A simple DNA gate motif for synthesizing large-scale circuits

Lulu Qian and Erik Winfree

Computer Science, Computation & Neural Systems, and Bioengineering
California Institute of Technology, Pasadena, CA 91125, USA

Abstract. The prospects of programming molecular systems to perform complex autonomous tasks has motivated research into the design of synthetic biochemical circuits. Of particular interest to us are cell-free nucleic acid systems that exploit non-covalent hybridization and strand displacement reactions to create cascades that implement digital and analog circuits. To date, circuits involving at most tens of gates have been demonstrated experimentally. Here, we propose a DNA catalytic gate architecture that appears suitable for practical synthesis of large-scale circuits involving possibly thousands of gates.

1 Introduction

DNA-based catalysts [1–3] and logic gates [4–6] have been proposed as general-purpose components for synthesizing chemical circuits [7, 8, 6] with applications in medical therapeutics, nanotechnology, and embedded control of chemical reactions. Progress in this direction will depend upon three future advances: (1) Developing input/output interfaces between the DNA circuits and biomedically relevant molecules [9], DNA nanomachines [10, 11], and general chemistries [12, 13]; (2) Developing DNA circuit construction techniques that scale up so that large and interesting circuits can be systematically created; and (3) Extending the DNA programming methodology beyond well-mixed solutions to include spatial structures at the molecular [14–16] and macroscopic scales [17–19].

In this paper we focus on the second challenge. We introduce a DNA catalytic gate motif suitable for scaling up to large circuits, along with an abstract circuit formalism that aids the design and understanding of circuit behavior. To illustrate the potential of this approach, we show how to implement arbitrary feedforward digital logic circuits, arbitrary relay circuits, and analog circuits exhibiting a variety of temporal dynamics. Thanks to the modular design of the gate motif, sequence design is straightforward. Furthermore, we argue that synthesis and preparation of circuit components can be parallel and scalable. Our estimates suggest that circuits involving thousands of distinct gates may be possible.

2 A simple catalytic gate with a threshold

Each catalytic gate may be represented abstractly as a node with one or more wires connected to the left and right sides (Fig. 1a). Each wire will correspond to a single-stranded DNA molecule with a specific sequence, called a signal strand, which may be absent (an inactive wire) or present at some concentration (an active wire). Each node will correspond to a gate strand that may bind to or release signal strands on demand. As described below, an active wire on one side of a gate can catalyze the the exchange of activity on wires on the other side. Furthermore, there may be a threshold that must be exceeded before catalysis occurs. While the catalytic gates are intrinsically symmetric, we will typically configure them asymmetrically, with an input side and an output side. When connected into circuits involving many interacting catalytic gates, complex circuit behavior can be obtained.

The DNA implementation of the elementary gate illustrated in Fig. 1a is shown in Fig. 1bc. The three wires correspond to free signal strands S_2TS_3 ("fuel"), S_1TS_3 ("output"), and S_3TS_4 ("input"), while the node indicates complexes involving the gate base strand $T'S_3'T'$. The state of the network is indicated by writing the amounts of each species in appropriate locations: the (relative) concentrations of signals on the wires, and the (relative) concentrations of gate:signal complexes within the nodes at the ends of the corresponding wires. The concentration of a threshold complex is written as a negative number in the location for the corresponding gate:signal complex for the signal it is absorbing. Thus, Fig. 1a specifies the

