Chemistry & Biology
Article

Biologically Relevant Molecular Transducer with Increased Computing Power and Iterative Abilities

Tamar Ratner,[1] Ron Piran,[1,2] Natasha Jonoska,[3] and Ehud Keinan[1,2,*]
[1]Schulich Faculty of Chemistry, Technion - Israel Institute of Technology, Technion City, Haifa 32000, Israel
[2]Department of Molecular Biology and the Skaggs Institute for Chemical Biology, The Scripps Research Institute, 10550 North Torrey Pines Road, La Jolla, CA 92037, USA
[3]Department of Mathematics, University of South Florida, Tampa, FL 33620, USA
*Correspondence: keinan@tx.technion.ac.il
http://dx.doi.org/10.1016/j.chembiol.2013.02.016

SUMMARY

As computing devices, which process data and interconvert information, transducers can encode new information and use their output for subsequent computing, offering high computational power that may be equivalent to a universal Turing machine. We report on an experimental DNA-based molecular transducer that computes iteratively and produces biologically relevant outputs. As a proof of concept, the transducer accomplished division of numbers by 3. The iterative power was demonstrated by a recursive application on an obtained output. This device reads plasmids as input and processes the information according to a predetermined algorithm, which is represented by molecular software. The device writes new information on the plasmid using hardware that comprises DNA-manipulating enzymes. The computation produces dual output: a quotient, represented by newly encoded DNA, and a remainder, represented by *E. coli* phenotypes. This device algorithmically manipulates genetic codes.

INTRODUCTION

The inherent molecular nature of any biomolecular computing device implies the advantage of direct interaction with biological systems. The ability to algorithmically control, manipulate, and reorganize biomolecular processes in vivo opens up new opportunities, not only with respect to biologically relevant communications, but also in creating programmable devices that can carry out different computational operations. Therefore, the value of a molecular computing device can be examined in terms of both its ability to be programmed to carry out sets of different arithmetic or logical operations and its ability to directly intervene with biological systems. For example, DNA-based computing devices, which enjoy the advantages of miniaturization and natural interaction with biomolecules, may directly impact biological systems (Livstone et al., 2003; Ratner and Keinan, 2009; Shoshani et al., 2011a; Ratner et al., 2012). Numerous architectures of biomolecular computing devices have been developed and proved experimentally utilizing ingenious biomolecular techniques (Lipton, 1995; Liu et al., 2000; Sakamoto et al., 2000; Faulhammer et al., 2000; Braich et al., 2002; Roweis et al., 1998; Winfree et al., 1998; LaBean et al., 2000; Winfree, 2000; Benenson et al., 2001; Mao et al., 2000; Rose et al., 2002; Komiya et al., 2000; Rothemund et al., 2004; Stojanovic and Stefanovic, 2003; Krishnan and Simmel, 2011; Adleman, 1994). Each of these studies has elegantly demonstrated some parts of the above-mentioned potential. Here, we further these efforts by the experimental construction of an advanced molecular computing machine, a molecular transducer. This machine can compute iteratively, i.e., take as input its own output, and thereby can exhibit the desired criteria for self-contained biomolecular processor.

Transducers are capable of information processing and interconversion of different types of information (Hopcroft et al., 2001). These computing devices, which model ubiquitous processes in both the living and inanimate worlds, may solve a broad spectrum of different problems. Moreover, the transducer output can serve as input for subsequent computing by the same or another transducer. This iterative computational process provides computational power that has been shown to be equivalent to a universal Turing machine (Hopcroft et al., 2001). Therefore, the experimental realization of a molecular transducer can be considered as a significant advancement in the evolution of prominent architectures of molecular computing.

A transducer is a finite-state machine that at each computational step progresses stepwise along an input tape, reads an input symbol, changes its internal state, and also writes a new symbol on the tape (Hopcroft et al., 2001). Formally, the device can be described as a graph with vertices, each representing an internal state, and arrows, representing the transitions between the states (Figure 1). Each arrow (edge) is labeled with a pair of characters that, for the transition indicated by the arrow, represents the current symbol read (the input symbol) and the symbol written on the tape (the output symbol).

Here, we report on the experimental realization of a DNA-based molecular transducer that performs long division by 3 on binary strings, it computes iteratively, and also produces a biologically relevant output. The input binary string of this device is encoded on a DNA plasmid. The device reads and processes the plasmid and writes new information by altering its sequence.

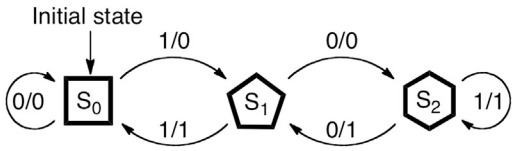

Figure 1. A Transducer Model with Three Internal States, S_0—The Initial State—S_1, and S_2, and Input and Output Symbols, 0 and 1
This machine performs long division of a binary number by 3. In each computational step, an arrow labeled *a/b* means that, if the device reads an input symbol *a* while being at state in the source of the arrow, it replaces the symbol *a* by an output symbol *b* and switches to the state in the target of the arrow.

Finally, the internal terminal state of the transducer computation is realized in the form of a bacterial phenotype upon transformation.

