C Reversible computing

C.1 Reversible computing as a solution

This section is based on Frank (2005b).

C.1.a Possible solution

¶1. Notice that the key quantity F_{E} in Eqn. II.1 depends on the energy *dissipated* as heat.

¶2. The $100k_{\mathrm{B}}T$ limit depends on the energy in the signal (necessary to resist thermal fluctuation causing a bit flip).

¶3. There is nothing to say that information processing has to dissipate energy; an arbitrarily large amount of it can be recovered for future operations.

"Arbitrary" in the sense that there is no inherent physical lower bound on the energy that must be dissipated.

¶4. It becomes a matter of precise energy *management*, moving it around in different patterns, with as little dissipation as possible.

¶5. Indeed, E_{sig} can be increased to improve reliability, provided we minimize dissipation of energy.

¶6. This can be accomplished by making the computation *logically reversible* (i.e., each successor state has only one predecessor state).

C.1.b Reversible physics

¶1. All fundamental physical theories are Hamiltonian dynamical systems.

¶2. All such systems are time-reversible. That is, if $\psi(t)$ is a solution, then so is $\psi(-t)$.

¶3. In general, *physics is reversible*.

¶4. Physical information cannot be lost, be we can lose track of it. This is entropy: "unknown information residing in the physical state."
Note how this is fundamentally a matter of *information* and *knowledge*.

What is irreversible is the *information inaccessibility*.
Entropy is ignorance.

C.1.c Reversible logic

¶1. To avoid dissipation, don't erase information. The problem is to keep track of information that would otherwise be dissipated. Avoid squeezing information out of logical space (IBDF) into thermal space (NIBDF).

¶2. This is accomplished by making computation *logically reversible*.

(It is already *physically* reversible.)

¶3. The information is rearranged and recombined *in place*. (We will see lots of examples of how to do this.)

C.1.d Progress

¶1. In 1973, Charles Bennett (IBM) first showed how any computation could be embedded in an equivalent reversible computation. Rather than discarding information, it keeps it around so it can later "decompute" it. This was *logical* reversibility; he did not deal with the problem of *physical* reversibility.

¶2. "DNA polymerization, which (under normal conditions, such as during cell division) proceeds at a rate on the order of only 1,000 nucleotides per second, with a dissipation of $\sim 40 k_{\mathrm{B}} T$ per step."
This is about 1 eV (see ¶4 below).
Note that DNA replication includes error-correcting operations.

¶3. **Energy coefficient:** Since "asymptotically reversible processes (including the DNA example) proceed forward at an adjustable speed, proportional to the energy dissipated per step," define an *energy coefficient*:

$$c_{\mathrm{E}} \stackrel{\mathrm{def}}{=} E_{\mathrm{diss}} / f_{\mathrm{op}},$$

"where E_{diss} is the energy dissipated per operation, and f_{op} is the frequency of operations."

¶4. "In Bennett's original DNA process, the energy coefficient comes out to about $c_{\mathrm{E}} = 1\mathrm{eV/kHz}$."
That is, for DNA, $c_{\mathrm{E}} \approx 40kT/\mathrm{kHz} = 40 \times 26$ meV/kHz ≈ 1 eV/kHz.

¶5. But it would be desirable to operate at GHz frequencies and energy dissipation below $k_{\mathrm{B}}T$.
Recall that at room temp. $k_{\mathrm{B}}T \approx 26$ meV (Sec. A ¶6, p. 31).
So we need energy coefficients much lower than DNA.
This is an issue, of course, for molecular computation.

¶6. **Information Mechanics group:** In 1970s, Ed Fredkin, Tommaso Toffoli, et al. at MIT.

¶7. **Ballistic computing:** F & T described computation with idealized, perfectly elastic balls reflecting off barriers. Minimum dissipation, propelled by (conserved) momentum. Unrealistic. Later we will look at it briefly.

¶8. They suggested a more realistic implementation involving "charge packets bouncing around along inductive paths between capacitors."

¶9. Richard Feynman (CalTech) had been interacting with IM group, and developed "a full quantum model of a serial reversible computer" (Feynman, 1986).

¶10. **Adiabatic circuit:** Since 1980s there has been work in *adiabatic circuits*, esp. in 1990s.
An *adiabatic process* takes place without input or dissipation of energy.
Adiabatic circuits minimize energy use by obeying certain circuit design rules. "[A]rbitrary, pipelined, sequential logic could be implemented in a fully-reversible fashion, limited only by the energy coefficients and leakage currents of the underlying transistors."

