C. REVERSIBLE COMPUTING

\[ x^t \rightarrow_{\text{wire}} y^t = x^{t-1} \]

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

\[ \begin{array}{ccc}
    c & \rightarrow & c \\
    a & \rightarrow & a' \\
    b & \rightarrow & b'
    \\
    0 & \rightarrow & 0 \\
    a & \rightarrow & a \\
    b & \rightarrow & b
    \\
    1 & \rightarrow & 1 \\
    a & \rightarrow & b \\
    b & \rightarrow & a
\end{array} \]

(a) \hspace{2cm} (b)

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., \text{Not}.)

C.4 Fredkin gate

1. Conservative logic gate: Any Boolean function that is invertible and conservative.
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.”)

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

\[
(0, a, b) \mapsto (0, a, b), \quad (1, a, b) \mapsto (1, b, a).
\]

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.
A conservative-logic circuit is a directed graph whose nodes are conservative-logic gates and whose arcs are wires of any length (cf. Figure 3).

Figure 3. (a) closed and (b) open conservative-logic circuits.

Any output of a gate can be connected only to the input of a wire, and similarly any input of a gate only to the output of a wire. The interpretation of such a circuit in terms of conventional sequential computation is immediate, as the gate plays the role of an “instantaneous” combinational element and the wire that of a delay element embedded in an interconnection line. In a \textit{closed} conservative-logic circuit, all inputs and outputs of any elements are connected within the circuit (Figure 3a). Such a circuit corresponds to what in physics is called a \textit{closed} (or \textit{isolated}) system. An \textit{open} conservative-logic circuit possesses a number of external input and output ports (Figure 3b). In isolation, such a circuit might be thought of as a \textit{transducer} (typically, with memory) which, depending on its initial state, will respond with a particular output sequence to any particular input sequence. However, usually such a circuit will be thought of as a portion of a larger circuit; thence the notation for input and output ports (Figure 3b), which is suggestive of, respectively, the trailing and the leading edge of a wire. Observe that in conservative-logic circuits the number of output ports always equals that of input ones.

The junction between two adjacent unit wires can be formally treated as a node consisting of a trivial conservative-logic gate, namely, the identity gate. In what follows, whenever we speak of the realizability of a function in terms of a certain set of conservative-logic primitives, the unit wire and the identity gate will be tacitly assumed to be included in this set.

A conservative-logic circuit is a time-discrete dynamical system. The unit wires represent the system’s individual state variables, while the gates (including, of course, any occurrence of the identity gate) collectively represent the system’s transition function. The number $N$ of unit wires that are present in the circuit may be thought of as the number of degrees of freedom of the system. Of these $N$ wires, at any moment $N_1$ will be in state 1, and the remaining $N_0 (= N - N_1)$ will be in state 0. The quantity $N_1$ is an additive function of the system’s state, i.e., is defined for any portion of the circuit and its value for the whole circuit is the sum of the individual contributions from all portions. Moreover, since both the unit wire and the gates return at their outputs as many l’s as are present at their inputs, the quantity $N_1$ is an integral of the motion of the system, i.e., is constant along any trajectory. (Analogous considerations apply to the quantity $N_0$, but, of course, $N_0$ and $N_1$ are not independent integrals of the motion.) It is from this “conservation principle” for the quantities in which signals are encoded that conservative logic derives its name.

It must be noted that reversibility (in the sense of mathematical invertibility) and conservation are independent properties, that is, there exist computing circuits that are reversible but not “bit-conserving,” (Toffoli, 1980) and vice versa (Kinoshita, 1976).

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

\[
\begin{align*}
\text{a} & \quad \text{b} \\
\text{a'} &= \bar{\text{u}}\text{a} + \text{ub} \\
\text{b'} &= \text{ua} + \bar{\text{ub}}
\end{align*}
\]

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.
CHAPTER II. PHYSICS OF COMPUTATION

3. COMPUTATION IN CONSERVATIVE-LOGIC CIRCUITS; CONSTANTS AND GARBAGE

In Figure 4a we have expressed the output variables of the Fredkin gate as explicit functions of the input variables. The overall functional relationship between input and output is, as we have seen, invertible. On the other hand, the functions that one is interested in computing are often noninvertible. Thus, special provisions must be made in the use of the Fredkin gate (or, for that matter, of any invertible function that is meant to be a general-purpose signal-processing primitive) in order to obtain adequate computing power.

Suppose, for instance, that one desires to compute the AND function, which is not invertible. In Figure 4b only inputs \( u \) and \( x_1 \) are fed with arbitrary values \( a \) and \( b \), while \( x_2 \) is fed with the constant value 0. In this case, the \( y_1 \) output will provide the desired value \( ab \) ("a AND b"), while the other two outputs \( v \) and \( y_2 \) will yield the "unrequested" values \( a \) and \( \neg ab \). Thus, intuitively, the AND function can be realized by means of the Fredkin gate as long as one is willing to supply "constants" to this gate alongside with the argument, and accept "garbage" from it alongside with the result. This situation is so common in computation with invertible primitives that it will be convenient to introduce some terminology in order to deal with it in a precise way.