RESULTS AND DISCUSSION

The Logic Device and Its Molecular Realization

Our design incorporates ideas from previously reported molecular finite-state automata (Benenson et al., 2001, 2004; Kossoy et al., 2007; Shoshani et al., 2011b) and the theoretical description of a molecular Turing machine (Rothemund, 1996). The transducer applicability is demonstrated here through a long division of binary numbers by 3. This problem allows for two types of output: a quotient of the division and a remainder, which can be 0, 1, or 2. A transducer with three states, S_0, S_1, and S_2, can perform the task of long division by 3 with each state encoding for one of the remainder possibilities (Figure 1). This computation requires decoding, encoding, and storage of information. Unlike the previously reported finite automata, which employed linear double-stranded DNA (dsDNA) inputs, in our implementation the transducer operates on a plasmid, allowing for both reading of the input symbol and writing of the output symbol (Ratner and Keinan, 2009; Shoshani et al., 2011a; Ratner et al., 2012).

We represented each of the input and output binary symbols by a 6 bp string of dsDNA (Figure 2A). The inputs of the transducer contained recognition sites for restriction enzymes BpmI, FokI, and a sequence encoding the string of symbols (Figure 2C). Each of these inputs was cloned into pTR plasmid (see Experimental Procedures and Figure S2 available online). The "software" of the transducer (transition rules) consisted of a set of short dsDNA molecules, named transition molecules (TMs) (Figure 2D). The "hardware" was composed of restriction enzymes, ligase, and a set of dsDNA molecules, named detection molecules (DMs) (Figures 2B and 2E). Successive molecular operations on the plasmid included reading the input symbol in one direction and simultaneously writing the output in the other direction (Figure 3). This strategy required double restriction and double ligation in each read/write cycle. The output of the computation, i.e., the binary representation of the quotient, was encoded on the plasmid as newly written DNA symbols, whereas *E. coli* phenotypes exhibit the remainders. The iterative computational power was demonstrated by a recursive application of the transducer on an obtained output plasmid (Figure 4).

The Molecular Computing Process

The computation was carried out by alternating exposure of the input plasmid to two mixtures: a restriction mixture containing three endonucleases, FokI, BpmI, and BseRI, and a ligation mixture containing T4 DNA ligase, ATP, TMs, and DMs. For the task of double restriction, we employed FokI as the primary computational cutter and BpmI as the secondary cutter. FokI cleaves downstream of its recognition site to produce a 4-base sticky end, whereas BpmI cleaves upstream of its recognition site to produce a 2-base sticky end (Figure 2B). The third restriction enzyme, BseRI, was employed to reduce background noise by digesting all singly cleaved plasmids.

The FokI restriction represents "reading" an input symbol and determining the new internal state. Each of the 6 bp sequences representing an input symbol, as well as the terminator sequence, could be cleaved by FokI in one of three possible modes. The first mode involves cleavage at the beginning of the symbol sequence, producing a single-stranded overhang of the first four nucleotides. The second and third modes involve cleavage of the sequence either 1 bp deeper or 2 bp deeper into that domain, leaving a single-stranded overhang of either the second through fifth or third through sixth nucleotides, respectively. These modes represent reading of an input symbol while the transducer is at one of its three internal states, S_0, S_1, and S_2, respectively (Figure 2A). The transition between the transducer's states corresponds to the restriction mode and is dictated by the distance between the recognition site of FokI and the input symbol sequence, as instructed by the newly incorporated TM (Figure 2D).

Our design also included the formation of a degenerate sticky end, CC, in all restrictions by BpmI. The inclusion of CC at the end of every symbol allowed for continuous writing of the output symbols on the plasmid without spacers, thus qualifying the output sequence as a new input (Figures 2A and 2D). The design of 4-base and 2-base sticky ends on each of the TMs and DMs also provided advantageous kinetic preference by first hybridizing the 4-base end and then closing the plasmid at the 2-base end.

Each of the three DMs contained a specific reporter gene, resistance to ampicillin, tetracycline, or kanamycin, correlated with one of the three possible terminal states S_0, S_1, and S_2, respectively (Figure 2E). Each DM could hybridize selectively to a different sticky end produced by a differently cleaved terminator sequence, identifying one of the three states, S_0, S_1, and S_2, and hence the remainder of the division. Thus, the output signal was manifested by specific antibiotic resistance upon transformation to bacteria.

The transducer was implemented by processing the input plasmid via repetitive cycles of restriction and ligation to produce the final-state output in the form of a modified cyclic plasmid. While the previously reported finite-state automata employed all components in a single pot (Benenson et al., 2001; Soreni et al., 2005; Kossoy et al., 2007), the transducer required two separate mixtures. This practice of alternate exposure to the restriction and ligation mixtures offered considerable advantages for the robustness of the system. First, all TMs, being substrates of BpmI and BseRI, are incompatible with these enzymes. Moreover, although all TMs are too short to be restricted by FokI they could still act as its competitive inhibitors. Second, segregation