¶11. As of 2004, est. $c_{\mathrm{E}} = 3$ meV/kHz, about 250× less than DNA.

¶12. "It is difficult to tell for certain, but a wide variety of post-transistor device technologies have been proposed ... that have energy coefficients ranging from 10^5 to 10^{12} times lower than present-day CMOS! This translates to logic circuits that could run at GHz to THz frequencies, with dissipation per op that is still less (in some cases orders of

magnitude less) than the VNL bound of $k_B T \ln 2$ ... that applies to all irreversible logic technologies. Some of these new device ideas have even been prototyped in laboratory experiments [2001]."

¶13. "Fully-reversible processor architectures [1998] and instruction sets [1999] have been designed and implemented in silicon."

¶14. But this is more the topic of a CpE course...

C.2 Physical assumptions of computing

Optional

These lectures are based primarily on Edward Fredkin and Tommaso Toffoli's "Conservative logic" (Fredkin & Toffoli, 1982).

C.2.a Dissipative logic

¶1. The following physical principles are implicit in the existing theory of computation.

¶2. **P1. The speed of propagation of information is bounded:** No action at a distance.

¶3. **P2. The amount of information which can be encoded in the state of a finite system is bounded:** This is a consequence of thermodynamics and quantum theory.

¶4. **P3. It is possible to construct macroscopic, dissipative physical devices which perform in a recognizable and reliable way the logical functions AND, NOT, and FAN-OUT:** This is an empirical fact.

C.2.b Conservative logic

¶1. "Computation is based on the storage, transmission, and processing of discrete signals."

¶2. Only macroscopic systems are irreversible, so as we go to the microscopic level, we need to understand reversible logic. This leads to new physical principles.

¶3. **P4. Identity of transmission and storage:** In a relativistic sense, they are identical.

¶4. **P5. Reversibility:** Because microscopic physics is reversible.

¶5. **P6. One-to-one composition:** Physically, fan-out is not trivial, so we cannot assume that one function output can be substituted for any number of input variables.
We have to treat fan-out as a specific signal-processing element.

¶6. **P7. Conservation of additive quantities:** It can be shown that in a reversible systems there are a number of independent conserved quantities.

¶7. In many systems they are *additive* over the subsystems.

¶8. **P8. The topology of space-time is locally Euclidean:** "Intuitively, the amount of 'room' available as one moves away from a certain point in space increases as a power (rather than as an exponential) of the distance from that point, thus severely limiting the connectivity of a circuit."

¶9. What are sufficient primitives for conservative computation?
The *unit wire* and the *Fredkin gate*.

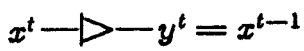

Figure II.10: Symbol for unit wire. (Fredkin & Toffoli, 1982)

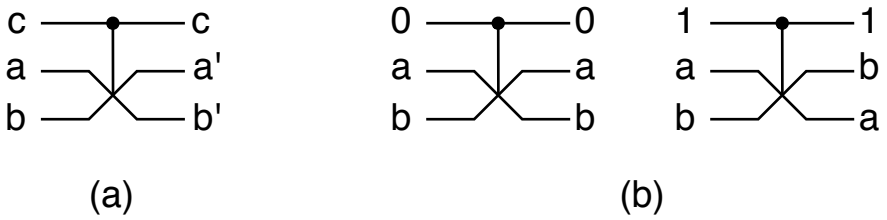

Figure II.11: Fredkin gate or CSWAP (conditional swap): (a) symbol and (b) operation.

C.3 Unit wire

¶1. Information storage in one reference frame may be information transmission in another.
E.g., leaving a note on a table in an airplane (at rest with respect to earth or not, or to sun, etc.).

¶2. The *unit wire* moves one bit of information from one space-time point to another space-time point separated by one unit of time. See Fig. II.10.

¶3. **State:** "The value that is present at a wire's input at time t (and at its output at time $t + 1$) is called the *state* of the wire *at time* t."

¶4. It is invertible and conservative (since it conserves the number of 0s and 1s in its input).
(Note that there are mathematically reversible functions that are not conservative, e.g., NOT.)