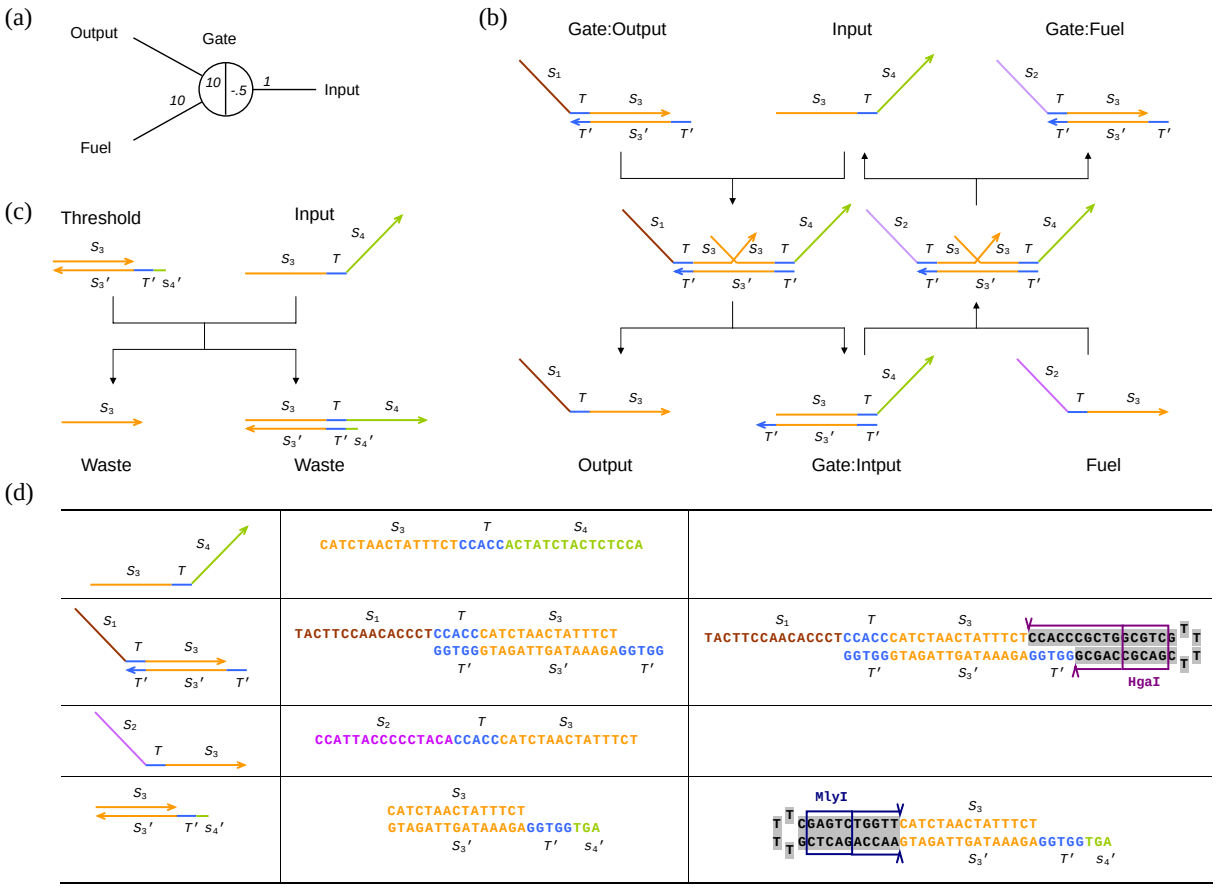

Fig. 1. The DNA motif for catalytic "seesaw" gates. **(a)** Abstract gate diagram. **(b)** The DNA gate motif and reaction mechanism. S_1, S_2, S_3, S_4 are the recognition domains; T is the toehold domain; T' is the Watson-Crick complement of T, etc. Arrowheads mark the $3'$ ends of strands. Black lines indicate elementary reactions via alternating strand displacement and toehold exchange. All reactions are reversible and unbiased; arrowheads indicate the dominant flows for the initial concentrations shown in (a). **(c)** The threshold motif and reaction mechanism. The toehold is extended by three bases (s_4', the complement of the first three $5'$ bases of S_4), providing an increased rate constant relative to the gate itself. **(d)** Example sequences. Gate complexes and signal molecules are shown in their operational form (second column). Hairpin precursor molecules (third column) are cleaved by restriction enzymes to create functional gates. Recognition domain sequences are 15 nt and the toehold domain sequence is 5 nt.

gate:output complex, fuel signal strand, and input signal strand of Fig. 1b with respective concentrations $10x$, $10x$, and $1x$; the threshold gate absorbing the input strand, shown in Fig. 1c, is present at $0.5x$; where $1x$ is a standard concentration, perhaps 30 nM. With these initial concentrations, the input strand will first overcome the threshold and then act catalytically to facilitate the equilibration of the output strand and the fuel strand to approximately $5.1x$ each. (This level can be estimated by noting that by symmetry the two wires will have similar activity, and that the total concentration on each wire and within the gate remains constant. Detailed formulas are provided in section 3.)

The reaction mechanism is a simplified version of the entropy-driven catalytic gate introduced in [3, 20]. Fig. 1b shows reactions between the three signal strands and the gate complexes involving those signal strands. Fig. 1c shows the threshold gate that absorbs the input strand. Example sequences are shown in Fig. 1d. The fundamental operation is toehold exchange, a toehold-mediated strand displacement reaction that results in a free right-side signal strand replacing a bound left-side signal strand, or visa-versa. For example, input signal strand S_3TS_4 can bind to gate:output complex $T'S_3'T'$:S_1TS_3 via toehold domain T,

unbiased three-strand branch migration within the S_3 recognition domain will lead to a state where either input strand S_3TS_4 or output strand S_1TS_3 is bound by no more than the short toehold domain T, and that signal strand will quickly fall off. (For simplicity, we ignore the abortive attempts, and only consider those reactions in which strand replacement occurs.) At this point, the gate base strand has its left toehold revealed, and its right toehold hidden: toehold exchange has been achieved, yielding the release of an output strand and the production of a gate:input complex. An analogous process allows the fuel strand to similarly displace the input strand from the new gate:input complex, completing a catalytic cycle that has the net effect of exchanging one left-side signal strand in solution (the fuel) for another left-side signal strand (the output). Note, however, that a free left-side signal strand (e.g. the fuel) cannot directly replace a bound left-side signal strand (e.g. the output bound to the gate). It can only do so in the presence of a right-side signal strand (e.g. the input) by an indirect sequence of events: the right-side signal strand displaces the bound left-side signal strand, and in turn the other left-side signal strand displaces the now-bound right-side strand. Thus, the right-side signal strand has catalyzed the exchange of the two left-side signal strands, which cannot exchange in its absence. This back-and-forth motion is the inspiration for our name for this motif: "seesaw gates".

In the above discussion, recognition domains S_1, S_2, and S_4 played no active role in the mechanism. Rather, these domains allow the signal strands to interact with other gates and thus dictate connectivity within a circuit. For example, output strand S_1TS_3 could also serve as a right-side signal strand for another gate with base strand $T'S_1'T'$, not shown here. However, while bound to base strand $T'S_3'T'$, signal strand S_1TS_3 cannot act as a catalyst for the downstream gate because its toehold domain is sequestered. Of course, if it is released into solution by a strand displacement reaction, then it is free to act upon its target gate $T'S_1'T'$. In summary, each gate base strand consists of a single recognition domain identifying the gate, flanked by two toehold domains, only one of which is exposed at any given time; while each signal strand consists of two recognition domains identifying the two gates it connects, separated by a central toehold domain that is sequestered and thus inert when the signal strand is bound to a gate base strand.

In addition to catalysis, threshold behavior can be helpful for circuit function, for example by cleaning up after leaky reactions. Fig. 1c shows a threshold gate that serves as a competitive inhibitor of the signal strand S_3TS_4. Because of the slightly longer toehold on the threshold gate, the signal strand will react faster with the threshold gate than with the original gate complex. (Toehold-mediated strand displacement reaction rates depend exponentially on toehold length, for short toeholds [21]. In this paper, we conservatively assume a 10-fold speed-up specific to the targeted signal strand, which is less than the ratio between the maximal rate constant for DNA hybridization and the measured rate constant for 5 nt toeholds [21]. We neglect that in gates with more than one input, there will be some crosstalk: a threshold gate for one input will also react with the other signal strand, although without the 10-fold speed-up.) Once the signal strand has reacted with threshold gate, it will never be released because all relevant toeholds are sequestered – effectively, inert waste has been produced. Only after all the threshold gates have been used up, then the remaining signal strands can react (perhaps catalytically) with the original gate.