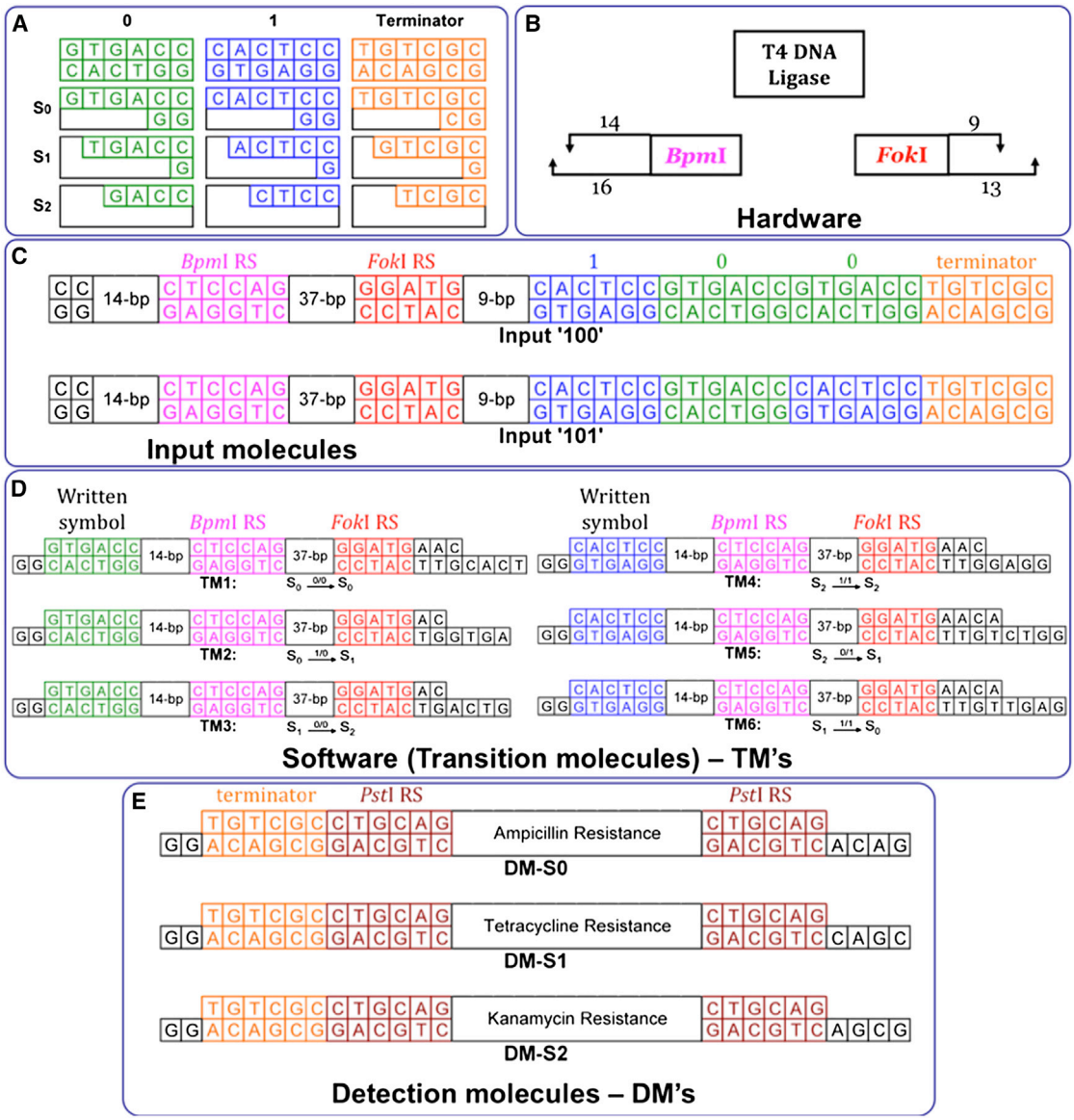

Figure 2. The Long Division by Three DNA-Based Transducer Components

(A) Input symbol 1, 0, and terminator and the identity of sticky ends produced upon specific restriction, which represents state S_0, S_1, or S_2.

(B) The hardware, including two restriction enzymes and T4 DNA ligase.

(C) Two input molecules containing a sequence of symbols 0 (green), 1 (blue), terminator (orange), and recognition sites of the computing enzymes: BpmI (pink) and FokI (red). The recognition site of BseRI (not shown) is located between these two sites.

(D) Six transition molecules representing the transition rules, each containing a written symbol, either 0 or 1, and recognition sites of BpmI and FokI. The recognition site of BseRI (not shown) is located between these two sites.

(E) Three detection molecules, each containing a written terminator (orange), reporter gene, and restriction site of PstI (brown).

See also Figure S2.

of the two mixtures prevents computing mistakes that could occur by premature restriction/ligation. Third, since restriction enzymes and T4 DNA ligase require different reaction conditions, segregating their buffers assures their optimal efficiency. Fourth, this practice allowed for removal of the cleaved fragments and side products, which could compete with the TMs. Finally, this separation offered convenient bacterial amplification of the intermediate cyclic plasmids. In order to monitor each computation step, we ligated the TMs and DMs in two separate steps and examined the resultant plasmids for antibiotic resistance (Figure S1 available online).

Mathematical Output and Biological Outcome

The computation with the transducer was demonstrated by dividing three integers 4, 5, and 6 written in their binary format, 100, 101, and 110 (Figure 4). The numbers were represented by DNA sequences on the corresponding input plasmids named pTR-In100, pTR-In101, or pTR-In110, respectively (Figure S2).

