Classical reversible computation with zero Lyapunov exponent

Warren D. Smith
wds@research.NJ.NEC.COM

date = February 25, 1999

Abstract — **In 1982, Edward Fredkin invented a way to build a time-reversible universal computer using frictionless "billiard balls" rolling on a table top, and in which ball "bounces" corresponded to Boolean logical operations. This model made it seem possible to perform unboundedly long computations while consuming zero power. But, achieving that feat seemed to require perfect precision and perfectly precise billiard balls making perfectly precise bounces forever. Actually, we would expect an exponential buildup of imprecision with time t. If uncorrected, this error buildup would destroy the computer; while on the other hand, continually correcting the errors seems to require power dissipation $> 1.34 k_B T$ per bounce. We show how to make a variant of Fredkin's computer with *zero* Lyapunov exponent, and in which we expect errors to grow at worst proportionally to $t^{3/2}$. This would seem to allow us to consume $k_B T$ per every P bounces, where P can be made to grow as a power law in the mass, precision, velocity, strength and rigidity of the billiard balls. E.g. with perfectly rigid billiard balls of constant size moving at constant velocity, P grows at least proportionally to the sixth root of their masses. But we still need to assume zero friction and we still are subject to the usual limitations of reversible circuitry.**

The new scheme's "zero-power" advantage seems to come at the cost of requiring more hardware than Fredkin's original scheme: emulating a (reversible with bounded tape) Turing machine for N steps takes hardware growing proportionally to N in my scheme, but not growing with N in Fredkin's scheme.

Considering that and some other people's results suggests the following tentative conjectural *"no free lunch" principle:* "Zero power" computing can be done, but it comes at a price. That price is: performing N computational steps requires a factor of $\geq N^{\beta}$ extra hardware×delay product, if the energy consumption is reduced from order $k_B T N$ to order $k_B T N^{1-\beta}$ (for any $0 \leq \beta \leq 1$).

Keywords — Reversible computation, ultimate limits on computation, Landauer bit erasure principle, zero Lyapunov exponent, billiard ball model, no free lunch.

1 Introduction

The construction in §3 this paper addresses the following key question about the fundamental physical limits on computation: "Does classical mechanics, in principle, allow unboundedly large arbitrary computations to be performed with only bounded energy expenditure?"

The answer will be seen to be more affirmative than previously thought. However, there still *seem* (conjecturally[1]) to be limits on our ability to get something for nothing. At the end, we tentatively formulate a "no free lunch principle" which seems compatible with everybody's analyses so far.

1.1 *Why erasing a bit costs at least $k_B T \ln 2$ energy*

Suppose you have a computer (some physical system) and it contains a bit (2-state system). You "erase" the bit, thus reducing the number of states the computer could have been in, by a factor of 2.

This reduces the entropy of the computer by an additive amount $k_B \ln 2$, where k_B is Boltzmann's constant. (Thermodynamic entropy of a system that could have been in W states, is, for the present purposes, $k_B \ln W$. The scaling factor relating thermodynamic and information theoretic entropy is $k_B \ln 2$ per bit.)

If we assume the "second law of thermodynamics" that the entropy of the entire universe cannot decrease, then the bit erasure must have raised the entropy of the rest of the universe by at least $k_B \ln 2$. If the rest of the universe happened to be a "heat bath" (e.g. large volume containing an ideal gas; or any other large system) at temperature T, then this would necessarily have required raising the energy of that bath by an amount $k_B T \ln 2$.

This is because the definition of "temperature" of a system is $T = \partial E / \partial S$ where S is its entropy and E its energy. **Remark.** Attempts to refrigerate your computer to decrease the cost of erasing a bit won't work because the best possible "Carnot efficiency" of a heat pump is T_1/T_2. In other words, if a bit erasure heats up the refrigerated heat bath by adding energy $k_B T_1 \ln 2$ to it, then the refrigerator would have to get rid of that heat energy by exporting it

[1] The italicized verb "seem" will be used in this paper to denote a conjecture.

to the outside world at temperature $T_2 > T_1$. Even with a perfect Carnot refrigerator, that will cost

$$\frac{T_2}{T_1} k_B T_1 \ln 2 = k_B T_2 \ln 2. \tag{1}$$

Remark. m-state "bits" would require $k_B T \ln m$ per erasure.

1.2 Reversible computers – the dream

The bit-erasure principle was popularized by R.Landauer but may have been known to earlier authors. This led those investigating fundamental physical limits on power consumption to ask:

1. **Logical question:** Can we avoid erasing bits entirely in the course of arbitrary computation?
2. **Physical/engineering question:** And if so, will the laws of physics really allow us to compute without consuming power?

It turns out [3] [5] [11] that the answer to the logical question is "yes," although in many computations it *seems* to be necessary to store an unboundedly growing number of non-erasable "garbage bits" forever, increasing the memory needs (and perhaps also the runtime, in physically realistic models of computing) of the computer by an unboundedly large factor.

I won't survey the considerable work on the engineering question now; I'll just describe the two papers [9] [13] most relevant for us.

Fredkin and Toffoli [9] considered reversible circuits made of "reversible logic gates." The "Fredkin gate," a 3-input, 3-output reversible[2] logic gate (figure 4, top), proves sufficient to build any circuit you can build with the usual non-reversible logic gates such as AND, OR, and NOT.