C.4 Fredkin gate

¶1. **Conservative logic gate:** Any Boolean function that is invertible and conservative.

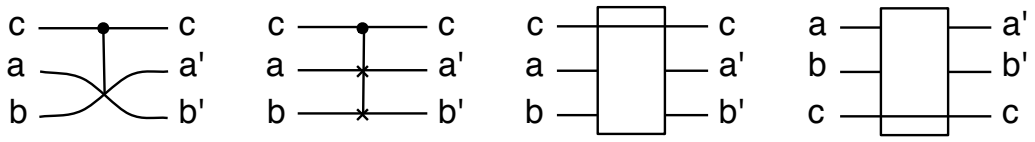

Figure II.12: Alternative notations for Fredkin gate.

¶2. **Conditional rerouting:** Since the number of 1s and 0s is conserved, conservative computing is essentially *conditional rerouting*

¶3. **Rearranging vs. rewriting:** Conventional models of computation are based on *rewriting*
(e.g., TMs, lambda calculus, register machines, term rewriting systems, Post and Markov productions).
But we have seen that overwriting dissipates energy (and is non-conservative).

¶4. In conservative logic we *rearrange* bits without creating or destroying them.
(No infinite "bit supply" and no "bit bucket." Note that these are physically real, not metaphors!)

¶5. **Fredkin gate:** The *Fredkin gate* is a conditional swap operation (also called CSWAP):

$$\begin{aligned}(0,a,b) &\mapsto (0,a,b),\\ (1,a,b) &\mapsto (1,b,a).\end{aligned}$$

The first input is a *control* signal and the other two are *data* or *controlled* signals.
Here, 1 signals a swap, but Fredkin's original definition used 0 to signal a swap.
See Fig. II.11 and Fig. II.14. Fig. II.12 shows alternative notations for the Fredkin gate.

¶6. Note that it is reversible and conservative.

¶7. **Universal:** The Fredkin gate is a universal Boolean primitive for conservative logic.

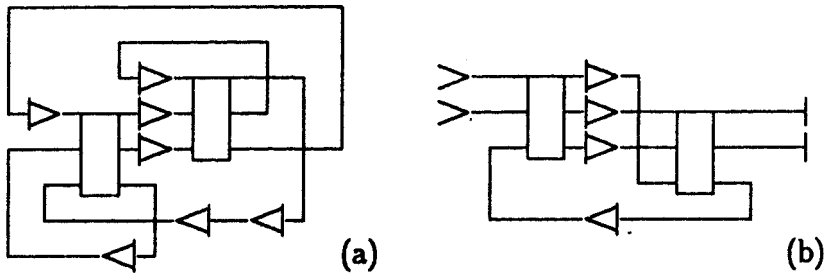

Figure II.13: "(a) closed and (b) open conservative-logic circuits." (Fredkin & Toffoli, 1982)

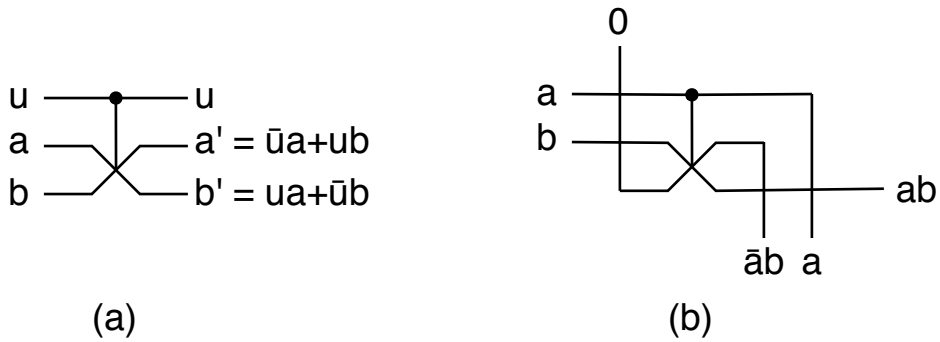

Figure II.14: (a) Logical behavior of Fredkin gate. (b) Implementation of AND gate by Fredkin gate by constraining one input to 0 and discarding two "garbage" outputs.