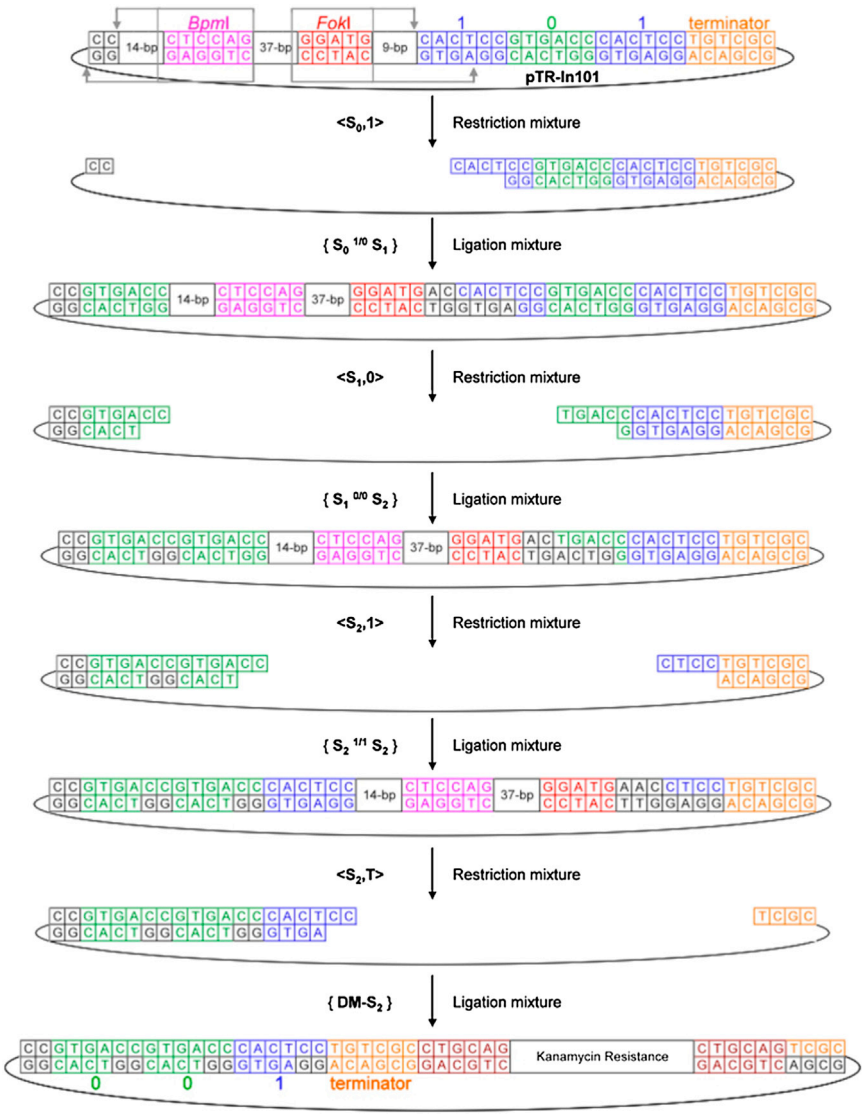

Figure 3. Long Division of 5—Binary 101—by 3

The symbol 0 (green), symbol 1 (blue), and terminator (orange) recognition sites of BpmI (pink) and FokI (red). The recognition site of BseRI is located inside the 37 bp spacer. The restriction mixture included BpmI, BseRI, and FokI. The ligation mixture included T4 DNA ligase, ATP, transition, and detection molecules. For details of the computation steps, see Figure S3. This division resulted in the integer 001, which was detected by DNA sequencing, and a remainder of 2, represented by S_2, which was detected by the resistance of *E. coli* expressing the plasmid to kanamycin. See also Figure S3.

The division of 5 (binary 101) by 3 resulted in quotient 1 (binary 001) and a remainder 2. The latter was represented by a plasmid encoding S_2 as the final state of the computation, which was manifested by *E. coli* resistance to kanamycin (Figures 3, 4, and 5).

Similarly, the division of 4 (binary 100) by 3 resulted in the quotient 1 (binary 001) and a remainder 1, represented by S_1 as the final state of computation, manifested by *E. coli* resistance to tetracycline (Figures 4, 5, and S3A). Likewise, the division of 6 (binary 110) by 3 resulted in the quotient 2 (binary 010) and a remainder 0, represented by a plasmid encoding the final state S_0, manifested by *E. coli* resistance to ampicillin (Figures 4, 5, and S3B).

With all three computations, we sampled at least ten *E. coli* colonies that survived on a specific antibiotic and found all of them to possess the expected computation plasmid and correct resistance gene. The very few colonies that survived on incorrect media (1.0% ± 0.8%) were found to be either contamination by bacteria lacking the computation plasmid or the result of incorrectly incorporated detection molecule. Remarkably, bacteria with incorrect detection molecule still possessed the correct quotient sequence. Furthermore, examination of several hundred plasmids at all intermediate computation steps did not reveal even a single case of erroneous computation, indicating very high fidelity (Figure 6A). Unprocessed (degenerate) input plasmids were discovered at levels of 0%–30% (Figure 6), reflecting limited enzymatic efficiency. Nevertheless, these limitations were irrelevant to the computational process and fidelity because the output was amplified by the bacterial colonies.

Iterative Computing

To demonstrate recursive computing, we used the output from the primary computing with input 101 as a new input for a secondary computing process. The plasmid bearing the output sequence 001 needed technical adjustments to make it suitable for the consecutive process. First, we removed the antibiotic resistance gene. Second, we cloned into the plasmid a restriction cassette that contained new recognition sites for BpmI, FokI, and BseRI at a location dictating S_0 as the initial state (Figure S4A). The next computing with this modified input, pTR-In001, resulted in the expected quotient 0 and remainder 1, represented by a plasmid encoding the final state S_1, which was manifested by the resistance of *E. coli* to tetracycline (Figures 4, 5, and S4B).