Theorem 1 (Completeness of the Fredkin gate) *Any 1-1 function mapping the set of N-bit binary words $\{0,1\}^N$ into itself may be implemented with a feedforward reversible circuit with N inputs and N outputs, plus C extra inputs and C extra outputs which are always logical-0, where $0 \le C \le 2N$ and provided we agree on the convention that each bit is carried on* two *wires such that logical-0 is $(1,0)$ and logical-1 is $(0,1)$.*

Proof sketch. Unfortunately the proofs of the best known results in this area – due to Fredkin, Toffoli, and their students, especially Bill Silver, in unpublished work around 1980 – are not in the published versions of [16] and [9]. The 37-page tech report version of [16] does contain a proof that the 3-input, 3-output "Toffoli gate" in figure 1 suffices to implement any 1-1 function mapping the set of N-bit binary words $\{0,1\}^N$ into itself using the usual convention (1 wire per bit) and provided $\le N-3$ extra inputs and outputs are available (which each contain agreed upon constant bits) as "temporary storage." The proof of that involves

1. We can build a circuit which negates the Nth bit in a word if that word's first $N-1$ bits are all 1s. ([16]'s figure 5.3 and induction.)
2. Using N of these plus some bit-negators (using a "gray code path" idea) we can build an "atomic permuter" circuit, meeting our design constraints, which will perform an exchange of any two desired N-bit words from among the 2^N possible such words. The other 2^N-2 words are transmitted unaltered by this circuit. ([16]'s theorem 5.1.)
3. A sequence of $\le 2^N$ atomic permuters suffices to get any 1-1 function. ([16]'s theorem 5.3.)

The total number of Toffoli gates required to build an arbitrary 1-1 function on N bit words in this way, is $O(N2^N)$.

In view of the fact that the number of possible such functions is $2^N!$, and that each gate could be connected in $O(N^3)$ ways to the previous circuit layer (assuming all layers of the circuit are $O(N)$ wires wide), at least order $\log(2^N!)/\log(N^3)$ gates are required, i.e. order $N2^N/\log N$. Hence Toffoli's construction is optimal in the worst case (for "$N^{O(1)}$ wires wide" circuits) up to a $O(\log N)$ factor. (Even with no width restriction, it is optimal up to an $O(N)$ factor.)

This contrasts with the well known fact that in conventional (irreversible) logic, one may use 2-input NAND gates and feedforward-only wiring to implement any boolean function with N inputs and 1 output. (Proof sketch: we can make a NOT gate and then make AND and OR gates. Then we can make multi-input AND and OR gates. Then using these one can build a ROM [read-only memory] in which, of course, we may encode the "truth table" of any boolean function.) There are $2^{(2^N)}$ such functions, so in polynomially wide implementations, at least $2^N/\log N$ gates are required, which compares with the upper bound of $O(N2^N)$ from the usual ROM construction. Thus the usual ROM construction has optimal gate count in the worst case up to a factor of $O(\log N)$ for circuits $N^{O(1)}$ wires wide. For circuits of unrestricted width, it is optimal up to a factor of $O(N)$.

Finally, to get our theorem, we need to describe how to use Fredkin gates with our 1-bit on 2-wires convention to build a Toffoli gate. For that, see figure 2. (The constant '0' bit wires can be reused at all levels of any circuit built from these Toffoli gates.) □

Then Fredkin invented a beautifully simple *physical* 2D realization of the Fredkin gate, consuming no power, using "ballistic billiard balls" (see §2).

The standard point of view, to paraphrase Toffoli [17], then became: "The dissipation in a computer is

$$\ge (QG + I \ln 2) k_B T \tag{2}$$

where I is the number of input/output bits, G is the number of elementary logic gate operations in the computation, and Q is a 'technological factor' which, as far as anybody knows, may be made as small as desired. Thus the dissipation *per*

[2] That is, it works the same forwards and backwards! Also it conserves 1's and 0's.

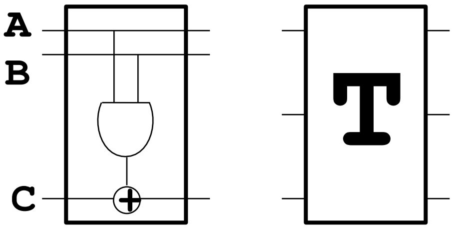

Figure 1: Toffoli "AND/NAND" 3-input 3-output reversible gate. A and B are sent through unaltered, and C is negated if $A \,\mathrm{AND}\, B$. (On right: Toffoli gate symbol.)

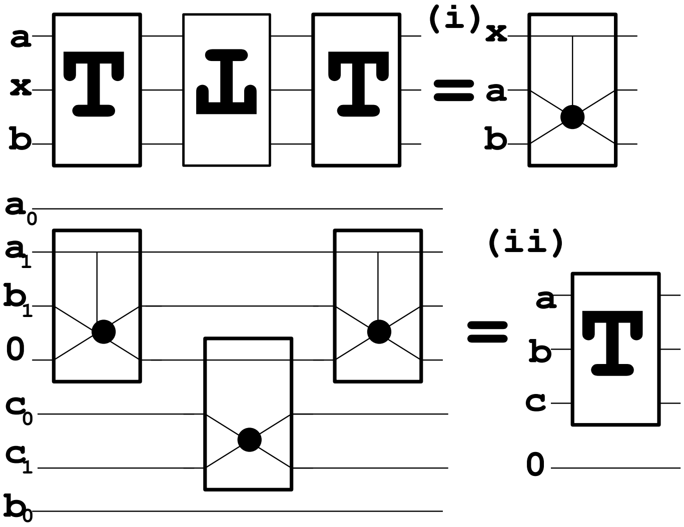

Figure 2: (i) Fredkin gate made out of 3 Toffoli gates. (ii) Toffoli gate (now with "1 bit on 2 wires" logic convention) made of 3 Fredkin gates. Note the constant-bit wire, carrying an '0' used for "temporary storage" of ab.

gate operation is

$$(Q + \frac{I \ln 2}{G}) k_B T \tag{3}$$

which tends to zero for computers running long computations without outputting intermediate results ($G \to \infty$), and in the limit of perfect technology ($Q \to 0+$). This does not mean a nonzero bound is ruled out, but for the moment, the road is clear."

