

# A Biomolecular Implementation of Logically Reversible Computation With Minimal Energy Dissipation

Joshua P. Klein, Thomas H. Leete, Harvey Rubin  
University of Pennsylvania, School of Medicine  
536 Johnson Pavilion  
Philadelphia PA 19104  
Email [rubinh@mail.med.upenn.edu](mailto:rubinh@mail.med.upenn.edu)

## Abstract

**Energy dissipation associated with logic operations imposes a fundamental physical limit on computation and is generated by the entropic cost of information erasure, which is a consequence of irreversible logic elements. We show how to encode information in DNA and use DNA amplification to implement a logically reversible gate that comprises a complete set of operators capable of universal computation. We also propose a method using this design to connect, or 'wire', these gates together in a biochemical fashion to create a logic network, allowing complex parallel computations to be executed. The architecture of the system permits highly parallel operations and has properties that resemble well known genetic regulatory systems.**

Logically reversible operations occupy a central role in considerations of the fundamental physical limits of information handling (1). The early work of Landauer (2) showed that energy dissipation occurs during the destruction of information of the previous state of the system rather than the acquisition of information during the computational process. Subsequently, Bennett showed that computation could be carried out completely with operations that are logically reversible, i.e., operations in which the output uniquely defines the input (3). One such reversible logic element is the Fredkin gate (FG) (4) which contains an input control channel A, and two additional input channels, B and C, which exchange values if A is set at 1 or will go through the gate unchanged if A is set at 0 (Fig. 1a). Fredkin gates constitute a complete set of operators in that any logic operation (e.g., AND, OR, NAND, NOT) can be constructed from a combination of FGs (Fig. 2a).

Once the theory of reversible computation was accepted and the relationship of the theory to the practical utility of low energy computation appreciated, intense activity ensued to build such devices based on a wide variety of technologies including optical gate interferometers (5), magnetic bubbles (6), Josephson junctions or discrete state systems (single electron parametron) (7), split level charge recovery devices (8), and quantum devices (9). It is historically interesting to note that Bennett proposed in his original paper (3), and again in a review with Landauer (10), that enzymatic

reactions close to equilibrium could represent a chemical implementation of reversible operations. Bennett, and also Feynman (11), suggested RNA polymerase as an example of a reversible COPY operation calling it an enzymatic Brownian Turing machine. RNA polymerase catalyzes the addition of a new base onto RNA chains by nucleophilic attack on ribonucleoside triphosphate (NTPs), yielding the newly elongated RNA chain and pyrophosphate ( $PP_i$ ) as products of the reaction. As Bennett pointed out (12), the energy loss per forward step in this example is equal to  $kT \ln r$ , where  $r$  is the ratio of concentrations of NTPs to  $PP_i$  relative to their equilibrium values,  $T$  is temperature in degrees Kelvin and  $k$  is Boltzmann's constant. At equilibrium,  $r = 1$ ,  $DG = 0$  and there is no energy dissipation but also no potential to drive the reaction forward. If, however, the concentrations of nucleotides are 1% greater than the equilibrium concentrations, the energy dissipation per step is  $kT \ln(10.1/10)$  or approximately  $0.01kT$ , vastly less than the dissipation from a standard silicon switching device which can be on the order of  $10^6kT - 10^8kT$  per step. Unfortunately, the COPY operation is limited in its computational scope. DNA amplification, however, can combine different elements by hybridization and polymerization, which permits more powerful computation. DNA polymerases, like RNA polymerase, can operate at near equilibrium conditions, therefore dissipating arbitrarily little energy. As we will show, DNA amplification can occur in an isothermal system, eliminating energy loss through heating and cooling of the reaction. Others realized that enzymatic reactions could in principle be constructed to simulate logic elements, however, prior efforts to implement a real device with these elements have achieved limited success (13).

Recent work using DNA as a computational element stimulated a new approach to problems in computation, complexity theory and combinatorics (14, 15, 16, 17). While the potential to carry out massively parallel computations with DNA inspired much of the early work in the field (18, 19), DNA can also create some of the primitives of computation (20).

Unifying the ideas of reversibility and biomolecular primitives with respect to computation, we designed a FG based on the enzymatic reactions that are used in DNA amplification and show how to build the full truth table of the FG biochemically. We also propose a method using this design to connect, or 'wire', these

gates together in a biochemical fashion to create a logic network.