The above-described version of the transducer requires human intervention. This stepwise approach allowed us to monitor the products and yields in each step, determine potential errors, etc. Yet, as described in the Supplemental Information, two recognition sites for single cutter restriction enzymes, BamHI and NcoI, were inserted into the input plasmid away from the computation region (Figure S2A). This cloning site allows for introduction of a biotinylated DNA cassette that could be used for automatic separation with avidin-coated magnetic beads (Figure S2D). This technology, together with the alternate

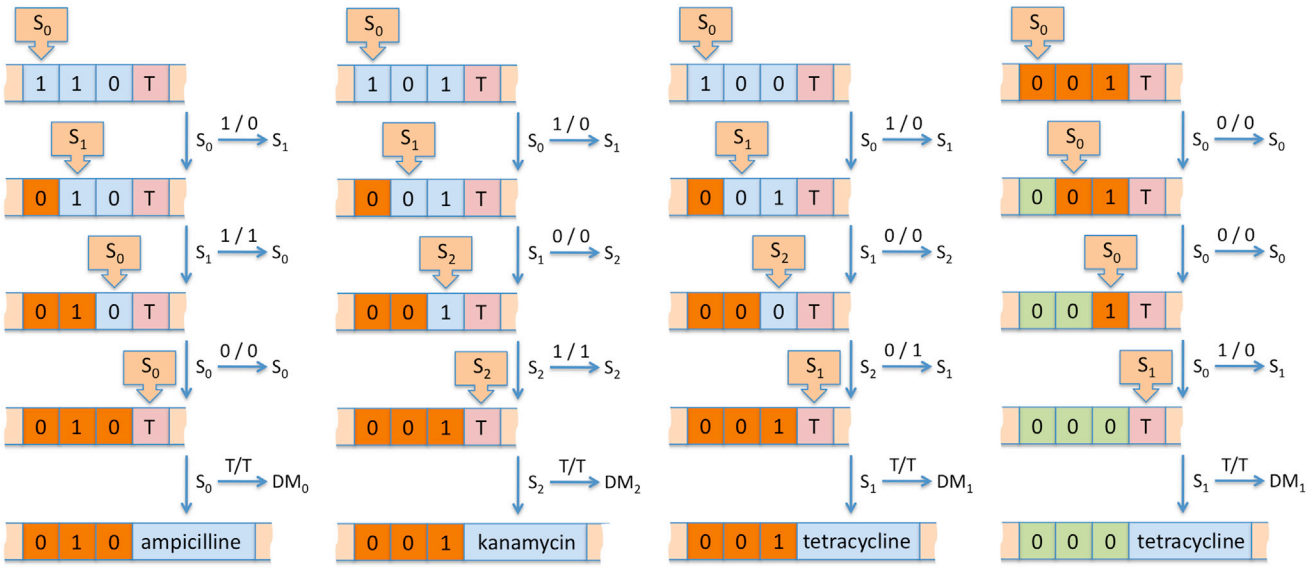

Figure 4. A Schematic Representation of Four Division Processes by the DNA-Based Transducer: 6 ÷ 3; 5 ÷ 3; 4 ÷ 3, and 1 ÷ 3
The latter represents an iterative computing performed with the output 001 that was obtained from the division 5 ÷ 3. Light orange: the computing head with its internal state; light blue and dark orange: input and output symbols in the first three operations, respectively; dark orange and green: input and output symbols in the iterative computing, respectively. See also Figure S4.

exposure to the restriction and the ligation solutions, which can be automatized without human intervention, offers scale-up opportunities.

SIGNIFICANCE

This report describes an experimental realization of a molecular transducer that not only computes iteratively, but also produces biologically relevant results. The above-described DNA-based transducer offers multiple benefits, such as the ability to read and transform genetic information, miniaturization to the molecular scale, and the aptitude to produce computational results, which interact directly with living organisms. Although this transducer was employed to solve a specific problem of a long division by 3, the general methodology shows that similar devices could be applied for other computational problems. For example, recursive division by a certain number is equivalent to root extract. Furthermore, the transducer's capabilities to compute iteratively and cooperate with other transducers offer attractive opportunities. For example: repetitive employment of several basic transducers, each can divide a given input number by a prime number, would allow for many combinations of these basic transducers. This strategy enables a general division by a broad variety of numbers without requiring a specific molecular device for each number. Moreover, repetitive employment of several transducers can solve more difficult problems, up to the level that require a universal Turing machine (Hopcroft et al., 2001). Therefore, their implementation on a genetic material may not just evaluate and detect specific sequences, but it can also alter and algorithmically process the genetic code according to a predesigned formula. This possibility opens up avenues for new approaches in a variety of methods in biotechnology such as individual gene therapies and cloning.

EXPERIMENTAL PROCEDURES

Strains and Media

DH5α *E. coli* strain was grown aerobically at 37°C in Luria-Bertani broth or agar plates supplemented with variable antibiotics: ampicillin, 100 μg/ml (Sigma), kanamycin, 10 μg/ml (Sigma), tetracycline, 10 μg/ml (Sigma), or chloramphenicol, 30 μg/ml (Fluka).

General Molecular Biology Techniques

DNA was separated by gel electrophoresis using 1% agarose for routine use (Sigma) and purified by the QIAquick Gel Extraction Kit (QIAGEN). DNAs were amplified by PCR using Red Load Taq Master mix (LAROVA GmbH) or GoTaq Green Master Mix (Promega). Mutageneses were preformed using the QuikChange Site-Directed Mutagenesis kit (Stratagene). *E. coli* transformations were performed using the calcium-chloride transformation method. DNA sequencing were performed at the Genomic Technologies Unit, Faculty of Medicine, Technion - Israel Institute of Technology. General procedures for DNA amplification, cloning, transformation, and agarose gel electrophoreses were carried out as described previously (Sambrook et al., 1989). All restriction enzymes, T4 DNA Ligase and buffers were purchased from New England Biolabs.