Still, the fact is (as we will see) that, with the Fredkin implementation, the "technological factor" Q apparently *can't* be made arbitrarily small! I then attempted to prove the impossibility of making Q arbitrarily small, but ended up proving possibility.

It might be possible to get similar results in a different (and perhaps better) manner, by using [13]. Margolus [13] showed how to embed the billiard ball model inside a cellular automaton, demonstrating that simple "reversible 2D cellular automata" were capable of universal computation. Also, Likharev [12] has indicated how to build electrical Fredkin or Toffoli gates using Josephson junction superconducting logic (the fundamental component he needed, the "parametric quantron," has been made). In the limit of ultra-slow computational speed, Likharev's gates consume arbitrarily small amounts of energy per operation.

Plan: In §2 we'll review Fredkin's billiard ball model and extend it to incorporate imprecision and thermal noise. We'll make it plausible that dissipation of order $k_B T$ per bounce would be both necessary and sufficient to allow it to run in the presence of thermal noise at a fixed temperature T, assuming the balls were manufactured with precision better than some threshold. In §3 we'll see how to reduce the "Lyapunov exponent" to zero and show how to reduce the power dissipation, in the presence of thermal noise and imprecision, to $k_B T$ per P bounces, where P may be made arbitrarily large by making the billiard balls more ideal (P exhibits a *power law* dependence on parameters describing how ideal the billiard balls are). We'll conclude by formulating a tentative "no free lunch principle" indicating that "zero power" computing, while possible, seems to come with a substantial price tag.

2 Fredkin's Billiard ball model

The Fredkin "Billiard ball model" is illustrated in figure 3. Perfectly round rigid frictionless billiard balls of unit diameter roll on a table with one of the four allowed velocity vectors $(\pm 1, \pm 1)$ so that their centers always lie at integer coordinates at integer times. Collisions between the balls or between balls and fixed horizontal or vertical "mirrors" preserve these properties.

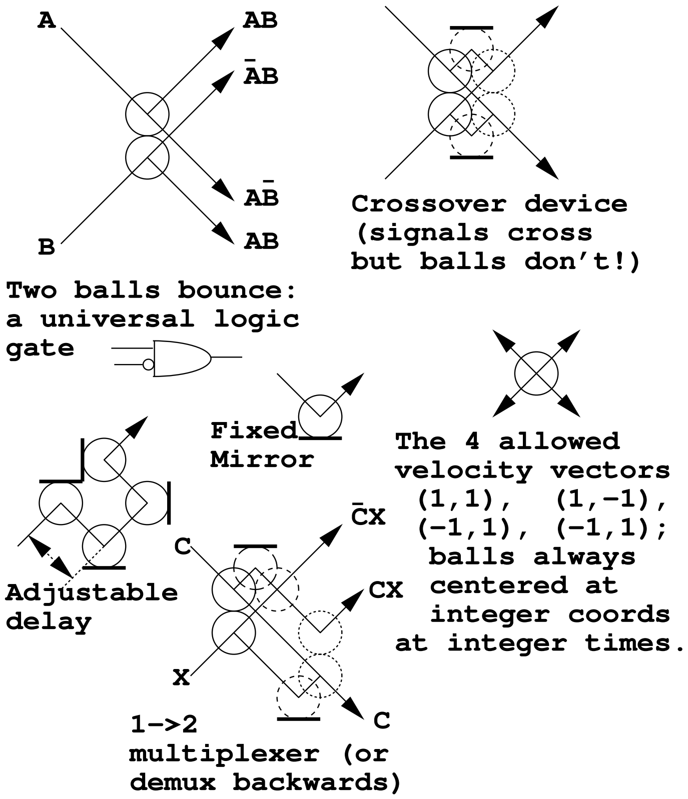

Figure 3: Fredkin Billiard ball model.

Interactions between balls and/or mirrors can implement Boolean functions, where "ball present" is regarded as logical 1 and "ball absent" as 0. Figure 4 shows how to implement the reversible Fredkin "controlled crossover" logical gate using balls and mirrors.

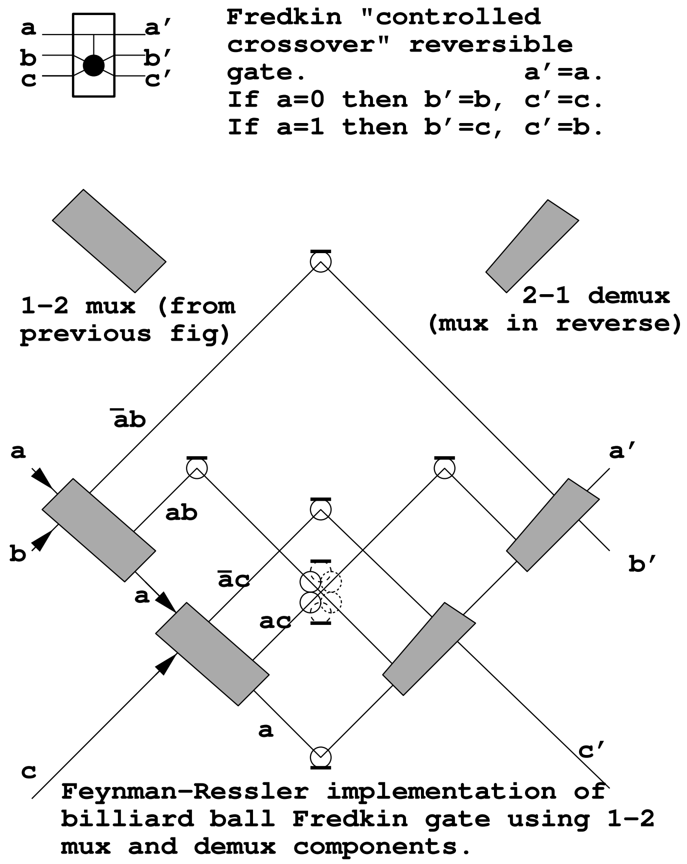

Figure 4: Fredkin gate via balls. Note that the logic gates shown, both here and in the preceding figure, involve delays independent of the bit values.