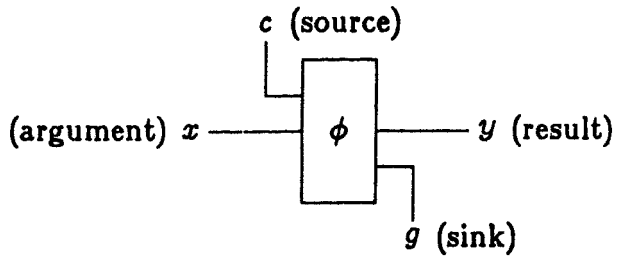

Figure II.15: "Realization of f by ϕ using source and sink. The function $\phi : (c, x) \mapsto (y, g)$ is chosen so that, for a particular value of $c, y = f(x)$." (Fredkin & Toffoli, 1982)

C.5 Conservative logic circuits

¶1. "A conservative-logic circuit is a directed graph whose nodes are conservative-logic gates and whose arcs are wires of any length [Fig. II.13]."

¶2. We can think of the gate as instantaneous and the unit wire as being a unit delay, of which we can make a sequence (or imagine intervening identity gates).

¶3. **Closed vs. open:** A *closed circuit* is a *closed* (or *isolated*) physical system.
An *open circuit* has external inputs and outputs.

¶4. The number of outputs must equal the number of inputs.

¶5. It may be part of a larger conservative circuit, or connected to the environment.

¶6. **Discrete-time dynamical system:** A conservative-logic circuit is a *discrete-time dynamical system.*

¶7. **Degrees of freedom:** The number N of unit wires in the circuit is its number of DoF.
The numbers of 0s and 1s at any time is conserved, $N = N_0 + N_1$.

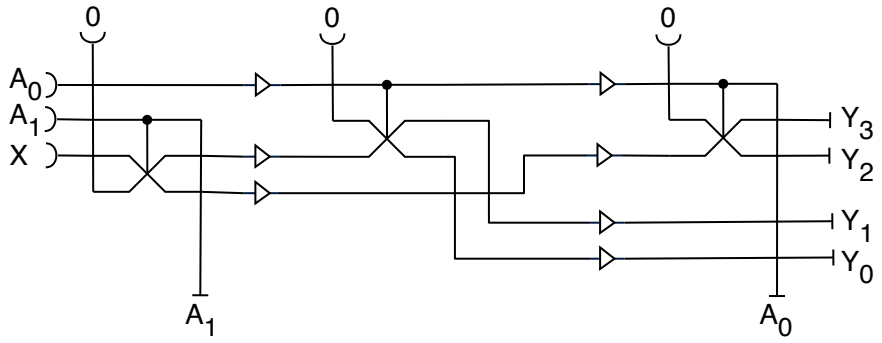

Figure II.16: 1-line-to 4-line demultiplexer. The address bits $A_1A_0 = 00, 01, 10, 11$ direct the data bit X into Y_0, Y_1, Y_2 or Y_3, respectively. Note that each Fredkin gate uses an address bit to route X into either of two wires. (Adapted from circuit in Fredkin & Toffoli (1982).)

C.6 Constants and garbage

¶1. The Fredkin gate can be used to compute non-invertible functions such as AND, if we are willing to provide appropriate constants (called "ancillary values") and to accept unwanted outputs (see Fig. II.14).

¶2. In general, one function can be embedded in another by providing appropriate constants from a *source* and ignoring some of the outputs, the *sink*, which are considered *garbage*.

¶3. However, this garbage cannot be thrown away (which would dissipate energy), so it must be recycled in some way.

C.7 Universality

¶1. **OR, NOT, and FAN-OUT:** Fig. **??** shows Fredkin realizations of other common gates.

¶2. **Demultiplexer example:** Fig. II.16 shows a 1-line to 4-line demultiplexer.

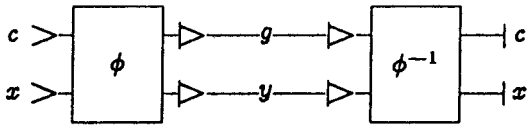

Figure II.17: Composition of combinational conservative-logic network with its inverse to consume the garbage. [fig. from Fredkin & Toffoli (1982)]

¶3. Hence you can convert conventional logic circuits into conservative circuits, but the process is not very efficient. It's better to design the conservative circuit from scratch.

¶4. **Universality:** "any computation that can be carried out by a conventional sequential network can also be carried out by a suitable conservative-logic network, provided that an external supply of constants and an external drain for garbage are available."
(Will see how to relax these constraints: Sec. C.8)

C.8 Garbageless conservative logic

¶1. To reuse the apparatus for a new computation, we will have to throw away the garbage and provide fresh constants, both of which will dissipate energy.