Synthetic DNA

All deoxyoligonucleotides were custom ordered from Syntezza Bioscience. All synthetic single-stranded DNA (ssDNA) contained hydroxyl groups on both 3′ and 5′ ends. Oligonucleotides that were used for site-directed mutagenesis and computation components were PAGE-purified by the supplier; otherwise they were desalted. All oligonucleotides were used without further purification. All sequences are provided in Table S1.

Transition Molecules, Input 101, and "Restriction-Cassette" dsDNA

All synthetic dsDNA molecules were prepared by annealing 100 pmol of commercially obtained deoxyoligonucleotides, 50 pmol of each ssDNA, in

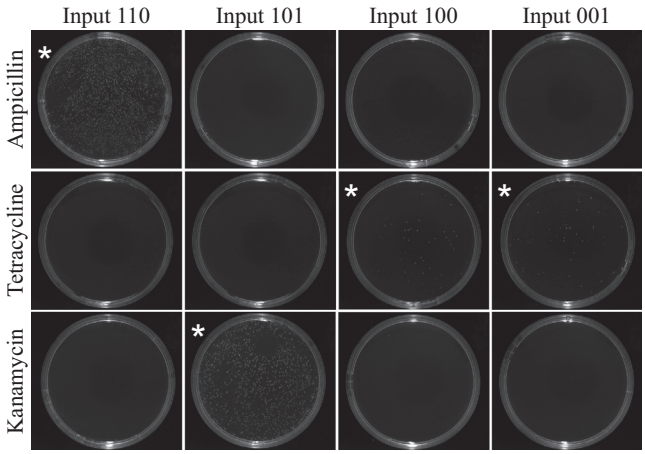

Figure 5. Biologically Relevant Output Signals
Computation with inputs 110, 101, and 100 led to bacterial colonies resistant to ampicillin (S_0), kanamycin (S_2) and tetracycline (S_1), respectively (marked with white asterisks). Iterative computing with input 001, which was obtained as the output from computing with 101, resulted in tetracycline resistance (S_1). See also Figure S1.

a final volume of 10 μl of distilled deionized water (DDW). The annealing was performed by heating the solution to 95°C followed by slow cooling to room temperature. The synthetic molecules included the sense and antisense strings, constructing the TMs, input molecule 101, and the restriction cassette.

Detection Molecules

The carrier vector of the output-detecting molecule for state S_0, was formed by two mutations on a circular pGEM vector (Promega). The first mutation was preformed at positions 2370–1. It inserted a synthetic adaptor of 15 bp, containing PstI and BsmBI recognition sites and a unique sequence that served (after restriction with BsmBI) as a 4-base sticky end. The second mutation was preformed at position 1217. It inserted a synthetic adaptor of 28 bp, containing PstI and BseRI recognition sites, a sequence that served (after restriction with BseRI) as a 2-base sticky end and a terminator sequence. The above-described mutations generated the plasmid pGEM-D_0.

The carrier vectors of the output-detecting molecules for states S_1 and S_2 were constructed by PCR amplification of the genes encoding for tetracycline and kanamycin resistance, which were obtained from pBR322 and pUG6, respectively. The amplification primers enabled incorporation of the following: (1) PstI and BsmBI recognition sites together with a unique sequence that served as a 4-base sticky end at the C terminus of the gene. This sticky end resulted from restriction with BsmBI, and (2) PstI and BseRI recognition sites, a sequence that served (after restriction with BseRI) as a 2-base sticky end, and a terminator sequence at the N terminus. Each of the PCR products was ligated to a pGEM-T Easy Vector (Promega) to generate pGEM-D_1 and pGEM-D_2.

In addition to the artificially designed BsmBI recognition site, pGEM-D_2 contained a recognition site, originally at position 920–925 in pUG6. The BsmBI recognition site was eliminated without altering the amino acid encoding sequence (arginine and leucine) at that kanamycin resistance gene, based on the degeneration of the genetic code, and maintaining the common genetic code in *E. coli*.

The three carrier vectors, pGEM-D_0, pGEM-D_1, and pGEM-D_2, which contained the various detection molecules, DM_0, DM_1, and DM_2, were digested separately by successive exposure to BsmBI and BseRI to form the three desired DMs having unique sticky ends.

Preparation of Computing Plasmids

The pJIR480 vector (Sloan et al., 1992) was mutated twice to form pTR. The first mutation altered the FokI recognition site at position 4211–4215 to an