To implement the biomolecular FG, we encode the input channels on strands of DNA (input templates) and use DNA primers to 'read' and operate on these templates. The input template is a 60 base pair double stranded DNA molecule, which contains three regions, A (10bp), B (25bp), and C (25bp), which encode the channels of the FG in the format 5'-CAB-3' (Fig. 1b). A, B, and C each have two possible values, 0 and 1. Each value is represented by two distinct sequences (designated B1, B1\*, B0 and B0\* for the B channel, etc.) which are used to differentiate output from input templates. The fundamental process we wish to accomplish is the transformation of an input DNA sequence to an output DNA sequence maintaining the overall coding design, so that the output can be used directly as a new input. We

achieve this by using a set of primers which contain the output template linked to regions at the 3' end of the primer which can hybridize to the input template. DNA polymerase requires the 3' end of the primer to be correctly hybridized to the template to initiate polymerization. When a given primer is mixed with a given input template, the 3' end of the primer will either correctly anneal and allow polymerization if the values match, or not anneal correctly if the values do not match, prohibiting any reaction. In this way the 3' end of the primer 'reads' the value of the corresponding channel of the input template. We include information about the C bit in the sequence of the A bit so that all three bits are read at once. This is done by using different sequences to represent each value of A depending on the relative values of B and C, as outlined in Table 1.

Table 1: Representation of A Sequences.

Value of A	Relative values of B and C	Sequence of A (see Table 2)
1	B not equal to C	X
1	B equals C	Y
0	B not equal to C	U
0	B equals C	V

It is important to note that primers are added to each reaction in pairs. The setup of our Fredkin gate allows us to always add the appropriate primer pair so that there is no possibility of generating an ambiguous product (i.e. a product where A = X but B = C). Although we are using more than one sequence to represent a given value for the A bit, the function of the A channel does not change; the A channel will switch the values of B and C whether the sequence is X or Y.

The FG physically consists of sets of PCR primers, DNA polymerase, dNTPs and buffer, and operates in three steps. First, the input template mixture is aliquoted to eight different reaction vessels (I-VIII), each of which contains a different primer pair. Each pair of primers will only correctly hybridize to one of the eight possible input templates. The first primer in each pair is 95 bp long, has the structure 5'-C\*A\*B\*CA-3', and thus 'reads' the control bit (A) on the input template. The second primer is 25 bp long, has the structure 5'-Br-3' ('r

denotes complimentary strand sequence), and 'reads' the B bit of the input template. Step 1 creates a 120bp intermediate that has the structure 5'-C\*A\*B\*CAB-3', and represents the covalent attachment of the output to the input. Step 2 is an amplification of this entire intermediate using a C\* and a Br primer, which has been found empirically necessary for reliable functioning of the gate. This step also serves as a secondary check for the first step, as we only add the appropriate pairs of C\* and Br primers in each well. Step 3 amplifies the 60bp output template using C\* and B\*r primers, and again serves as a check for the appropriate reaction by our choice of primer pairs. Note that the output template has the same form as an input template, and may immediately be run backward through the same gate. We demonstrate this experimentally for the input/output templates C0 AX B1 --> C1\* AX\* B0\* and the reverse, C1\* AX\* B0\* --> C0 AX B1.