This consistent and modular motif makes systematic construction of circuits logically straightforward, as discussed below, and further makes laboratory procedures for synthesizing gates and circuits plausible to carry out on a large scale. A substantial difficulty with our previous work [6, 3] was that each gate substrate was a complex of multiple strands that had to be separately annealed together, and each complex had to be purified to remove excess single-stranded species and malformed gate substrates. Here, we aim to simultaneously prepare all gate complexes together in a single test tube; to do so, we must ensure that different gate species do not interact, and that the strands needed to form a given gate complex find each other efficiently. For our solution, we draw inspiration from the observation [22, 23] that mixtures of hairpin molecules, when annealed, are likely to form non-interacting intramolecular hairpins even if at room temperature there exist lower free energy states involving intermolecular complexes. This occurs because the intramolecular hairpins are typically stable at some moderately high temperature, above the melting temperature of the intermolecular complexes – thus, during annealing, the hairpins form first and become kinetically trapped. The implication for gates is that if each gate species can be synthesized initially as a hairpin precursor, annealing all such gate precursors in a single reaction will result in a high yield of properly formed non-interacting molecules. Fig. 1d shows our realization of this scheme. After annealing, incubation with appropriate restriction enzymes removes the now-undesired linker subsequence, resulting in a well-formed complex of two strands. The entire solution could be purified by gel, since all gates are the same size; all threshold gates could be purified

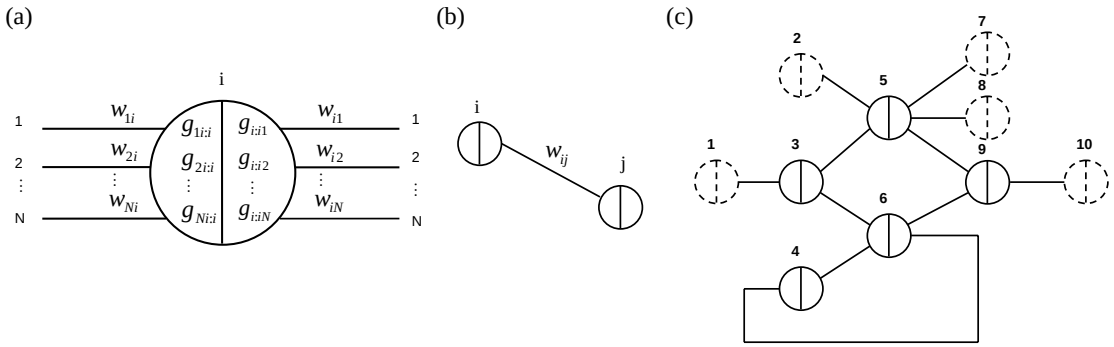

Fig. 2. Abstract diagrams for seesaw gate circuits. **(a)** The general form of a gate node. Each gate may be connected to many wires on each side, potentially all N nodes in the network, including itself. **(b)** The general form of a wire. Each wire is specifically connected on its left end to the right side of a gate node, and connected on its right end to the left side of a gate node. **(c)** An example circuit with five real gates, five virtual gates, and eleven wires. Each wire is identified by the two gates it connects; thus the virtual gates serve to provide full names to their incident wires.

similarly. One way or another, making circuit function robust to sloppy parallel synthesis methods will be crucial to scaling up existing DNA circuits to hundreds or thousands of gates.

3 Abstract circuit formalism and function

The abstract network representation introduced in Fig. 1a facilitates concise reasoning about circuits involving many interacting catalytic gates. In general, a circuit consists of a number of gate nodes and a number of wires between gate nodes. Each wire connects exactly two gates (Fig. 2b). Each gate consists of a left side and a right side, and it may connect to any number of wires on each side (Fig. 2a). Virtual gates (dotted lines) are gates whose total concentration is zero; they are used solely as syntactic sugar to provide a consistent naming scheme for input and output wires. When it is necessary to make the distinction, gates with non-zero concentration are referred to as real gates (solid lines). A seesaw gate circuit then consists of a number of gate nodes connected by wires between their left and right sides (Fig. 2c).

The implementation of any such circuit diagram using DNA molecules is straightforward. (1) For a circuit with N gates, begin by chosing a set of N sufficiently distinct m-mer sequences $S_1, S_2, \ldots, S_N$. Sequence length m must increase (approximately logarithmically) with N; the value $m = 15$ was used in Fig. 1. Use the same univeral toehold sequence T in all cases. (2) To each gate i, associate the gate base strand $T'S_iT'$. To each wire ij, associate the wire signal strand S_iTS_j. This determines the sequences for all molecules in the circuit. (3) Based on the desired initial concentrations of each signal strand, gate complex, and threshold, synthesize the appropriate hairpin molecules, cleave them with restriction enzymes, and purify the desired complexes. (4) To run the system, add the appropriate input signal strands and observe (e.g. by fluorescence) the target output signal strands.

For standard nomenclature, a wire between gates i and j, with gate i on the left, is called w_{ij}, representing the free signal strand S_iTS_j. Likewise, a gate i base strand $T'S_i'T'$ bound to the left end of wire ij signal strand S_iTS_j is referred to as $g_{i:ij}$; similarly $g_{ki:i}$ is used when the right end of signal strand S_kTS_i is bound to gate i. Note that $g_{i:ii}$ and $g_{ii:i}$ refer to distinct complexes: in the former case, the signal strand is bound on its left end, leaving the gate's left toehold accessible; while in the latter case the signal strand is bound on its right end, leaving the gate's right toehold accessible. The threshold complex for wire ij into gate j is called $th_{ij:j}$, and similarly for $th_{i:ij}$. In a slight abuse of notation, in the formulas below, w_{ij} refers both to the signal strand molecule itself and to the concentration of that species, and similarly for the gate and threshold complexes. Finally, note that although each gate has an identifying number corresponding to its base strand sequence, circuit diagrams may be drawn without any numerical annotation as the names are not relevant to circuit function.