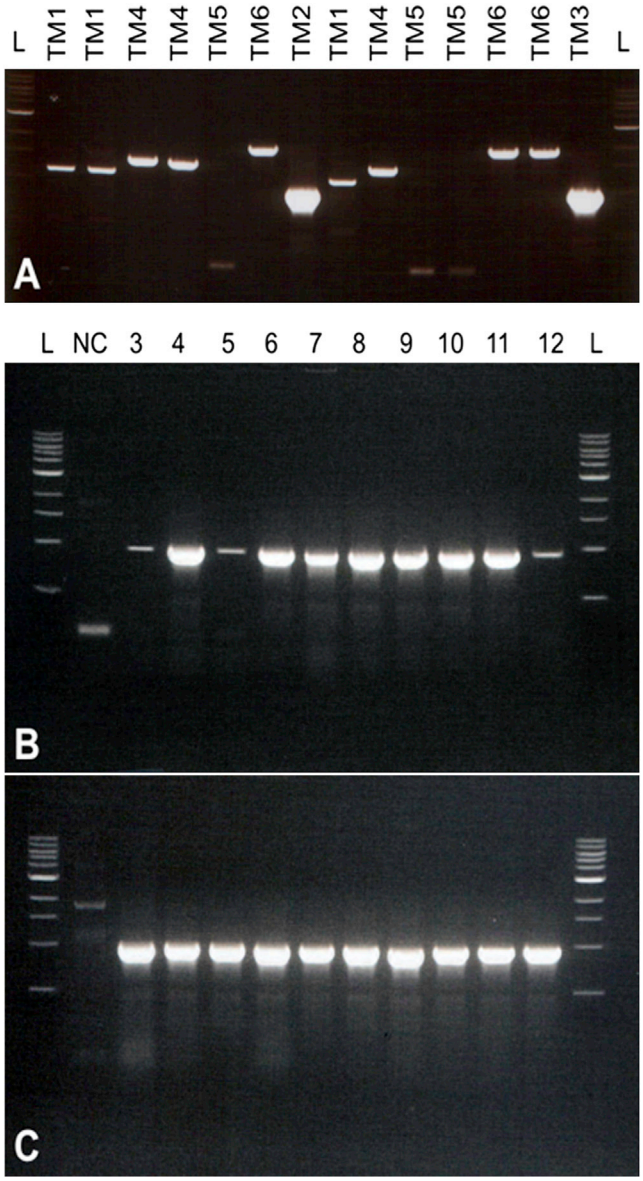

Figure 6. Identification of which TM Was Cloned into the Computation Plasmid
After each computation step, the resulting plasmids were amplified to identify which TM was cloned. The amplification was done using a general primer and one of six specific primers, each unique to a single TM.
(A) Scanning of the resulting plasmid from the second computation step with input 100. Lanes 1 and 16, ladder; lanes 2–15 exhibit PCR products of amplification with primer homology to one of the TMs (indicated above the gel image). Each amplification reaction was done on ten plasmid lines. The expected TM to be ligated is TM3. The TM ligated at the previous computation step was TM2. This electrophoresis gel demonstrated that none of the unexpected TMs (TM1, TM4–6) were cloned into the computation plasmid. A specific amplification is seen at lane 8, indicating degenerated plasmids, and at lane 15, indicating ligation to the expected MT3.
(B and C) PCR products of amplification with specific primers homologous to the TMs expected to ligate at the examined computation steps. Each amplification reaction was done on a single plasmid line. Lanes 1 and 13, ladder. Gel electrophoresis B indicates 70% efficiency of the ligation reaction. Gel electrophoresis C indicates 100% efficiency of the ligation reaction.
See also Table S1.

NcoI site. The second mutation inserted a BamHI recognition site at position 4665–4667. These changes were carried out in order to delete the FokI recognition site and to enable the insertion of other cassettes (using BamHI and NcoI) for future studies.

Linearization of pTR was achieved by simultaneous cleavage of pTR with KpnI and SacI. Input 101 was inserted into pTR to form pTR-In101 using standard molecular biology techniques. An appropriate plasmid, containing the input sequence, was detected by PCR with AMP-In-S and AMP-In-AS primers, and sequenced. A new recognition site for BseRI was added by mutagenesis at position 3482 between the BpmI and FokI sites. This procedure resulted in the insertion of a synthetic adaptor of 16 bp, containing a BseRI recognition site and a 10 bp spacer. Site-directed mutagenesis was used to convert pTR-In101 to pTR-In100 by exchange of the second "1" symbol into a "0" symbol. Similarly, pTR-In100 was converted to pTR-In110 by exchange of the first "0" symbol into a "1" symbol. All final products were sequenced.

Stepwise Computing

The computation process was carried out by sequential exposure of the computation vector to three reaction mixtures (Figure S1). The restriction was done by first adding BpmI with buffer 4 and BSA in DDW at 37°C for 55 min, then adding BseRI followed by incubation at 37°C for 5 min, and finally adding FokI followed by incubation at 37°C for another 5 min. The enzymes were then inactivated by heating to 65°C for 20 min and the products were purified by gel electrophoresis. The second treatment was done with a ligation mixture containing all six TMs (or DMs), T4 DNA ligase, and T4 DNA ligase buffer in DDW. The products were transformed into competent cells and grown on a selective agar medium containing chloramphenicol (or, when DMs were employed, ampicillin, kanamycin, or tetracycline). If specific resistance to one of the latter three was observed, the computing process was terminated. The expected cloning results were confirmed by PCR using the AMP-In-AS primer and exchanged primers characteristic to each TM and sequenced.

Consecutive Computation

Deletion of the antibiotic resistance gene was carried out by restriction of the output vector pTR-001 by PstI followed by self-ligation. The product was transformed into competent cells and grown on a selective agar medium containing chloramphenicol. The grown colonies were duplicated on agar medium containing kanamycin. DNA from colonies that grew on chloramphenicol but could not survive on kanamycin was produced and sequenced. The product was cleaved using XhoI endonuclease and the restriction cassette was inserted to form the modified pTR-In001. The latter was transformed into competent cells and grown on selective agar medium containing chloramphenicol. The cloning results were confirmed by PCR using AMP-In-AS and AMPRestCassette primers, and sequenced.

SUPPLEMENTAL INFORMATION

Supplemental Information includes four figures and one table and can be found with this article online at http://dx.doi.org/10.1016/j.chembiol.2013.02.016.

ACKNOWLEDGMENTS

This study was supported by the National Science Foundation under grant no. 0523928. N.J. is supported in part by NSF grants CCF-1117254 and DMS-0900671. E.K. thanks the US-Israel Binational Science Foundation (BSF), the Russell Berrie Nanotechnology Institute, and the Institute of Catalysis Science and Technology, Technion. E.K. is incumbent of the Benno Gitter & Ilana Ben-Ami Chair of Biotechnology, Technion.

