspondence to drawings on paper of computer circuits.

Memory and Communication

If you write a letter and send it to a friend, that's communication. If you keep the letter, that's memory. A clever physicist can always change one to the other by means of a coordinate transformation:

"We define a fundamental act of communication as follows:

$R_{x', y', z', t'} <= S_{x, y, z, t}$; we Receive, at x', y', z', t' information Sent earlier at x, y, z, t.

We define a fundamental act of memory as follows:

$R_{x', y', z', t'} <= S_{x, y, z, t}$; we Read, at x', y', z', t' information Stored earlier at x, y, z, t. Normally, x', y', z'=x, y, z."

Thus the distinction between the words "memory" and "communication" is left to describe our intent; physics can't tell the difference.

In kinetic theory, we can deduce properties of ensembles of particles as the average results of the consequences of individual particles following simple dynamical laws. However, assume we know nothing more than the following rules:

1. A simple gas that is compressed isothermally gives off heat and increases in pressure.
2. A simple gas that is compressed adiabatically increases in temperature and pressure.
3. Processes 1 and 2 are reversible, i.e. a simple gas that is expanded isothermally absorbs heat and decreases in pressure.
4. A perpetual motion machine is not possible.

Assumptions 1, 2, 3 and 4 are sufficient to show that a Carnot[9] engine is the most efficient possible way to convert heat energy into mechanical energy. The reason is simple; a Carnot engine is reversible, therefore if a Carnot engine can convert an amount of heat energy to mechanical energy, it can then operate in reverse to convert that mechanical energy back into the same amount of heat energy. If there were a more efficient engine needing less heat, then the Carnot engine operating in reverse could convert a portion of the mechanical energy back into all of the heat required by the more efficient engine. This would leave some free, surplus mechanical energy, violating 4. above. This argument makes no use of the kinetic theory; as far as Carnot is concerned his results are correct even if the working fluid is continuous rather than a gas made up of molecules. This is because everything that is shown true about the Carnot engine derives from nothing more than reversibility and conservation of energy.

The discovery that Computers can also be modelled as reversible processes was made independently by Bennett[10] for Turing Machines, the author[11] for computer logic and ordinary computers, and by Toffoli[12] for Cellular Automata. By assuming reversibility at a microscopic level and by simplifying kinetic theory into the Billiard Ball Model[13] (BBM) of

computation, we can understand the exact evolution of each and every particle. Strangely enough, a result of Turing, called "The Halting Problem" shows that it is not possible, in general, to calculate by any analytic method, any good statistical measures about the evolution of a given initial condition. This is because the BBM can be a universal computer like a Turing Machine.

The Billiard Ball Model

We assume that there are two kinds of particles: light particles and heavy particles. All particles are perfect spheres with a radius of one. These particle are imagined to be just like perfect billiard balls with the exception that they are only engaged in translational motion. We assume that they are all of the the same size, and we ignore friction or other sources of errors in the particles. When two particles collide, they simply bounce elastically, conserving kinetic energy and momentum with no losses or other forms of energy dissipation in the process. The most unusual aspect of the model has to do with the initial conditions. While the BBM assumes absolutely no errors or uncertainties, that assumption does not affect the main conclusions. Quantum Mechanics, rather than hampering things because of uncertainty, can come to our rescue by naturally constraining systems to a small number of states despite small disturbing effects.

Initial Conditions

Imagine a Cartesian space-time lattice. The initial condition requires that at time 0 the coordinates of every particle are integers. At time 0 the absolute value of the X component of the velocity of every light particle is ± 1 (± 1 unit of x per unit of time) and ± 1 for the Y component of velocity. The initial conditions and the assumption that the particles move without error or friction means that whenever time has an integer value, every particle has integer coordinates. At the moment when two particles collide, they both have integer coordinates and the line connecting their centers is parallel to either the X axis or the Y axis. If the line is parallel to the X axis, then each particle changes the sign of the X component of its velocity. Particles only move in the four diagonal directions. Collisions take place every so often as the system evolves. It is like a further abstraction of the perfect gas model, where our interest is focussed on the exact evolution of every particle, as opposed to the statistical evolution of an ensemble.

In order to keep things bounded and to define paths through space that correspond to wires, we introduce heavy particles that are identical to light particles except that their velocity is zero and their mass is infinite. The rules for particle interaction cause any heavy particle to remain stationary. Heavy particles act like reflectors or walls. Such particles are introduced as a convenience in order to allow the model to correspond closely to circuit diagrams drawn on paper. Heavy particles are shown as cross-hatched blocks.