As we saw in theorem 1, the Fredkin gate also is complete for reversible boolean functions. It is also complete for *all* boolean functions if hardwired 1's and 0's are available. This may be seen by making an AND gate, and a NOT gate and $1 \to 2$ ball copying device each out of one Fredkin gate, for example[3]. Actually, bit copying is not always necessary to get fanout$>$ 1 because in many cases we can "reuse" a bit, such as the a-bit on a Fredkin gate (which passes through the gate unaltered) at the next gate.

Actually, just the AND gate with one inverted input shown at the top left of figure 3, which corresponds to a single ball-bounce, is complete if we have hardwired 1's and 0's available[4]. But in billiard ball logic, it isn't as trivial as in normal digital logic to get "hardwired 1s and 0s." A hardwired 1 input to a gate requires a precisely timed ball to enter that input; and if the gate is intended to be reused N times, we need a stream of N precisely timed balls.

The fact that crossover and arbitrary delay devices are available in billiard ball logic (figure 3) then allows us to "wire" any circuit we want. In particular we could build a finite size circuit to implement 1 step of a Turing Machine (or Von Neumann machine) [14] universal computer with any bounded number of bits of memory, and then by feeding the outputs of this circuit back to its inputs, we would get a circuit which kept going forever (or until it ran out of memory) performing 1 Turing machine step per cycle. No power supply (beyond the energy needed to initialize the machine, and perhaps to measure its "output") is needed. Thus

Theorem 2 (Turing machine simulations with billiard balls) *A Fredkin billiard ball circuit involving* $O(L)$ *balls and mirrors,* plus $O(LN)$ more *balls for use in making hardwired '1' inputs, suffices to emulate any particular Turing machine with tape length* L*, for* N *timesteps. If the Turing machine is* reversible*, then* $O(L)$ *balls and mirrors suffice, independent of* N*.*

Reversible Turing machines can simulate any Turing machine [3] but this simulation (so far as is currently known [11]) may require amplification of memory space requirements by unboundedly large factors.

Indeed, the machine could be run until it produced its answer. One would then make copies of the bits of the answer and then reflect all the balls backward so that the computation would then "undo" itself. The result would be the initial state of the computer back again – including all the streams of hardwired 1's – ready to run another computation. The only power consumed would correspond to the copying of the "output" bits and the initial settings of the "input" bits – the other balls would all be re-usable for an arbitrary number of additional computations each with different inputs and outputs.

2.1 Non-idealities

Unfortunately in real life there may be rolling or sliding friction, imperfect rigidity of the balls and mirrors, and imprecision due both to imperfect sphericity and to thermal noise and the $\Delta p \Delta x > \hbar/2$ uncertainty principle.

Rolling (or sliding, if the "balls" were really upright circular cylindrical "hockey pucks") friction is avoidable by not having a tabletop – put the balls in vacuum in a gravity free environment, and they will stay in a plane automatically. Also if frictional drag forces are proportional to velocity v for small v, then frictional energy losses per bounce are in principle reducible by a factor of X (for any X) by making the balls move X times more slowly.

Imperfect rigidity of the balls and mirrors is going to lead to ordered translational motion of the balls being converted during collisions into elastic vibrational "ringing" modes inside the balls and mirrors, which later could die out and be converted into heat. This is in principle avoidable by making the balls sufficiently rigid, low mass, low radius, and low speed that the energy available during an impact, is not large enough to excite the lowest-energy vibrational mode, with energy hf for a mode of frequency f. (This is actually the idea behind some quantum toy models [1]

[3] If $c = 0$ then $c' = (a \,\mathrm{AND}\, b) = ab$. If $b = 1$, $c = 0$ then $b' = (\mathrm{NOT}\, a) = \overline{a}$, $c' = a' = a$.

[4] Same proof sketch as the proof of the completeness of NAND, plus we need to build a $1 \to 2$ copying device.

[2] of reversible computation – we eliminate all states from the state space except for the ones we want, eliminating any possibility of entropy leakage.) That would be achievable by making the "balls" actually be molecules or even atoms. For example, ^{4}Helium atoms in the ground state have perfect spherical symmetry (at least, if nuclear polarization "hyperfine" effects are ignored) and minimum excitation frequency $f = 47922361\,\mathrm{GHz}$. Thus any collision between two ^{4}He atoms, both traveling at speed $< v = 21.8\mathrm{km/second}$ will be lossless. (This value of v was found by solving $m_{\mathrm{He}} v^2 = hf$ where h is Planck's constant and m_{He} is the mass of a ^{4}He atom.) The use of atoms would also eliminate "manufacturing errors" (e.g. balls of differing sizes)[5].