Received: November 27, 2012
Revised: February 5, 2013
Accepted: February 7, 2013
Published: May 23, 2013

REFERENCES

Adleman, L.M. (1994). Molecular computation of solutions to combinatorial problems. Science *266*, 1021–1024.

Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., and Shapiro, E. (2001). Programmable and autonomous computing machine made of biomolecules. Nature *414*, 430–434.

Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., and Shapiro, E. (2004). An autonomous molecular computer for logical control of gene expression. Nature *429*, 423–429.

Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W., and Adleman, L. (2002). Solution of a 20-variable 3-SAT problem on a DNA computer. Science *296*, 499–502.

Faulhammer, D., Cukras, A.R., Lipton, R.J., and Landweber, L.F. (2000). Molecular computation: RNA solutions to chess problems. Proc. Natl. Acad. Sci. USA *97*, 1385–1389.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001). Introduction to Automata Theory, Languages, and Computation (Boston: Addison-Wesley).

Komiya, K., Sakamoto, K., Gouzu, H., Yokoyama, S., Arita, M., Nishikawa, A., and Hagiya, M. (2000). Successive State Transitions with I/O Interface by Molecules. In 6th international workshop on DNA-based computers, DNA 2000, A. Condon and G. Rozenberg, eds. (Berlin: Springer-Verlag), pp. 19–26.

Kossoy, E., Lavid, N., Soreni-Harari, M., Shoham, Y., and Keinan, E. (2007). A programmable biomolecular computing machine with bacterial phenotype output. ChemBioChem *8*, 1255–1260.

Krishnan, Y., and Simmel, F.C. (2011). Nucleic acid based molecular devices. Angew. Chem. Int. Ed. Engl. *50*, 3124–3156.

LaBean, T.H., Winfree, E., and Reif, J.H. (2000). Experimental progress in computational by self-assembly of DNA tilings. In Proceedings of DNA Based Computers, V.E. Winfree and D.K. Gifford, eds. (Providence RI, Cambridge, MA: American Mathematical Society), pp. 123–140.

Lipton, R.J. (1995). DNA solution of hard computational problems. Science *268*, 542–545.

Liu, Q., Wang, L., Frutos, A.G., Condon, A.E., Corn, R.M., and Smith, L.M. (2000). DNA computing on surfaces. Nature *403*, 175–179.

Livstone, M.S., van Noort, D., and Landweber, L.F. (2003). Molecular computing revisited: a Moore's Law? Trends Biotechnol. *21*, 98–101.

Mao, C., LaBean, T.H., Relf, J.H., and Seeman, N.C. (2000). Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. Nature *407*, 493–496.

Ratner, T., and Keinan, E. (2009). Programmable DNA-Based Finite Automata. In DNA-Based Algorithmic Bioprocesses, A.A. Condon, D. Harel, J.N. Kok, and N.C. Series, eds. (Berlin: Springer-Verlag), pp. 505–516.

Ratner, T., Shoshani, S., Piran, R., and Keinan, E. (2012). Biologically relevant molecular finite automata. In Biomolecular Information Processing, E. Katz, ed. (Weinheim, Germany: Wiley-VCH), pp. 145–179.

Rose, J.A., Deaton, R.J., Hagiya, M., and Suyama, A. (2002). Equilibrium analysis of the efficiency of an autonomous molecular computer. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. *65*, 021910.

Rothemund, P.W.K. (1996). A DNA and restriction enzyme implementation of Turing machines. In DNA Based Computers, Proceedings of a DIMACS Workshop, R.J. Lipton and E.B. Baum, eds. (Princeton, NJ: American Mathematical Society), pp. 75–119.

Rothemund, P.W.K., Papadakis, N., and Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biol. *2*, e424. http://dx.doi.org/10.1371/journal.pbio.0020424.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K., and Adleman, L.M. (1998). A sticker-based model for DNA computation. J. Comput. Biol. *5*, 615–629.

Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., and Hagiya, M. (2000). Molecular computation by DNA hairpin formation. Science *288*, 1223–1226.

Sambrook, J., Fritsch, E.F., and Maniatis, T. (1989). Molecular Cloning: A Laboratory Manual (Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press).

Shoshani, S., Ratner, T., Piran, R., and Keinan, E. (2011a). Biologically relevant molecular finite automata. Isr. J. Chem. *51*, 67–86.

Shoshani, S., Wolf, S., and Keinan, E. (2011b). Molecular computing with plant cell phenotype serving as quality controlled output. Mol. Biosyst. *7*, 1113–1120.

Sloan, J., Warner, T.A., Scott, P.T., Bannam, T.L., Berryman, D.I., and Rood, J.I. (1992). Construction of a sequenced Clostridium perfringens-Escherichia coli shuttle plasmid. Plasmid *27*, 207–219.

Soreni, M., Yogev, S., Kossoy, E., Shoham, Y., and Keinan, E. (2005). Parallel biomolecular computation on surfaces with advanced finite automata. J. Am. Chem. Soc. *127*, 3935–3943.

Stojanovic, M.N., and Stefanovic, D.A. (2003). A deoxyribozyme-based molecular automaton. Nat. Biotechnol. *21*, 1069–1074.

Winfree, E. (2000). Algorithmic self-assembly of DNA: theoretical motivations and 2D assembly experiments. J. Biomol. Struct. Dyn. *17*(*Suppl 1*), 263–270.

Winfree, E., Liu, F., Wenzler, L.A., and Seeman, N.C. (1998). Design and self-assembly of two-dimensional DNA crystals. Nature *394*, 539–544.