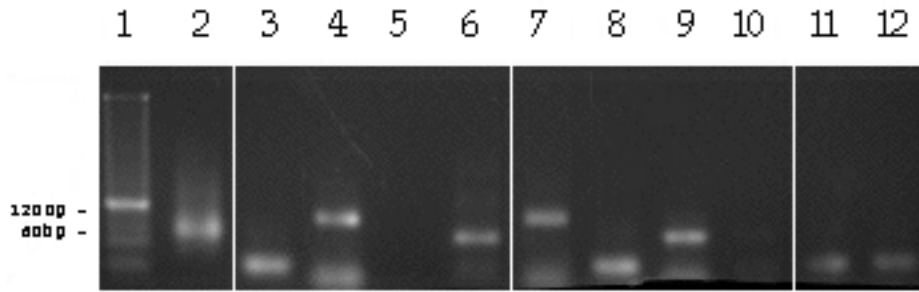


Figure 1: PCR products of the recoder-based Fredkin gate and a schematic drawing of the sequential PCR reactions; 5ul of 50ul reactions on a 3% agarose gel stained with ethidium bromide. PCR reactions consist of 0.5ul template DNA (2-5ng), 0.8ul 100uM each top strand primer (50-100ng), 0.8ul 100uM each bottom strand primer (50-100ng), 5ul 2mM dNTP mixture, 5ul 10X Perkin-Elmer buffer (100mM Tris [pH 8.3], 500mM KCl, 15mM MgCl<sub>2</sub>, 0.1% gelatin), 0.5ul AmpliTaq polymerase (Perkin-Elmer) in 50ul total volume. Reaction conditions were [94°/15sec., 50°/20sec.] X 20 cycles on a Hybaid Omn-E thermal cycler. Bands at 60bp and 120bp represent amplified products; bands at 25bp represent primers. Lane 1, 25bp ladder; lane 2, C0 AX B1 template; lane 3, C0 AX B1 + [C0\* AX\* B1\* C1 AX, C0 AX B1 C1\* AX\*, B0r, B0\*r] after end amplification => no product; lane 4, C0 AX B1 + [C1\* AX\* B0\* C0 AX, C1 AX B0 C0\* AX\*, B1r, B1\*r] after end amplification => C1\* AX\* B0\* C0 AX B1; lane 5, negative product from lane 3 + [C0, C0\*, B1r, B1\*r] => no product; lane 6, C1\* AX\* B0\* C0 AX B1 from lane 4 + [C1, C1\*, B0r, B0\*r] => C1\* AX\* B0\*; lane 7, C1\* AX\* B0\* + [C0\* AX\* B1\* C1 AX, C0 AX B1 C1\* AX\*, B0r, B0\*r] after end amplification => C0 AX B1 C1\* AX\* B0\*; lane 8, C1\* AX\* B0\* + [C1\* AX\* B0\* C0 AX, C1 AX B0 C0\* AX\*, B1r, B1\*r] after end amplification => no product; lane 9, C0 AX B1 C1\* AX\* B0\* + [C0, C0\*, B1r, B1\*r] => C0 AX B1; lane 10, negative product from lane 8 + [C1, C1\*, B0r, B0\*r] => no product. Negative control (minus template) reactions--lane 11 [C0\* AX\* B1\* C1 AX, C0 AX B1 C1\* AX\*, B0r, B0\*r] => no product, and lane 12 [C1\* AX\* B0\* C0 AX, C1 AX B0 C0\* AX\*, B1r, B1\*r] => no product.

While it appears that we are unduly influencing the outcome of these reactions by our choices of primers at each step, it should be noted that we are not 'choosing' at all. At each step, the primers added to each well are predetermined, and do not change regardless of the values encoded in the input template mixture. While each reaction vessel operates on the input template mixture, an output strand will be generated *if and only if* both primers in a given reaction vessel correctly read an input strand in the input template mixture. The results of all eight reactions are then mixed prior to exit from the gate to create the output template mixture. Therefore, the output has the same biomolecular and logical architecture as the input, without direct operator knowledge of the input.

Universal computation is accomplished by rewiring the desired output channels from  $FG_i$  into the subsequent input channels of  $FG_{i+1}$  according to any given wiring diagram. We will accomplish the rewiring of output templates from  $FG_i$  into input templates for  $FG_{i+1}$  using a process analogous to the operation of the gate itself. During the rewiring process, the primer pairs will be called rewiring strands, and have a slightly different structure. In step 1 of the rewiring, an 85 bp long rewiring strand (with the structure 5'-CABC\*-3') anneals to the output strand (5'-C\*A\*B\*-3') at the C\* region, and polymerase fills in the single stranded regions to form a 120 bp long double stranded intermediate product (5'-CABC\*A\*B\*-3'), which is analogous to step 1 of the Fredkin gate. Step 2 of the rewiring then amplifies the entire intermediate product with end primers C and B\*r. Step 3 amplifies the rewired template with C and Br primers to generate a new input template (5'-CAB-3').

To transfer the value of the C\* position of the FG output strand to one of the positions of the new input template, we synthesize a library of rewiring strands such that for a given rewiring strand, the values of the C\* and the channel to which the information will be transferred are the same. Thus if we want to transfer the C\* value to the A channel, we would use rewiring strands that have the same values in these two positions (e.g. C0 AX B1 C1\*). We refer to this as a C\*-A (C\* to A) recoder. Likewise, a C\*-B recoder would recode the C\* value to the B channel. In instances where only a single value needs to be recoded into the new input template (i.e. input to a NOT gate which has only one variable channel and the other two channels are defined), a single rewiring is sufficient to generate the final recoded input strand. Note that once again by our 'choice' of primer pairs in these steps, we can ensure that only a properly rewired product is produced.