Because all components are in a standard form, the set of chemical reactions modelling the toehold exchange steps and threshold absorption steps can be written concisely. For all $i, j, k \in \{1, 2, \ldots, N\}$,

$$w_{ji} + g_{i:ik} \quad \underset{k_0}{\overset{k_0}{\rightleftharpoons}} \quad g_{ji:i} + w_{ik} \tag{1}$$

$$w_{ij} + th_{ij:j} \overset{k_1}{\rightarrow} waste \overset{k_1}{\leftarrow} th_{i:ij} + w_{ij} \tag{2}$$

where the variables refer to the molecular species. Using standard mass action chemical kinetics, this gives rise to a system of ordinary differential equations (ODEs) for the dynamics. In the following, w_{ij} and similar terms refer to the concentrations of the respective species, rather than to the species themselves.

$$\frac{d\,w_{ij}}{dt} = k_0 \left(\sum_{n=1}^{N} w_{ni} \cdot g_{i:ij} + w_{jn} \cdot g_{ij:j} - w_{ij} \cdot g_{ni:i} - w_{ij} \cdot g_{j:jn} \right) - k_1 \left(w_{ij} \cdot th_{i:ij} + w_{ij} \cdot th_{ij:j} \right) \tag{3}$$

$$\frac{d\,g_{i:ij}}{dt} = k_0 \left(\sum_{n=1}^{N} w_{ij} \cdot g_{ni:i} - w_{ni} \cdot g_{i:ij} \right) \qquad \frac{d\,g_{ij:j}}{dt} = k_0 \left(\sum_{n=1}^{N} w_{ij} \cdot g_{j:jn} - w_{jn} \cdot g_{ij:j} \right) \tag{4}$$

$$\frac{d\,th_{i:ij}}{dt} = -k_1 \cdot w_{ij} \cdot th_{i:ij} \qquad \frac{d\,th_{ij:j}}{dt} = -k_1 \cdot w_{ij} \cdot th_{ij:j} \tag{5}$$

These dynamics have conserved quantities for each gate node i and for each signal wire ij:

$$\sum_{n=1}^{N} g_{ni:i} + g_{i:in} \overset{def}{=} c_i \qquad g_{i:ij} - th_{i:ij} + w_{ij} + g_{ij:j} - th_{ij:j} \overset{def}{=} c_{ij} \qquad \frac{d\,c_i}{dt} = \frac{d\,c_{ij}}{dt} = 0 \tag{6}$$

Furthermore, the equilibrium (if it is obtained) enforces a simple relationship between the free and bound forms of the signal strands, namely that their ratio with respect to a particular gate must be identical for all wires connected to that gate. This follows immediately from Eq. 1 and the detailed balance equation $k_0 \cdot w_{ji} \cdot g_{i:ik} = k_0 \cdot g_{ji:i} \cdot w_{ik}$. To wit, for each gate i, at equilibrium all wires achieve the ratio

$$\frac{w_{1i}}{g_{1i:i}} = \frac{w_{2i}}{g_{2i:i}} = \cdots = \frac{w_{Ni}}{g_{Ni:i}} = \frac{w_{i1}}{g_{i:i1}} = \frac{w_{i2}}{g_{i:i2}} = \cdots = \frac{w_{iN}}{g_{i:iN}} \overset{def}{=} r_i \tag{7}$$

As an example, we can calculate the equilibrium concentrations for a single real gate, i, connected only to virtual gates, as in Fig. 1a. The equilibrium wire concentration $w_{ij}(\infty)$ depends upon the gate ratio and the initial concentrations on that wire:

$$w_{ij}(\infty) = \frac{w_{ij}(\infty)}{g_{i:ij}(\infty) + w_{ij}(\infty)} (g_{i:ij}(\infty) + w_{ij}(\infty)) = \frac{r_i}{1 + r_i} (g_{i:ij}(0) + w_{ij}(0) - th_{i:ij}(0)) = \frac{r_i}{1 + r_i} c_{ij}$$

The equilibrium gate ratio r_i can be calculated from the conserved quantities of Eqs. 6 and the equilibrium constraints of Eq. 7:

$$c_{ij} = w_{ij}(1 + 1/r_i) \qquad c_i = (1/r_i) \sum_j w_{ji} + w_{ij}$$

from which it follows with a little algebra that

$$\frac{r_i}{1 + r_i} = 1 - \frac{c_i}{\sum_j c_{ji} + c_{ij}}.$$

For the concentrations given in Fig. 1a, we can work out $\frac{r_i}{1+r_i} = 1 - \frac{10}{10+10+1-0.5} \approx 0.51$, so at equilibrium $w_{13} = w_{23} \approx 10x \times 0.51 = 5.1x$, as claimed earlier.

The uniformity of components in seesaw gate circuits also facilitates their simulation. Using a general purpose mass action chemical kinetics simulator written by David Soloveichik in Mathematica, we have written routines for concisely representing, constructing, and simulating models of seesaw gate circuits.

4 Feedforward digital logic circuits

Digital logic has two compelling features for circuit construction: first, it has proven to be very expressive for the synthesis of a wide range of desired behaviors; and second, it is intrinsically robust to a variety of manufacturing and operational defects. The basic principle underlying digital logic is that an intrinsically analog signal carrier may be considered simply to be either ON or OFF if at each stage of computation, signals are either pushed toward the ideal ON value or pushed toward the ideal OFF value. This is called signal restoration, because if noise or device imperfections slightly corrupt a signal, that deviation from ideal behavior is cleaned up (perhaps not completely) by subsequent processing without altering the interpretation of the signal as ON or OFF. In this section we will consider a digital abstraction in which signal concentrations below $0.1x$ are considered OFF, while signal concentrations above $0.9x$ are considered ON. Intermediate signal levels – in the transition region – are considered a fault because proper behavior of logic gates is no longer guaranteed for input levels in the intermediate range.

Feedforward digital circuits are an important class of combinational circuits, i.e. memoryless circuits in which the inputs uniquely determine the outputs without need to re-use any parts of the circuit. It is well known that any boolean function can be computed (usually quite efficiently) by a well-designed feedforward circuit built from AND, OR, and NOT gates. Interestingly, with a small cost in circuit size and a minor change in representation, called "dual-rail logic", AND and OR by themselves suffice. We will therefore provide constructions for these basic operations, in addition to signal fan-in, fan-out, routing, and thresholding that may be required to paste the computing gates together into large circuits.