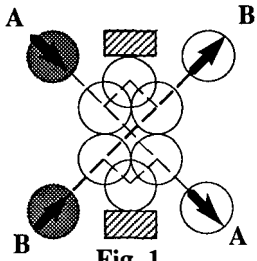


Fig. 1.
The Crossover Gate

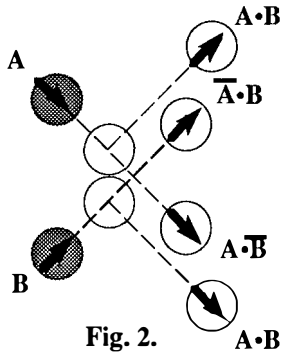


Fig. 2.
The Interaction Gate. A 2 input, 4 output, universal, reversible logic gate.

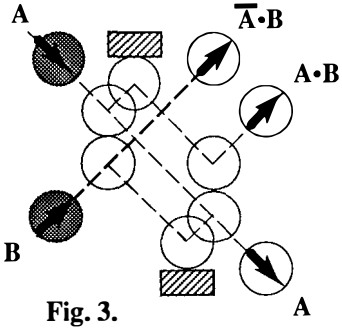


Fig. 3.
The Feynman Gate. A 2 input, 3 output, reversible, universal logic gate.

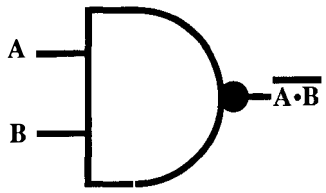


Fig. 4
The 2 input NAND gate, an irreversible, universal logic gate; used in real computers.

All ordinary computers are built out of a large number (millions) of gates like the 2 input NAND gate shown in Fig. 4. NAND gates are considered to be universal since combinational circuits of such gates can perform any desired logic function, including memory. BBM circuits such as the Interaction gate Fig. 2 or the Feynman gate Fig. 3 are similarly universal while also being reversible.

The Hardware is Space!

A bit of memory in the BBM is a point in space-time that may or may not have a particle present. By convention, if the particle is present, we say that the state is a 1; if the particle is absent, the state is a 0. Given a volume of space, v, (v=the number of places that a particle might be) and a number of particles, n, the amount of information that can be represented is equal to the Log_2 of the binomial number B(v, n) or $Log_2(v!/(n!(v-n)!))$. Conversely, the minimum volume required to do a computation must certainly be greater than what is required to represent the maximum amount of information, that needs to be present at one time. The most efficient number of particles (in order to minimize volume) is n=v/2. Physics mandates that the number of particles be fixed, unlike ordinary computers where the number of ones in memory can vary anywhere from zero to the total number of bits in memory. Since a particle in the BBM must be conserved, the number of ones is constant. Even if information is represented by spin (up or down) then conservation of angular momentum means the number of ones cannot change. This is only slightly less efficient than using bits, since the Limit(Log(B(v, v/2))/Log($2^{\wedge}v$))=1 as v—>∞. E.g. a volume of 1000 with five hundred particles represents the same information as 995 bits.

The minimum time required to do the computation is related to the largest total distance that must be traversed from an initial bit of information to a final bit of information. This path may include many paths connecting various intermediate bits. Each path is the sum of a set of subpaths, where a subpath takes a bit from one gate to the next. In the BBM a gate is simply a point in space where an informational interaction might take place. It is obvious that the dimensionality of the space will greatly affect this time. Ie, 3D is better than 2D which is much better than 1D.

What is known about the BBM is that it is possible to use it to implement fairly conventional computer models that have the distinction that the basic components are microscopically reversible. This means that the macroscopic operation of the computer is also reversible. This fact allows us to address the same question posed by Carnot about heat engines. "What is required for a computer to be maximally efficient?" The answer is that if the computer is built out of microscopically reversible components, then it can be perfectly efficient. How much

energy does a perfectly efficient computer have to dissipate in order to compute something? The answer is that the computer does not need to dissipate any energy. If the user wants a copy of the answer, then making the copy must cost about Log(2) kT for each bit of information in the copy.

To be precise, we will imagine that we have a reversible floppy disk containing three sections: a computer program, the data, and a blank section for the results. We want to put the floppy into our reversible computer and have it run the program on the data, and then write the answer onto the floppy. The first step is that the computer swaps the contents of a Block of its Memory, BoM, for the contents of the floppy; the program, the data and a blank area that will receive the results of the computation. This is a reversible and non-dissipative process. Then the computer gets to work and runs the program until it has the answer in its memory. The computer then makes a copy of the result and writes it onto the floppy. This is the only non-reversible process and it costs Log(2)kT for each bit of the result. The computer then operates in reverse to undo anything not already undone, converting the result in its memory (along with other intermediate data) back into the original program and data. The computer then swaps the program and data in its memory for the BoM on the floppy. At this point the computer is in exactly the same state as it was before running the program, but the floppy has been changed in that the results have been added to the original data on the floppy. Thus the amount of energy that must be dissipated is exactly proportionate to the number of bits in the result, and not related to the amount of computational *work* that the computer had to do.