Rewiring output from multiple FG's into a single input template will be done in multiple sequential recodings. The process is the same as for a single recoding except for the final amplification of the recoded template. In this case, the product of the recoding is itself a recoding strand. To accomplish this, the 5'-CABC\*A\*B\*-3' intermediate product is amplified with 5'-C-3' and 5'-C\*Br-3' primers to generate a product of the structure 5'-CABC\*-3'. During this amplification we must also change the nature of the recoder so that it can recode to a new channel (i.e. change it from a C\*-A to a C\*-B recoder). To do this, we synthesize the 5'-C\*Br-3' primer such that the value of C\* is the same as the value of the new target channel of that particular rewiring strand. Thus if we want to change a C\*-A recoder to a C\*-B

recoder, we only use 5'-C\*Br-3' primers that have the same value at the B and C\* positions. The products of this reaction will have the structure 5'-CABC\*-3', with the value of the A channel correctly recoded to the output from the C\* channel of the first gate, and the value of the B and C\* regions will be the same so that the new recoder can act as a C\*-B recoder for a second step. At this point, the input to FG<sub>i+1</sub> is partially built. Since this output was designed to be a recoder, it is used, by the same process, to recode the output from another gate operating at step i to generate the input for a FG at step i+1. This multiple rewiring functions without operator knowledge of input strands or desired output strands. All that is required is knowledge of the elements in the wiring diagram.

For each type of recoder (i.e. C\*-A), there are eight possible recoder strands representing all permutations of the three recoded channels. The FG output strand at step i is recoded using each of the eight recoders in separate reactions, much like the FG operation. In this instance, four of the recoders will have the corresponding value at the C\* channel, generating a positive reaction for those four recodings. To then recode the output from a second FG at step i to the B channel, we use the eight outputs from the first recoding (four positive, four negative) for the second recoding. The four negative channels will have no recoder DNA in them, and again generate a negative result. Of the four positive channels, two will also contain the correct value of C\* (which now corresponds to the value of B) and will generate positive results, while the other two will have the incorrect value of C\* and will be negative. A third recoding to the C channel from a third FG output at step i will only be positive for one of the two positive outputs from the second recoding, and will represent the recoded input template for the next FG (i+1 step) based on the output of the three previous gates. Note that in this operation we do not pool the output until the end of the third recoding operation.

FGs are logically reversible in that there is a 1:1 mapping of the output to the input. We showed here that the original input is recovered by simply using the DNA output of the FG as the input for the series of FGs that defines the reverse steps. The DNA melting and reannealing steps are reversible (21) as is the polymerization step,  $DNA_n + dNTP \rightarrow DNA_{n+1} + PP_i$  where its microscopic reverse is pyrophosphorolysis,  $DNA_{n+1} + PP_i \rightarrow DNA_n + dNTP$  (22). However, the operations required to return the output to the original *logical* state do not necessarily return the system to the identical initial *thermodynamic* state. However, in a fashion similar to Bennett's enzymatic Brownian Turing machine example (3, 10), the system can be run as close to equilibrium as desired. The temperature could be set close to the T<sub>m</sub> of the primers and templates and the concentrations of the dNTPs and PP<sub>i</sub> set close to the equilibrium value (22), and the free energy of the system would approach a minimal value, albeit at the cost of a correspondingly slow rate of computation. Figure 4 shows PCR without thermocycling. Ultimately, the

system is not reversible in the sense that if the reaction were to run back to the starting conditions, we would be left with nucleotides and no template, hence the forward reaction could not proceed.

If we relinquish the criteria for minimal energy dissipation, how fast could the logic gates run in this system? At concentrations of substrates typical for the PCR, the half time for the annealing step is approximately 3 sec (21) and polymerization rate is approximately 15 bases per second (22). In principle, by maximizing the rates of the PCR on the 60 base pair input strands, one PCR event could take less than ten seconds. Realistically, however, rewiring and set up steps for additional PCR reactions when done manually adds time to the complete computation. However, it may be possible to carry out these reactions on extremely large numbers of DNA input strands. While we have demonstrated how to send input templates through a FG, the true power of DNA is the potential to perform computations massively in parallel. Repositioner strands could accomplish this by transferring the value of the C channel, and a unique identity address region on the template, to the new template. Currently, however, parallel computing with DNA is subject to the so called word design problem, i.e., limitations in generating a library of unique templates large enough to be interesting while retaining the property of unambiguous PCR priming.