¶2. **Exponential growth of garbage:** This is a significant problem if dissipative circuits are naively translated to conservative circuits because:
(1) the amount of garbage tends to increase with the number of gates, and
(2) with the naive translation, the number of gates tends to increase exponentially with the number of input lines.
"This is so because almost all boolean functions are 'random', i.e., cannot be realized by a circuit simpler than one containing an exhaustive look-up table."

¶3. However there is a way to make the garbage the same size as the input (in fact, identical to it).

Figure II.18: The "spy circuit" for tapping into the output. [fig. from Fredkin & Toffoli (1982)]

¶4. First observe that a *combinational* conservative-logic network (one with no feedback loops) can be composed with its inverse to consume all garbage (Fig. II.17).

¶5. The desired output can be extracted by a "spy circuit" (Fig. II.18).

¶6. Fig. II.19 shows the general arrangement for garbageless computation. This requires the provision of n new constants (n = number output lines).

¶7. Consider the more schematic diagram in Fig. II.20.

¶8. Think of arranging tokens (representing 1-bits) in the input registers, both to represent the input x, but also a supply of n of them in the black lower square.

¶9. Run the computation.

¶10. The input argument tokens have been restored to their initial positions. The $2n$-bit string $00\cdots0011\cdots11$ in the lower register has been rearranged to yield the result and its complement $y\bar{y}$.

¶11. Restoring the $0\cdots01\cdots1$ inputs for another computation dissipates energy.

¶12. **Feedback:** Finite loops can be unrolled, which shows that they can be done without dissipation.
(Cf. also that billiard balls can circulate in a frictionless system.)

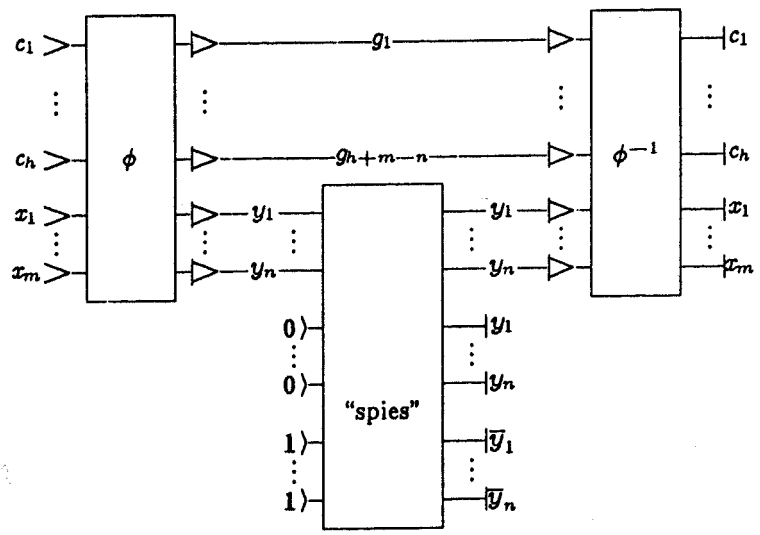

Figure II.19: Garbageless circuit. (Fredkin & Toffoli, 1982)

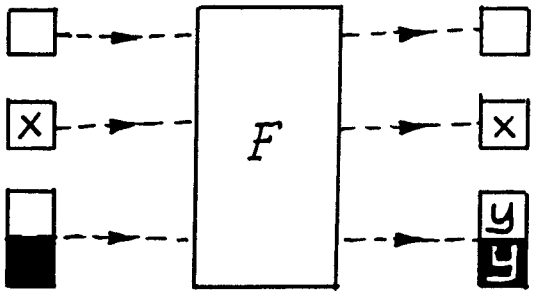

Figure II.20: "The conservative-logic scheme for garbageless computation. Three data registers are 'shot' through a conservative-logic black-box F. The register with the argument, x, is returned unchanged; the clean register on top of the figure, representing an appropriate supply of input constants, is used as a scratchpad during the computation (cf. the c and g lines in Figure [II.19]) but is returned clean at the end of the computation. Finally, the tokens on the register at the bottom of the figure are rearranged so as to encode the result y and its complement $\neg y$" (Fredkin & Toffoli, 1982)