That still leaves us with the problem of imprecision, and *this* problem seems impossible to avoid! Both Feynman [8] and Zurek [18] have discussed this. The problem is that tiny errors in the momentum and positions of the billiard balls and mirrors are magnified exponentially by a sequence of collisions. Zurek's analysis of this is particularly elegant (although he was sloppy with his constants, thus getting a too-weak result) – observe that an infinitesimal error in angle $\Delta\theta$ (away from a perfect horizontal or vertical collision) is amplified each collision by a factor $1 + \ell/r$, where ℓ is the free path to the next collision and r is the ball radius. The volume V_n of phase space describing the uncertainties in the positions of the balls after n collisions will thus be

$$V_n = V_0 \prod_{j=1}^{n} (1 + \frac{\ell_j}{r}) \tag{4}$$

so that the increase in entropy (k_B times ln of phase space volume) per collision is $k_B \ln(1 + \ell/r) \geq k_B \ln(1 + 2\sqrt{2})$ (since $\ell \geq 2\sqrt{2}r$). Hence, each collision effectively increases entropy by an additive amount $k_B \ln(1 + 2\sqrt{2})$, and in this sense the dynamics are effectively *not* reversible. Any correction scheme for getting rid of this entropy increase by pumping it into the surrounding temperature-T heat bath, at least if it must treat it as a random perturbation whose correlation structure is unknown[6], must consume energy of $\geq k_B T \ln(1 + 2\sqrt{2}) > 1.34 k_B T$ per bounce, and the whole point of the Fredkin scheme – to avoid the $k_B T \ln 2$ power dissipation requirement by being reversible – is then refuted. In fact, since $1 + 2\sqrt{2} > 2$, the Fredkin scheme actually does *worse* than an optimal irreversible strategy!

If we instead postulated that each collision imparts an additive "random thermal noise" kinetic energy term to the balls involved, of magnitude $k_B T/2$ per degree of freedom (the equipartition principle implies that this would happen if each ball collision achieved "thermal equilibrium"), which we need to get rid of, we would again reach a similar conclusion[7]. And indeed, approximately this amount of kinetic energy noise would be expected if each ball were nonrigid and its surfaces were thus oscillating according to thermally excited vibrations inside each ball.

Either way, it seems very plausible we must dissipate order $k_B T$ energy per bounce.

Conversely, we claim this is tight: it *is* possible to accomplish the necessary error corrections with power dissipation of order $k_B T$ per bounce. The idea is to route the balls in "alleys" with walls which contain segments connected to damped harmonic oscillators free to move in the wrong (i.e. not parallel to the pipe) directions. The harmonic oscillator restoring forces and the damping prevent the balls from going "sideways" very far or acquiring very large sideways velocity, no matter how long the free path. The balls also cannot arrive at their destinations very much early or late, because each alley contains a time measuring device (to tell how far from an integer valued time it was, when the rod arrived at some location) attached to a little "kicker" halfway down the alley which increases or decreases the ball's velocity appropriately. The dampers dissipate order $k_B T$ heat into a bath of viscous oil, which we need to refrigerate; and the kickers expend energy of order $k_B T$ per kick.

Claim 3 *Power dissipation of order $k_B T$ per bounce seems both necessary and sufficient in principle to do computation in Fredkin's billiard ball model.*

But is this necessary for *all* schemes of this nature, or is it just a fluke associated with Fredkin's *particular* scheme? As we will now see, the answer appears to be the latter.

3 Reducing the Lyapunov exponent to zero

We say a system has "Lyapunov exponent" L if some infinitesimal perturbation grows with time t like $\exp(Lt)$. The key ingredient of Zurek's entropy counter-argument was the fact that a Lyapunov exponent of the billiard ball model was $\geq 1 + 2\sqrt{2}$ per bounce. We will now see how to reduce this to 0.

The idea is as follows. Instead of using balls, we use long rigid rods (all identical prisms of square cross-section) sliding at unit velocity in frictionless "pipes" (of rectangular cross-section) – see figure 5.

Pipes can bend (with appropriate widening and alteration of the cross-section shape so that the rods can pass through the bends) in 3D, allowing us to avoid the need for special "crossover" and "time delay" devices. Pipes can also cross in 2D at right angles, implementing the same "AND with one input inverted" logic gate (assuming that the delays have been chosen appropriately so that the rods collide in roughly the configuration shown) as was pictured in the top left of figure 3.

It is essential for our conception that the rods have only *one* degree of freedom – that of sliding along their pipe – and no others (rotating, rattling, vibrating). In principle this is

[5] Even imperfect sphericity would not matter in principle if we could achieve perfect 4-way symmetry of our molecules (provided they never rotated). But actually uncertainty principles would force such rotational misalignements to occur, so this is an idle hope.

[6] This caveat prevents this argument from qualifying as a "proof," which is why I call the final result a "claim" rather than a "theorem." But it is very plausible that any computation of the correlation structure must be at least as difficult a computation as the one the balls are performing (assuming they are performing an optimal algorithm), so that it is not possible to take advantage of knowledge of correlations.

[7] Waiting many collisions before getting rid of this heat would consume more power than getting rid of it once per collision, due to the exponential error buildup.

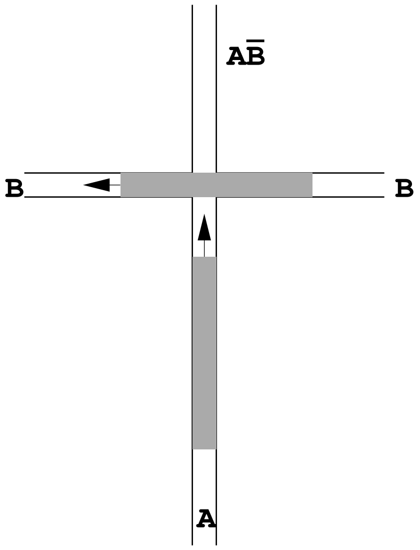

Figure 5: AND gate with an input negated in rod/pipe model. The back-bounced 'A' rod could later be bounced forward again, but will be unable to catch up with the leading edge of the computation to interfere with it.

possible (in classical mechanics of rigid bodies) by shaping the pipes correctly. To see that, imagine 2D rods and pipes for simplicity. Suppose we wish the right-hand side of the pipe to be a segment of some known smooth convex curve C. Suppose the rods are $2 \times w$ rectangles, $w \ll 2$. Then we can define a second curve B to be the envelope of all the length-2 secant line segments of C, and a third curve A to be the curve "parallel to" B but a distance w inside it. Then A will define the left-hand side of the pipe. Now