The OR gate (Fig. 3a) is a simple extension of the basic catalyst (Fig. 1a) in which there are two input wires, 13 and 23, that serve as catalysts for the release onto output wire 34, driven by exchange with the "power supply" wire 35. Under the assumption that the threshold gate reaction rate constant, k_1, is 10 times faster than the gate reaction rate constant k_0, clean digital behavior is seen in the model. For larger ratios k_1/k_0, the sharpness of the threshold increases correspondingly.

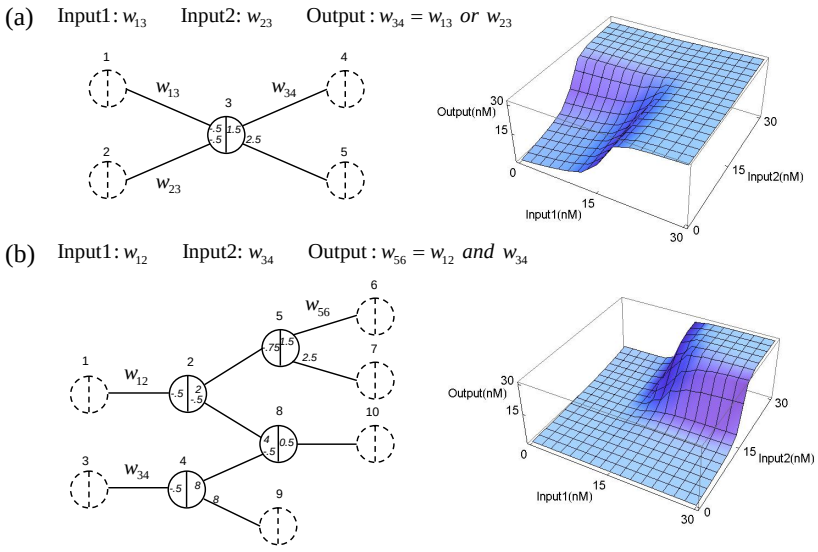

Fig. 3. The circuit diagram and input/output behavior of boolean logic gates. **(a)** A two-input OR gate. **(b)** A two-input AND gate. Simulations are performed with the reference concentration $1x = 30$ nM.

The AND gate is a little trickier. Essentially, the idea here is similar to what is done with transistors: control flow from a power supply using two regulatory gates in series, and only if both gates are active will the output be affected. However, in our case there is no actual flow, just a sloshing back and forth of the signals on the seesaw gates. (Recall that there is a constant total concentration of a given signal strand either free in solution or bound to the gate at either end.) That said, consider the circuit shown in Fig. 3b. First, if input signals 12 and/or 34 are OFF (i.e. between $0x$ and $0.1x$), then they will be absorbed by their respective thresholds. Now consider that if input 34 is ON (i.e. between $0.9x$ and $1x$), then it can catalyze the power supply 49 to push strands onto the wire 48. Gate 8 serves as a signal reflector, effectively routing a signal emanating from the right side of a gate to arrive at the right side of another gate. (Recall that wires can only connect left sides to right sides, so this cannot be achieved directly.) Once the input to gate 8 exceeds its threshold, the right-side catalyst will be released, in turn catalytically releasing strand 28 to impinge on gate 2. But, if input 12 remains OFF, there will be no catalysis and therefore no signal will be

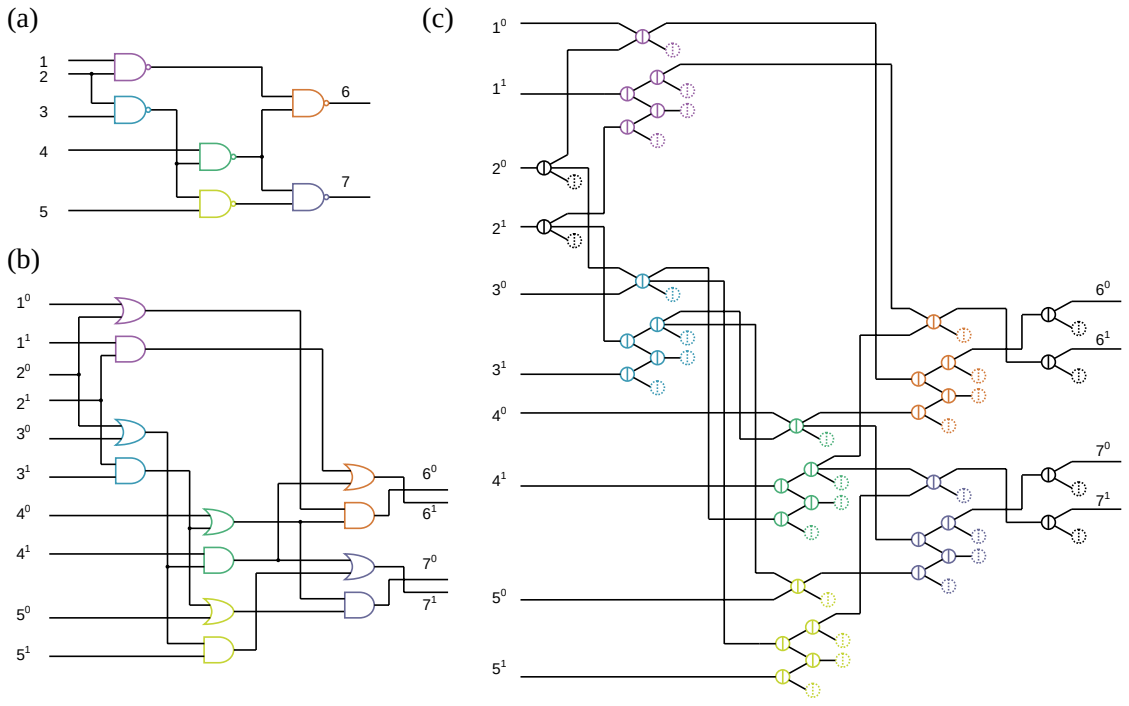

Fig. 4. Compiling boolean logic circuits. **(a)** A sample circuit with 6 gates. **(b)** Translation into an equivalent dual-rail circuit with 12 gates. **(c)** Translation into an equivalent seesaw gate circuit with 36 gates.