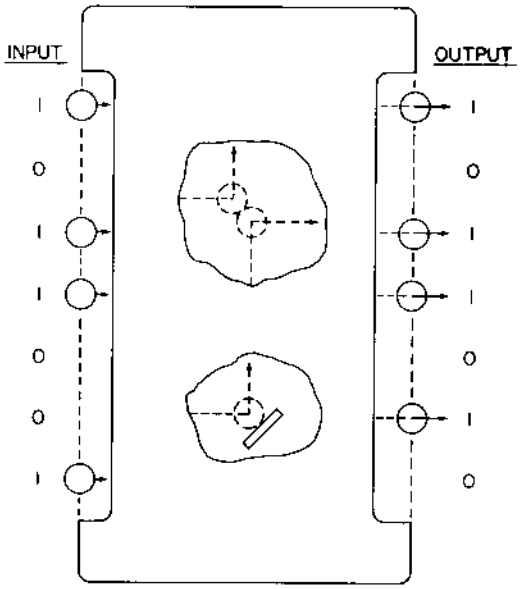

Figure II.21: Overall structure of ballistic computer. (Bennett, 1982)

C.9 Ballistic computation

"Consider a spherical cow moving in a vacuum. . ."

¶1. **Billiard ball model:** To illustrate dissipationless *ballistic computation*, Fredkin and Toffoli defined a *billiard ball* model of computation.

¶2. It is based on the same assumptions as the classical kinetic theory of gasses: perfectly elastic spheres and surfaces.
In this case we can think of pucks on frictionless table.

¶3. Fig. II.21 shows the general structure of the billiard ball model.

¶4. 1s are represented by the presence of a ball at a location, and 0s by their absence.

¶5. Input is provided by simultaneously firing balls into the input ports for the 1s in the argument.

¶6. Inside the box the balls ricochet off each other and fixed reflectors, which performs the computation.

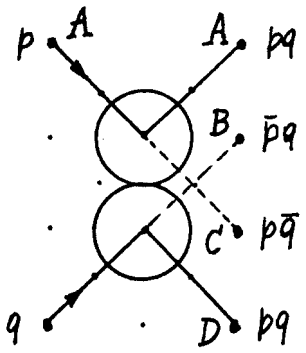

Figure II.22: "Billiard ball model realization of the interaction gate." (Fredkin & Toffoli, 1982)

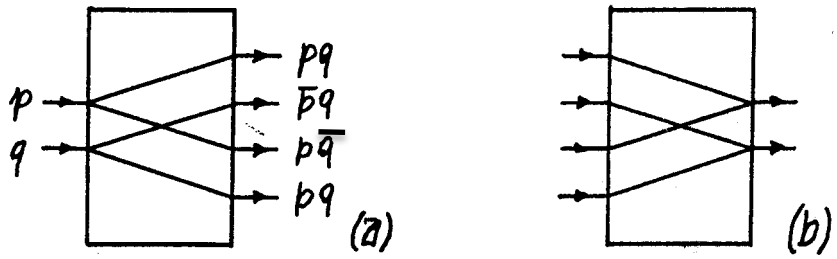

Figure II.23: "(a) The interaction gate and (b) its inverse." (Fredkin & Toffoli, 1982)

¶7. After a fixed time delay, the balls emerging (or not) from the output ports define the output.

¶8. Obviously the number of 1s (balls) is conserved.

¶9. The computation is reversible because the laws of motion are reversible.

¶10. **Interaction gate:** Fig. II.22 shows the realization of the computational primitive, the *interaction gate*.

¶11. Fig. II.23 is the symbol for the interaction gate and its inverse.

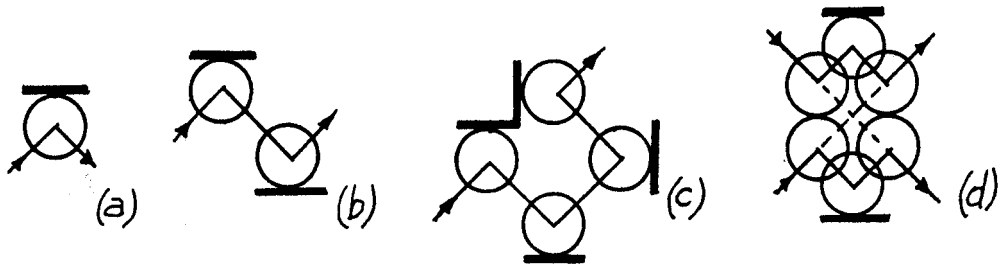

Figure II.24: "The mirror (indicated by a solid dash) can be used to deflect a ball's path (a), introduce a sideways shift (b), introduce a delay (c), and realize nontrivial crossover (d)." (Fredkin & Toffoli, 1982)

¶12. **Universal:** The interaction gate is universal because it can compute both AND and NOT.