From the point of view of physics, the resources that corresponds to an operating computer is simply a volume of space, and a number of particles in motion. (It sounds just like the real world.) The particles communicate by moving, remember by existing and compute by interacting.

The Price of Computing

It may be hard to believe, but the picture we have painted of the computer from the perspective of physics allows us to understand the evolution of the price of computing. Many things made have costs that follow the so-called learning curve. The price decreases as the total number of items made increases. While the learning curve affects the

cost to manufacture computer components, it is not the major factor in the continuing decline of the cost of computation.

The steady decline in cost of the Pure Informational Parts of computers (lets call them PIPs) is related to the physics of computation. By PIPs we mean those parts that deal only with information; this includes memory chips, processor chips, all integrated circuits that just do digital logic, disks, etc. It does not include keyboards, printers, displays, etc. If one poses the question "What units of physics are necessarily related to the PIPs?", we get a startling answer; different than for any other things that are made by or used by man. The answer is that no particular amount of mass or length or time are necessarily related to the PIPs. If you consider a chair in the same light, it is obvious that the unit L (length) is necessarily related to chairs. We cannot make a useful chair that is 10 times smaller than normal. If you consider a paperweight, it cannot be made to weigh 1/100 of its weight and still perform its function. The motor in a car cannot be designed to deliver only 1/100 of a horsepower and still propel a car satisfactorily. However, every PIP can be reduced in size, reduced in mass, made to use less power, made to use less time (work faster), because it deals only with information. Information is not necessarily related with any minimum amount of any physical quantity; not length, not mass, not time, not power! What this means is that it is simply up to the engineers to use their imagination to continually find ways to use less and less in order to compute more and more.

While we can see limits associated with any particular technology, there is no reason to think that we will be unable to come up with better technologies as the need arises. All we need are states (particles) and space-time. If nature is finite (if there is a fundamental unit of length) then that will be a barrier, but we are still far from wherever that limit might be. As to the particles, it seems clear that we will be able to represent bits with single particles (such as an electron or a photon) or by simple 2 state systems such as spin. Today's computers are still far from such limits. As to space, today's computational structures (integrated circuits) are not only large by atomic standards, but they are basically all 2-D structures! As Feynman said long ago "There's plenty of room at the bottom."[14]

We don't want to imply that miniaturization can go on forever, but the current pace (about a factor of 2 every 2 years) can surely continue for another 100 years (a factor of 2^{50} or 10^{15}) which will bring us to a liter

of computer containing 10^{23} bits and able to out-compute the combined power of all the millions of computers in the world today. All this for $1000 in 1991 dollars.

---

1 A. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Society Series 2, 42 (1936) 230-265

2 S. Ulam, Random Processes and Transformations, Proceedings of the International Congress on Mathematics. (held in 1950) 2 (1952), 264-275.

3 J. von Neuman, Theory of Self-Reproducing Automata (edited and completed by Arthur Burks), University of Illinois Press (1966)

4 T. Toffoli and N. Margolus, Invertible Cellular Automata: A Review, Physica D 45 (1990) 229-253

5 G. Boole, An Investigation into the Laws of Thought, 1854

6 M. Minsky, Computation, Finite and Infinite Machines (Prentice Hall, Englewood Cliffs, NJ, 1967)

7 E. Fredkin, Digital Mechanics, Physica D 45 (1990) 254-270

8 R. Feynman, Quantum-Mechanical Computers, Foundations of Physics 16 (1986) 507-531

9 S. Carnot (Nicolas-Leonard-) Reflections on the Motive Power of Fire (1824)

10 C. Bennett, Logical reversibility of computation, IBM J. of Research and Development 6 (1973)

11 E. Fredkin, Conservative Logic, informal memo, Caltech, 1975

12 T. Toffoli, Cellular Automata Mechanics, Technical Report 208, Computer and Communication Sciences Department, University of Michigan (1977).

13 E. Fredkin and T. Toffoli, Conservative Logic, International Journal of Theoretical Physics 21 (1982) 219-253

14 R. Feynman, There's Plenty of Room at the Bottom, reprinted in Miniaturization, H. Gilbert ed., New York: Reinhold (1961)