pushed onto wire 25. On the other hand, if input 12 is also ON, the signal on 28 will provide power to push signal onto 25, which will in turn exceed the threshold for gate 5 and catalyze the production of output 56. The final case we must consider is if input 34 is OFF but input 12 is ON. In this case, there is at most $0.5x$ input 12 remaining after absorption by the threshold; some of it will end up bound to gate 2 after having pushed some 25 onto the output wire; however, at most a total of $0.5x$ can be pushed onto 25, and this will be absorbed by the gate 5 threshold. Because the total signal on gate 2 must remain at $2x$, we see that all signal strands will remain bound to gate 2, the wires 12 and 25 will be empty, and no output 56 will be produced. This is of course considered OFF by the digital abstraction. Full simulations bearing out these intuitive considerations are shown.

Fan-in and fan-out are handled easily using these constructions. More than two inputs (additional fan-in) to an OR gate simply requires additional wires connecting to the seesaw gate node, and slight adjustments to the initial concentrations. More than one output (fan-out) from an OR gate is correspondingly simple: connect more output wires. Fan-out from an AND gate is analogous, but additional fan-in poses problems: a longer sequence of regulatory gates in series would require the initial power supply concentration to be considerably larger. Thus it is preferable to construct a multi-input AND gate as a binary tree of two-input AND gates.

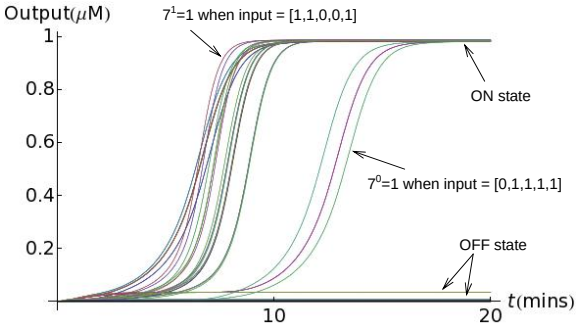

Fig. 5. Simulation results for all 32 possible input vectors. The concentrations of all four dual-rail output species are shown as a function of time. Delays vary with the input, depending the shortest decision path through the network.

Because NOT gates appear to be difficult to implement directly (although we have not yet outlawed the possibility), the dual-rail convention is used. The following procedure may be used to convert a single-wire digital logic circuit (that may use AND, OR, NOT, NAND, and NOR) into an equivalent dual-rail circuit (which uses only AND and OR), as illustrated in

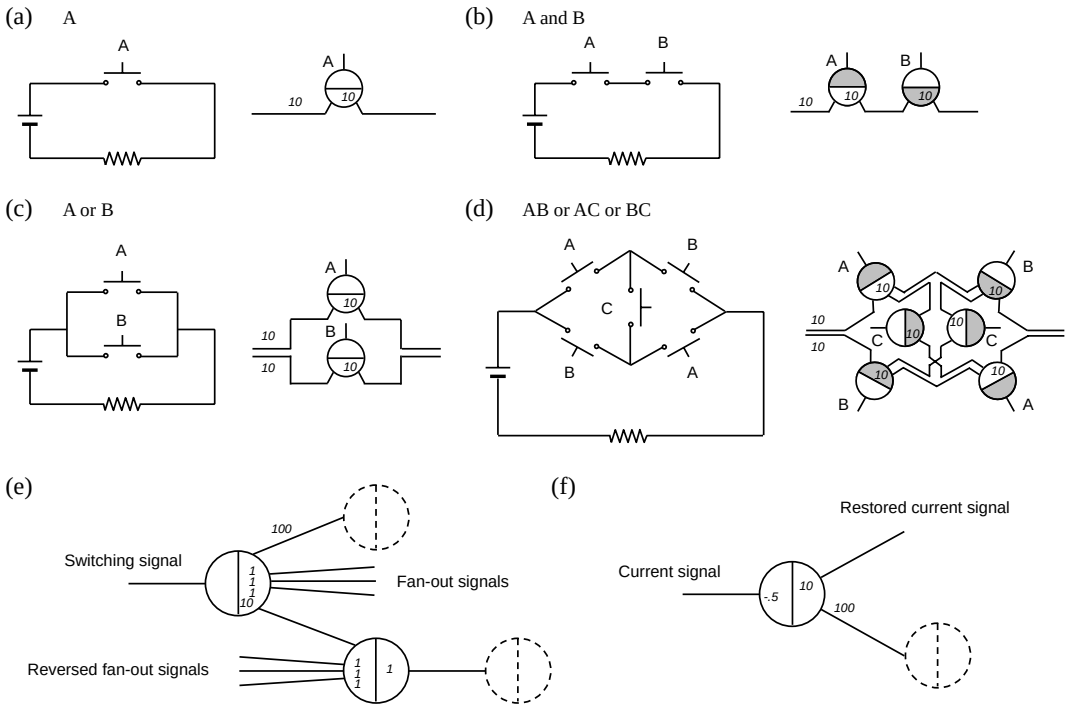

Fig. 6. Implementation of relay circuits. **(a)** A simple circuit with current source (battery) and controlled device (denoted by a resistor), and the corresponding seesaw gate circuit. **(b)** AND logic. The "left" side of each seesaw gate is shaded. **(c)** OR logic. **(c)** A more complex circuit. **(e)** Input signal fan-out for both left-side and right-side wires. **(f)** Restoration for output and intermediate signals.