¶13. **Interconnections:** However, we must make provisions for arbitrary interconnections in a planar grid. So need to implement signal *crossover* and control *timing*.
(This is *non-trivial* crossover; *trivial* crossover is when two balls cannot possibly be at the same place at the same time.)

¶14. Fig. II.24 shows mechanisms for realizing these functions.

¶15. Fig. II.25 shows a realization of the Fredkin gate in terms of multiple interaction gates. (The "bridge" indicates non-trivial crossover.)

¶16. **Practical problems:** Minuscule errors of any sort (position, velocity, alignment) will accumulate rapidly (by about a factor of 2 at each collision).

¶17. E.g., initial random error of $1/10^{15}$ in position or velocity (about what would be expected from uncertainty principle) would lead to a completely unpredictable trajectory after a few dozen collisions.
It will lead to a Maxwell distribution of velocities, as in a gas.

¶18. "Even if classical balls could be shot with perfect accuracy into a perfect apparatus, fluctuating tidal forces from turbulence in the atmosphere

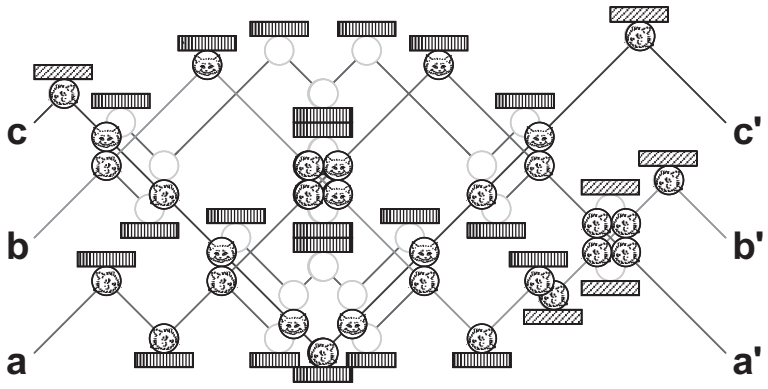

Figure II.25: Realization of the Fredkin gate in terms of multiple interaction gates. [NC]

---

of nearby stars would be enough to randomize their motion within a few hundred collisions." (Bennett, 1982, p. 910)

¶19. Various solutions have been considered, but they all have limitations.

¶20. "In summary, although ballistic computation is consistent with the laws of classical and quantum mechanics, there is no evident way to prevent the signals' kinetic energy from spreading into the computer's other degrees of freedom." (Bennett, 1982, p. 911)

¶21. Signals can be restored, but this introduces dissipation.

D Sources

Bennett, C. H. The Thermodynamics of Computation — a Review. *Int. J. Theo. Phys.*, 21, 12 (1982), 905–940.

Berut, Antoine, Arakelyan, Artak, Petrosyan, Artyom, Ciliberto, Sergio, Dillenschneider, Raoul and Lutz, Eric. Experimental verification of Landauer's principle linking information and thermodynamics. *Nature* 483, 187–189 (08 March 2012). doi:10.1038/nature10872

Frank, Michael P. Introduction to Reversible Computing: Motivation, Progress, and Challenges. CF '05, May 4–6, 2005, Ischia, Italy.

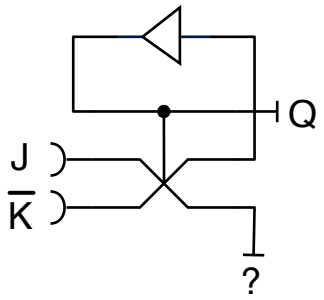

Figure II.26: Implementation of J-$\bar{\text{K}}$ flip-flop (adapted from Fredkin & Toffoli (1982)).

Fredkin, E. F., Toffoli, T. Conservative logic. *Int. J. Theo. Phys.*, 21, 3/4 (1982), 219–253.