Lemma 4 *Let C be a smooth convex plane curve with radius of curvature (which may vary with position) R. If $R > 1$ then a unique smooth curve B always exists. If $R > 1 + w$ then a unique smooth curve A always exists and the rod will always feature a 3-point contact with the pipe walls (at its two right-hand corners touching C, and some non-corner point on its left edge, touching A) so that it indeed has only one degree of motional freedom[8].*

Proof sketch: Any two length-2 secants of C whose endpoints are dovetailed in order along C, must cross at a unique point due to the dovetailing and C's convexity. If these line segments are parameterized by the arclength s along C of their lower endpoint, then segment(s) will cross segment ($s + \mathrm{d}s$) at a unique point $B(s)$. The locus of the points $B(s)$ will define the curve B, and B is easily seen to be convex and smooth if C is and $R > 1$. Now additionally A will always exist (and be convex and smooth) if $R > 1 + w$. □

Incidentally, even for nonrigid rods and pipes, and slightly imperfectly made pipes, still we can force one degree of motional freedom due to *quantum* effects – if the frequency f of any vibrational mode, or rod-in-pipe rattling scenario, obeys $hf \gg E$, where E is the kinetic energy of two rods, then it will be impossible to excite these degrees of freedom.

As we've already said, the logic gate in figure 5 is complete if hardwired 1's and 0's are available (and bit-copying, but this may be accomplished if you have hardwired 1's and 0's, because, e.g. $1\overline{1\overline{x}} = x$.)

Hardwired 0's are of course trivial (do nothing). Single hardwired 1's are also easy (a precisely timed incoming rod). However, unlike in the Fredkin model, it *seems* to be impossible to create *streams* of hardwired 1's, equally spaced in time. Figure 6 shows the failure of an attempt to accomplish that.

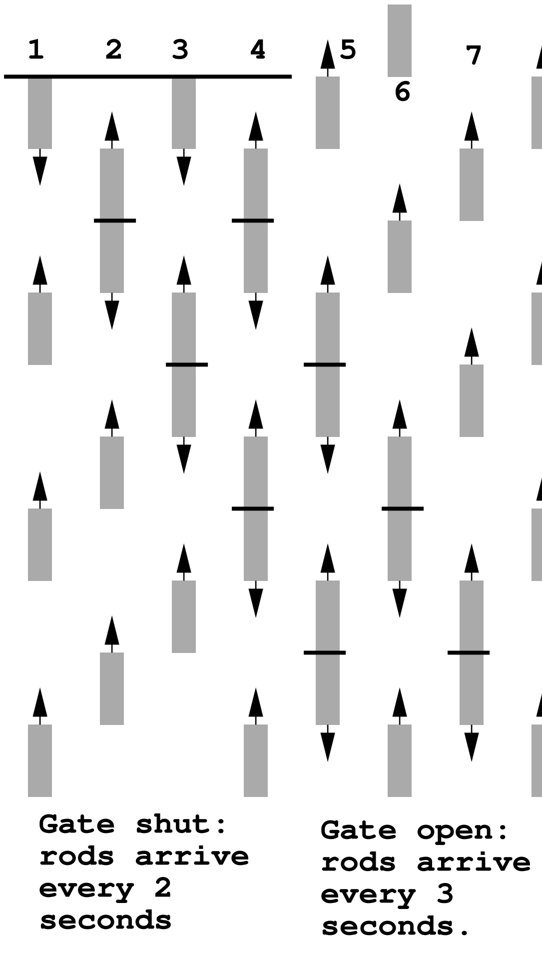

Figure 6: Failure of attempt to create time-periodic stream of 1's.

Therefore, although we can still create a Turing machine with bounded "tape,", we apparently cannot do so by building a circuit to perform 1 Turing step and then feeding its outputs back to its inputs; we instead need to create N such circuits, in series, each of pure feedforward design, to perform an N-step Turing computation.

The same remarks (as after the proof of theorem 2) apply here about undoing the computation to reset the computer to allow its later re-use.

Theorem 5 (Turing machine emulation) *Any N-step*

[8] One can actually write down expressions for $B(s)$ for many common C. In particular if C is a line, B is the same line and A is a parallel line w away. If C is a circle of radius R, then B and A are concentric circles of radius $\sqrt{R^2-1}$ and $\sqrt{R^2-1} - w$ respectively.)

calculation performed by a Turing machine of tape length L, may be emulated by $O(LN)$ pipes and rods in the rod-in-pipe model.

In the rod/pipe model, notice that a small error (either: a small 1D positional displacement of a rod from its desired position, or a small perturbation in the velocity of a rod) is *not magnified* during a collision. Errors due to incorrect velocities or to manufacturing errors (rods of slightly wrong length) can build up, but not exponentially, instead only linearly, with time (e.g. a wrong velocity rod will have a positional error increasing linearly with time) and the operation of the machine will be entirely unaffected until these errors exceed some threshold.

If we postulate that each 90° collision adds a random kinetic energy to the rod whose direction is reversed, of order $k_B T/2$, then small errors in velocity will tend to increase proportionally to the square root of the number of bounces, causing positional errors to grow like the 3/2 power of the number of bounces so far. Manufacturing errors, if uncorrelated with bounces, will produce only linear growth in positional error at worst, and hence ultimately will not be as important as these thermal errors.

This *sub-exponential* growth of errors corresponds to a rate of entropy increase which is *arbitrarily small* per bounce. A similar correction scheme based on time of arrival measurements and corrective "kicks" would then seem capable of waiting for many bounces before applying a correction[9] and expending, on average, an arbitrarily small multiple of $k_B T$ per bounce.