The fundamental strategy of the processes that we developed to implement a biomolecular FG are strikingly similar to those that already exist in nature. For example, the fate of regions on either side of the control bit is determined by whether an element is properly bound to a regulatory region, reminiscent of certain aspects of the control of gene regulation (23). Likewise, the repositioning steps are another way of accomplishing a recombination event. Furthermore, a positioner strand can be designed so that the process can easily be generalized to allow the results of the computation to be recoded into the sequence of a functional gene which can subsequently be expressed, creating a direct correspondence between computation and biological activity. The engineering aspects of the system remain to be characterized, including the details of the energy/time tradeoff and how many, and how complex, a system of circuits can be built with these biomolecular elements of reversible logic (BERLs). The extent to which BERLs correspond to, or can simulate, the logical circuits of replicating systems will depend on the results of these investigations.

#### References

1. R. Landauer, *Science* 272, 1914 (1996).
2. R. Landauer, *IBM J. Res. Dev.* 5, 183 (1961).
3. C. H. Bennett, *IBM J. Res. Dev.* 17, 525 (1973).
4. E. Fredkin and T. Toffoli, *Int. J. Theor. Phys.* 21, 219 (1982).
5. J. Milburn, *Phys Rev. Lett.* 62, 2124 (1989).
6. H. Chang, *Int. J. Theor. Phys.* 21, 955 (1982).

7. K.K. Likharev, *Intl. J. Theor. Phys.* 21, 311 (1982); K. K. Likharev and A.N. Korotkov, *Science* 273, 763 (1996).
8. S. Younis and T.F. Knight, Proc. *Symp. on Interg. Syst. MIT Press*. 234 (1993)
9. D. Deutsch, *Proc. R. Soc. London Ser. A* 400, 97 (1985); S. Lloyd, *Science*, 273, 1073 (1996); A.O. Orlov, I. Amlani, G.H. Bernsein, C.S. Lent, G.L. Snider, *Science*, 277, 928 (1997); N.A. Gershenfeld and I.L. Chuang, *Science*, 275, 350 (1997); D.G. Cory, A.F. Fahmy, T.F. Havel, Proc. Natl. Acad. Sci. USA, 1634 (1997).
10. C.H. Bennett and R. Landauer, *Sci. Amer.* 253, 48 (1985).
11. R.P. Feynman, *Feynman Lectures on Computation*. Anthony J.G. Hey and Robin W. Allen, Eds. (Addison-Wesley, Reading, MA, 1996).
12. C.H. Bennett, *Int. J. Theoret. Phys.* 21, 905 (1982).
13. A. Arkin and J. Ross, *Biophys J* 67, 560 (1994); A. Hjelmfelt, E.D. Weinberger, J. Ross, *Proc Natl Acad Sci U S A* 88, 10983 (1991).
14. L.M. Adleman, *Science* 266, 1021 (1994).
15. R. J. Lipton, *Science* 268, 542 (1995).
16. Q. Ouyang, P.D. Kaplan, S. Liu, A. Libchaber, *Science* 278, 446 (1997).
17. T.H. Leete, M.D. Schwartz, R.M. Williams, D.H. Wood, J.S. Salem, H. Rubin, *Proceedings of the 2nd DIMACS Workshop on DNA Based Computers*, Princeton NJ. 49 (1996)
18. H. Rubin, *Nat Struct Biol.* 3, 656 (1996).
19. *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers*, Philadelphia, PA 1997.
20. F. Guarnieri, M. Fliss, C. Bancroft, *Science* 273, 220 (1996).
21. D. Poland and H. Scheraga, *Theory of Helix-Coil Transitions in Biopolymers* (Academic Press, New York, 1970); J. G. Wetmur, *Crit Reviews in Biochem and Molec. Biol.* 26, 227 (1991).
22. M.E. Dahlberg and S.J. Benkovic, *Biochemistry* 30, 4835 (1991).
23. C-H. Yuh, H. Bolouri, E.H. Davidson, *Science* 279, 1896 (1998).
24. This work was funded by a grant from the NSF # 98-SC-NSF-1007