![Figure 4](image)

Figure 4. Behavior of the Fredkin gate (a) with unconstrained inputs, and (b) with \( x_2 \) constrained to the value 0, thus realizing the AND function.

Figure 5. Realization of \( f \) by \( G_3 \) using source and sink. The function \( \phi : (c, x) \mapsto (y, g) \) is chosen so that, for a particular value of \( c \), \( y = f(x) \).

Terminology: source, sink, constants, garbage.

Given any finite function, one obtains a new function \( f \) "embedded" in it by assigning specified values to certain distinguished input lines (collectively called the source) and disregarding certain distinguished output lines (collectively called the sink). The remaining input lines will constitute the argument, and the remaining output lines, the result. This construction (Figure 5) is called a realization of \( f \) by means of \( G_3 \) using source and sink. In realizing \( f \) by means of \( G_3 \), the source lines will be fed with constant values, i.e., with values that do not depend on the argument. On the other hand, the sink lines in general will yield values that depend on the argument, and thus cannot be used as input constants for a new computation. Such values will be termed garbage. (Much as in ordinary life, this...

...Figure II.15: “Realization of \( f \) by \( \phi \) 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, \( N = N_0 + N_1 \).
C. REVERSIBLE COMPUTING

Figure II.16: “Realization of the (a) OR, (b) NOT, and (c) FAN-OUT functions by means of the Fredkin gate.” (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. II.16 shows Fredkin realizations of other common gates.

2. Demultiplexer example: Fig. II.17 shows a 1-line to 4-line demultiplexer.

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
CHAPTER II. PHYSICS OF COMPUTATION

Figure II.17: 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).)

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
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.18).

5. The desired output can be extracted by a “spy circuit” (Fig. II.19).

6. Fig. II.20 shows the general arrangement for garbageless computation. This requires the provision of $n$ new constants ($n = \text{number output lines}$).

7. Consider the more schematic diagram in Fig. II.21.

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.
CHAPTER II. PHYSICS OF COMPUTATION

Figure II.20: Garbageless circuit. (Fredkin & Toffoli, 1982)

Figure II.21: “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.20) 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)
C. REVERSIBLE COMPUTING

10. The input argument tokens have been restored to their initial positions. The $2^n$-bit string 00· · · 0011· · ·1 in the lower register has been rearranged to yield the result and its complement $y\bar{y}$.

11. Restoring the 0· · ·01· · ·1 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.)

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.22 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.

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.
All of the above requirements are met by introducing, in addition to collisions between two balls, collisions between a ball and a fixed plane mirror. In this way, one can easily deflect the trajectory of a ball (Figure 15a), shift it sideways (Figure 15b), introduce a delay of an arbitrary number of time steps (Figure 15c), and guarantee correct signal crossover (Figure 15d). Of course, no special precautions need be taken for trivial crossover, where the logic or the timing are such that two balls cannot possibly be present at the same moment at the crossover point (cf. Figure 18 or 12a). Thus, in the billiard ball model a conservative-logic wire is realized as a potential ball path, as determined by the mirrors.

Note that, since balls have finite diameter, both gates and wires require a certain clearance in order to function properly. As a consequence, the metric of the space in which the circuit is embedded (here, we are considering the Euclidean plane) is reflected in certain circuit-layout constraints (cf. P8, Section 2).

Essentially, with polynomial packing (corresponding to the Abelian-group connectivity of Euclidean space) some wires may have to be made longer than with exponential packing (corresponding to an abstract space with free-group connectivity) (Toffoli, 1977).

Figure II.22: Overall structure of ballistic computer. (Bennett, 1982)

Figure II.23: “Billiard ball model realization of the interaction gate.” (Fredkin & Toffoli, 1982)
Figure II.24: “(a) The interaction gate and (b) its inverse.” (Fredkin & Toffoli, 1982) Note that the second pq from the bottom should be \( \overline{pq} \).

10. **Interaction gate:** Fig. II.23 shows the realization of the computational primitive, the *interaction gate*.

11. Fig. II.24 is the symbol for the interaction gate and its inverse.

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.25 shows mechanisms for realizing these functions.

15. Fig. II.26 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.
with free-group connectivity) (Toffoli, 1977).

some wires may have to be made longer than with exponential packing (corresponding to an abstract space
considering the Euclidean plane) is reflected in certain circuit-layout constraints (cf. P8, Section 2).

function properly. As a consequence, the metric of the space in which the circuit is embedded (here, we are
Note that, since balls have finite diameter, both gates and wires require a certain clearance in order to
wire is realized as a potential ball path, as determined by the mirrors.

moment at the crossover point (cf. Figure 18 or 12a). Thus, in the billiard ball model a conservative-logic
crossover, where the logic or the timing are such that two balls cannot possibly be present at the same

Figure II.26: Realization of the Fredkin gate in terms of multiple interaction
gates. [NC]
18. “Even if classical balls could be shot with perfect accuracy into a perfect apparatus, fluctuating tidal forces from turbulence in the atmosphere 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