For example, here is an explanation of the claim in the abstract that with perfectly rigid rods of constant length moving at constant velocity, P (the number of bounces per $k_B T$ energy dissipation) grows proportionally to the sixth root of their masses. Let the rods each have unit length, unit velocity, and mass m. Assume each bounce causes a random velocity perturbation of order $\sqrt{k_B T/m}$. Thus in time t we expect a positional perturbation of order $t^{3/2}\sqrt{k_B T/(\tau m)}$ where τ is the typical time delay between logic gates. Thus we can afford to wait a time $t_c \approx [\tau m/(k_B T)]^{1/3}$ before applying a correction, and the correction will consume energy of order $k_B T\sqrt{t_c/\tau}$ per rod. Hence the energy expenditure (in units of $k_B T$) per bounce (i.e. per time τ) is $\lesssim \sqrt{\tau/t_c}$. Hence P, the number of bounces achievable per expenditure of $k_B T$ energy, is the reciprocal of this, i.e.

$$P \gtrsim \tau^{-1/3}(k_B T)^{-1/6} m^{1/6}. \tag{5}$$

Because 1D and 2D "cellular automata" [13] both can perform universal computation, the maximum pipe length between gates could be kept below some constant times the length of a rod, so with unit rod velocities we may regard τ as a constant, and of course we assume $k_B T$ is a constant, so the dependence of P on m is $\propto m^{1/6}$ as claimed.

[9]Unlike in footnote 7, this waiting idea now pays off because the energy required for a correction grows sublinearly with the number of bounces, as opposed to exponentially. We can wait until just before the point where the threshold error magnitude is reached, and this threshold depends like a power law formula on parameters governing how "ideal" the rigid rods are.

3.1 Side remark about memory cells

Any computer engineer must find it peculiar that we can build a "computer" but have nowhere indicated how to build a read/write memory cell capable of storing 1 bit indefinitely. That is because our computers have no such memory cells – their state is dynamically maintained as flowing signals.

It turns out to be possible to build such a memory cell in both Fredkin's original billiard ball model and my rod/pipe modification. The bad news is, I still cannot see any way to *use* such memory cells (in my model) to build a finitary "Von Neumann machine." (That is, in my model, all constructions known so far of Von Neumann machines would, in order to run for N timesteps, require extra balls, or other hardware, growing unboundedly with N.) This contrasts with the original Fredkin model, in which that is possible [15].

A memory cell in my model is illustrated in figure 7.

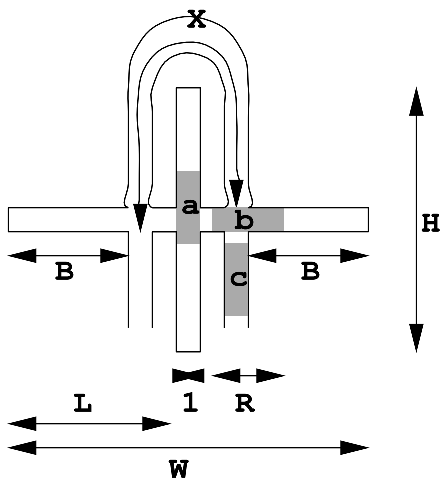

Figure 7: 1-bit memory cell. Rod a is moving down, b left, and c up, all at unit speed. $R \times 1$ rods. Lengths: $R = 3$, $B = 5$, $L = 7$, $H = 11$, $W = 15$, $X = 16$.

Rod b is trapped in the right half of the memory, representing storing a '1' bit. It bounces back and forth, always hitting rod a every 8 time units. The fact that the memory cell as shown contains a '1' may be verified by use of the "reading rod" such as c. To toggle the state of the memory, we can send in a "writing rod" such as a version of c delayed by 4 time units. This will be hit by b from the right, causing b to be delayed by 4 time units, enabling it to be "out of phase" with a, enabling it to get through to the left hand side of the apparatus. There, it gets delayed by another 4 time units by a second interaction with the writing rod, getting back in sync with rod a and then remaining trapped indefinitely on the left side. (The memory has toggled from '1' to '0.') We can similarly toggle it back to a '1.' Note, multiple read and/or write pipes for rods such as c are permissible by crossing in the 3rd dimension or at different locations. Thus reading need not interfere in any way with writing, and multiple writes need not interfere with one another.

However, it does not seem possible to re-read this memory more than a constant number of times (assuming its state is always unknown to the reader) because the reading rods will interfere with each other! This is yet another illustration of the fact that in my variant of Fredkin's scheme, I generally only know how to design *feedforward* circuitry (or anyhow, circuitry which feeds back no more than a constant number of times).

In Fredkin's original billiard ball model, it is possible to use essentially the same idea to construct a similar 1-bit memory cell. (The perpetual bouncing is in polygonal closed paths rather than back and forth.) However, that cell, unlike ours, may be read an unbounded number of times. It is also possible in the Fredkin model to build a multibit memory by using a stream of recirculating bits. Other ideas are in [15].

I suspect that *no* modification like mine of Fredkin's billiard idea is capable of simultaneously achieving zero Lyapunov exponent and allowing circuits to be reused. This is because allowing reuse seems always to require the balls to bounce in some direction other than 180° backwards. But that and an assumption of imprecision in arrival times, seems to require allowing *two* dimensional freedom of motion, and that in turn seems to allow entropy loss during collisions, at which point you are dead.

4 Conclusion

It is, at least on paper, possible to build a reusable "reversible ballistic computing machine" consuming k_BT energy per P logical operations, where P can be made arbitrarily large (growing as a power law in certain physical parameters describing how close to ideal the system is). Hence there cannot be any easy way to rule out the existence of such machines.