Fig. 4ab. In the new circuit, which will contain at most twice as many gates, each wire z is replaced by two wires, z^0 and z^1. If neither new wire is ON, this indicates that the logical value of z has not been computed yet; if z^0 is ON, this indicates that the logical value of z must be OFF; while if z^1 is ON, this indicates that the logical value of z must be ON. (If both z^0 and z^1 are ON, then the circuit is experiencing a fault.) With this representation, each original two-input boolean logic gate can be implemented using one AND gate and one OR gate. XOR can be easily implemented in dual-rail logic as well, although it requires 4 OR gates and 2 AND gates. In terms of DNA molecules, the difficulty implementing NOT gates derives from the need to detect and act on the absence of an input strand, but the action must not take place before the computation has reached this point in the circuit. One possible solution could be to use a clock signal to initiate the action, or lack of it, at the appropriate time. This is effectively what dual-rail logic does, except that explicit clocks are unnecessary because every signal value (once computed) is represented by the presence of an appropriate molecule that, in itself, can serve as a timing signal as well.

To demonstrate these concepts, we compiled a small circuit (5 inputs, 6 NAND gates, 2 outputs) from its original netlist specification (Fig. 4a), to the equivalent dual-rail circuit consisting of only AND and OR gates (Fig. 4b), and then to its implementation as a network of seesaw gates (Fig. 4c). Because the thresholds on downstream gates drain the outputs of upstream gates and thus shift the equilibrium, for successful composition it was necessary to adjust the initial concentrations shown for isolated subcircuits in Fig. 3. It was also necessary to provide fan-out gates for inputs that were used more than once, and to add final read-out gates to properly drain the adjusted AND and OR subcircuits. The corresponding system of ODEs describing the network's mass action kinetics were then simulated for all 32 possible input combinations. (The compiler and simulator was written in Perl and Mathematica.) As shown in Fig. 5, in every case both outputs reached either a clear "0" or "1" concentration level, which was verified to be correct. This provides concrete evidence that the digital logic subcircuits compose well.

5 Relay contact circuits

In his seminal 1940 Master's Thesis [24], Claude Shannon established a systematic symbolic approach to the analysis and design of digital circuits. A prevalent technology at the time was relay contact circuits, in which input switching signals (A, B, C, ...) opened or closed electrical contacts in a network, either allowing current to flow through the network, or not. Like circuits made of AND, OR, and NOT gates, relay contact circuits can concisely implement arbitrary boolean functions. (Here we consider only circuits of primary relays – those directly controlled by external input signals.) To illustrate the flexibility of the seesaw gate motif, we provide a general method to compile relay contact circuits down to equivalent seesaw gate circuits.

The basic primitives for constructing relay contact networks are simple. A circuit with one switch (Fig. 6a) corresponds to a single seesaw gate. The current signal is represented by the free signal strands of the seesaw gate, and the switching signal is represented by the input catalyst strands. Signal is produced on the output wire if and only if the switching signal A is ON. Two switches in series perform an AND operation (Fig. 6b). Since the connectivity of seesaw gates is oriented, we make each wire always connect two different sides of the gates in the design. The two seesaw gates corresponding to the circuit "A and B" therefore must have different sides connected. Two switches in parallel perform an OR operation (Fig. 6c). The only innovation here is that two distinct input and output wires are used. Fan-out at the current source can provide such a signal, and an OR gate at the current drain can consolidate the output into a single wire, if desired.

For more complex circuits, such as the one in Fig. 6d, a number of additional issues arise. First, there may be two possibilities for the current direction through a given relay (such as C). We use a pair of seesaw gates in such cases. The current flow from A to C to A will go through the seesaw gate C on the left while the current flow from B to C to B will go through the other one on the right. This provides the basis for a general construction. Each original contact relay is replaced by a pair of seesaw gates, one prepared for each possible current flow direction. Output of each gate is routed to all other connected gates in the given flow direction. Thus, for each input setting, if there exists a connected path in the relay circuit, then there is a directed path of catalytically active gates in the seesaw gate circuit. Second, because we need different switching signal strands for different seesaw gates representing the same logical input (such as A, B, and C in Fig. 6d), and we also may need to connect them to either the "left" or the "right" side of the gate (as for C in Fig. 6d), we make use of a signal fan-out unit producing any number of signals in both the forward and reversed directions (Fig. 6e). Finally, if we use a uniform gate:output concentration (say $10x$), the current signal will decrease as it passes through each seesaw gate. Thus, we need to restore the current at any place the signal becomes weak. This can be achieved by an amplifier with a threshold (Fig. 6f). Following this procedure, any relay contact circuit can be implemented with seesaw gates.

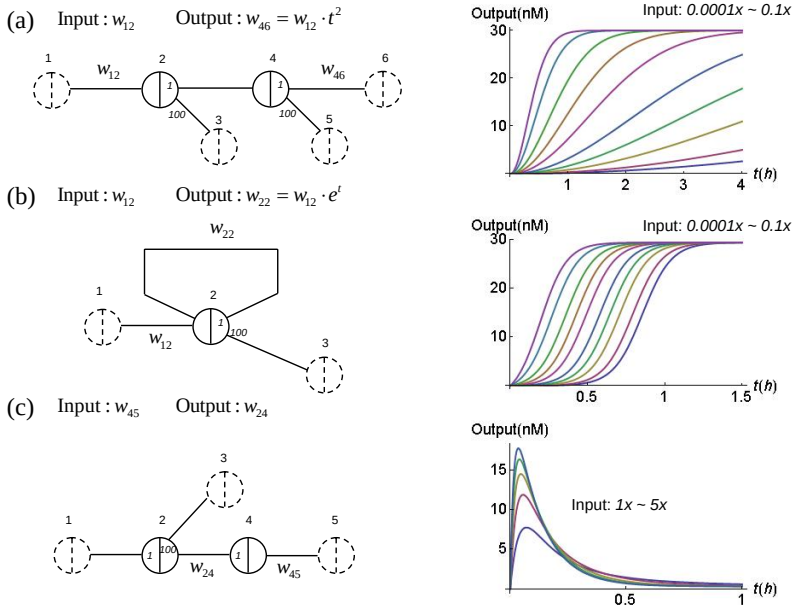

Fig. 7. Analog time-domain circuits. **(a)** A catalytic cascade that exhibits initially quadratic growth. Temporal trajectories are shown for a serious of exponentially decreasing initial input concentations. **(b)** A positive feedback circuit that exhibits initially exponential growth. A series of exponentially decreasing input concentrations yields a series of trajectories with linearly increasing half-completiong times. **(c)** A pulse-generating circuit. Pulse amplitude depends on the input concentration. Simulations use $1x = 30$ nM.