But, I nevertheless suspect that there *is* a "no free lunch" principle. I will now quickly survey some known results about zero power computing, and will formulate (admittedly rather vaguely) a "no free lunch principle" which seems compatible with all of them. Future work should concentrate on trying to confirm, refute, or refine this conjecture.

1. Fredkin's billiard ball model (BBM) exponentially amplifies trajectory errors and therefore must consume energy $\geq 1.34 k_BT$ per bounce if those errors are corrected (since error correction decreases entropy).

2. But my rod-in-pipe modification of Fredkin's BBM involves *sub*exponential error growth and therefore genuinely does have "zero" power consumption. But this comes at a price: I seemingly have to use an amount of hardware growing proportional to N to emulate (in time N) N steps of a Turing machine. In contrast the original Fredkin BBM could do so with hardware proportional to Turing machine tape length (for a reversible Turing machine, anyway) *independent* of N. That is because Fredkin could keep re-using the same transformation over and over inside the same hardware; I don't know how to reuse my hardware (once used, it gets "contaminated") without a total reset.

3. One can also make hybrids of zero-power and conventional logic.

4. C.Bennett's "enzymatic DNA Brownian" [7] physical implementation of a reversible Turing machine (RTM) consumes zero power but the price is: it runs N RTM steps in expected time proportional to N^2.

5. One can make a "biased Brownian motion" variant which consumes more power than zero but runs faster than N^2. Specifically, you can consume energy $k_BTN^{1-\beta}$ for any $0 \leq \beta \leq 1$ to do N RTM steps in time $O(N^{1+\beta})$.

6. Other physical schemes have been suggested [10] [12] for low power computation. These both allow reduction of energy consumption by a factor of X, but again at the cost of making the computation run X times more slowly.

7. Benioff [1] [2] has analysed quantum models of computation, *but* so far as I know, no satisfactory analysis of such a model has been made *in the presence of nonidealities.* These will cause "reflections backward in time" of the computation in a manner rather analogous to how "electron phonon scattering" can cause electrical resistance in crystals. Presumably power will then be necessarily consumed to fix this, but how much? It naively requires hardware of order LN to emulate an RTM computation N steps long (with tape length L), but maybe something better is possible – what is the required growth in hardware with N?

Conjecture 6 (Tentative "no free lunch principle") *"Zero power" computing can be done, but it comes at a price. That price is: performing N computational steps requires a factor of $\geq N^\beta$ extra hardware×delay product, if the energy consumption is reduced from order k_BTN (the minimum possible value with conventional irreversible steps) to order $k_BTN^{1-\beta}$ (for any $0 \leq \beta \leq 1$).*

5 Acknowledgement

Harold Stone pointed out some missing details in my exposition in an earlier draft; in response I added §3.1, §4, and lemma 4.

References

[1] P.Benioff: Quantum mechanical models of Turing machines which dissipate no energy, Phys. Rev. Lett. 48,23 (1982) 1581-1584.

[2] P.Benioff: The computer as a physical system: a microscopic quantum mechanical Hamiltonian model..., J.Stat.Phys. 22,5 (1980) 563-591.

[3] C.H.Bennett: Logical reversibility of computation, IBM J. Research & Devel. 17 (1973) 525-532.

[4] C.H.Bennett: The thermodynamics of computation - a review, Int'l J. Theor. Phys. 21 (1982) 905-940.

[5] C.H.Bennett: Time/space tradeoffs for reversible computation, SIAM J.Comput. 18,4 (1989) 766-776.

[6] Charles Bennett: Notes on the History of Reversible Computation, IBM J.Research and Development 32, 1 (January 1988) 16-23.

[7] Charles H. Bennett and Rolf Landauer: The Fundamental Limits of Computation, Scientific American (July 1985) 48-56 (38-53 in some versions).

[8] R.P.Feynman: Feynman lectures on computation, (ed. Anthony J.G. Hey & Robin W. Allen) Addison-Wesley 1996.

[9] E.Fredkin & T.Toffoli: Conservative logic, Int'l J. Theor. Phys. 21 (1982) 219-253. Warning: major printing errors occurred at the transitions between pages 245-6 and 247-8.

[10] J.G.Koller, W.C.Athas, L."J."Svenson: Thermal logic circuits, PhysComp 94 (1994) 119-127.

[11] R.Y.Levine & A.T.Sherman: A note on Bennett's time space tradeoff for reversible computation, SIAM J. Comput. 19,4 (1990) 673-677.

[12] K.K.Likharev: Classical and quantum limitations on energy consumption in computation, Int. J. Theor. Phys. 21 (1982) 311-326.

[13] N.Margolus: Physics-like models of computations, Physica 10D (1984) 81-95.

[14] M.L.Minsky: Computation: finite and infinite machines, Englewood Cliffs, N.J., Prentice-Hall 1967, & reprinted.

[15] Andrew L. Ressler: *Practical circuits using conservative reversible logic*, Bachelor's thesis MIT 1979; *The design of a conservative logic computer and a graphical editor simulator*, Master's thesis, MIT Artificial Intelligence Laboratory, 1981.

[16] Tommaso Toffoli: Reversible Computing, MIT Lab for Computer Science Technical memo MIT/LCS/TM-151 (Feb. 1980). (545 Technology square, Cambridge MA 02139 USA.) 37 pages. Bit map available electronically at `http://pm.bu.edu/tt/publ/revcomp-rep.pdf`. A 13-page abbreviated version was published in Int'l Colloq. Automata, Languages, Programming 7 (1980) 632-644, Springer (LNCS #85).

[17] T.Toffoli: Comment on 'dissipation in computation', Phys. Rev. Lett. 53, 12 (1984) 1204.

[18] W.H.Zurek: Reversibility and stability of information processing systems, Phys. Rev. Lett. 53,4 (1984) 391-394.