6 Analog time-domain circuits and feedback

The behavior of seesaw gate circuits is intrinsically analog. Following the approach of [3], we construct amplifier cascades with initial quadratic growth and with initial exponential growth. More complex temporal dynamics can also be synthesized, such as a pulse generator.

The amplifier shown in Fig. 7a is a two-stage feedforward cascade. Input signal 12 catalytically pushes strands onto wire 24, which exhibits linear growth with time. Signal 24 also serves as catalysts for the release onto output wire 46, which in turn grows quadratically with time.

The amplifier shown in Fig. 7b is a one-stage feedback cascade. Initially, input signal 12 catalytically releases strands onto output wire 22. However, signal strand 22 contains the recognition domain of gate 2 on both left side and right side, and therefore can play the role of the input signal once released, binding to the gate:output complex by its right side. Thus, strand 22 is also catalytically active in releasing additional copies of itself, resulting in exponential growth of the signal.

The pulse generator shown in Fig. 7c illustrates a non-amplifying temporal dynamic. The basic idea is that the input strand 45 initially releases a large amount of output 24, but this is slowly absorbed by gate 2. The large gate 2 concentration ensures that signal on wire 24 returns to a small value at equilibrium, because the gate-to-wire ratio for wire 24 must be the same as that for 23, which cannot be higher than $0.01x$.

7 Discussion

This project was inspired by the remarkable success of scaffolded DNA origami [15, 25, 26] for programming the self-assembly of hundreds of DNA strands into a single target structure. Self-assembly was extraordinarily reliable despite that DNA sequences could not be optimized to avoid undesired binding interactions and that unpurified DNA strands were used, for which the strand concentrations were not precisely known and many molecules were presumably incomplete or damaged. We were therefore looking for an analogous design for hybridization-based DNA catalysts and circuits that would require minimal sequence design effort and work well even with unpurified strands and unreliable concentrations. Does our proposed seesaw gate motif live up to hopes and expectations as a DNA circuit component suitable for scaling up to large and complex circuits? We see some encouraging features, some concerns, and some clear challenges.

First, design of large feedforward digital circuits looks promising. At the highest level, abstract specifications for circuit function can be expressed concisely using existing hardware description languages such as Verilog [27, 28] and VHDL [29], then compiled down to a gate level netlist specifying elementary gates (AND, OR, NOT, NOR, NAND, XOR) and their connectivity. Thus, the sheer complexity of large-scale circuit design can be managed by off-the-shelf tools. The next step is compiling the digital logic netlist down to the seesaw gate circuit abstraction, using the constructions described above for dual-rail logic. This is straightfoward if no circuit size optimizations are attempted. To achieve the final step of designing molecules, we must assign sequences to each gate base strand. For this purpose, a single large set of sufficiently distinct domain sequences would enable construction of any circuit containing up to the given number of seesaw gates.

Can we design sufficiently many domain sequences to scale up to circuits with many gates? There are two design criteria: (1) signal strands should not have secondary structure and should not interact with each other, and (2) strand displacement should be unable to proceed if the invading sequence is not an exact match. The first criterion can be satisfied by standard DNA sequence design methods [30, 31]; here we take the easy approach by using a three letter code (A, C, and T) for the signal strands, thus ensuring that problematic secondary structures and interactions are unlikely [32–34]. The gate base strand will therefore consist of (A, G, and T). A universal 5 nt toehold sequence may be used for all nodes, since specificity of interactions arises from uniqueness of the recognition domain sequences. Because of the complete independence of domains within the seesaw gate motif, no system-level conflicts are generated when strand sequences are generated by concatenation. The second criterion may be formalized as combinatorial sequence constraints. For example, we could require that at least 30% of bases are different for any two distinct recognition domains; as each mismatch impedes branch migration speed by a factor of roughly 10 [35, 36], even 5 mismatches will dramatically reduce crosstalk. Additionally, we require that mismatches are spread out, so that when the wrong signal strand interacts with a gate, it will quickly encounter difficulties and dissociate; specifically, the longest run of matches must be less than 35% of the domain length. Finally, to prevent synthesis errors and

ensure comparable melting temperatures, we require that there are no more than 4 A's or T's in a row, no more than 3 C's or G's in a row, and that sequences have between 30% and 70% GC-content. (Cf. constraints 1,7,8 of [37].) Using a "sphere-packing" technique [38], we have found sets of codes of sizes 76, 559, 5181, and $\gg 11,000$ for recognition domains of lengths 10, 15, 20, and 25, respectively, confirming the theoretically expected exponential growth in codebook size. This is enough to construct some interesting circuits.

Can so many distinct sequences be synthesized and handled in the laboratory? Promising synthesis techniques include the Agilent SurePrint inkjet microarray spotter, which is advertized to be capable of synthesizing roughly $250,000$ distinct sequences per slide [39, 40]; based on typical DNA array densities and slide dimensions, this corresponds to ≈ 0.040 fmol per spot. If each spot provides a $0.25x$ concentration (the minimal increment we use) in our chosen reaction volume, then a gate complex using $6x$ would require the use of 24 spots. If we define the complexity of a seesaw gate circuit to be the total initial concentration of strands, in $1x$ units, then the Agilent technology would allow us to create circuits of complexity up to $62,500x$. (This corresponds to either $\approx 15,000$ OR gates or ≈ 4500 AND gates for the concentrations used in Fig. 5.) NimbleGen has a similar technology, capable of producing $350,000$ distinct 85-mers [41], and we can expect technologies capable of synthesizing longer strands in the near future. The hairpins in Fig. 1d are up to 91 nucleotides long, which we consider to be synthesizable. After synthesis, linkers attaching strands to the slide can be cleaved, and a mixture of all strands can be collected in a single tube, annealed to form hairpins, digested with restriction enzymes to produce gates, and then gel-purified to eliminate non-functional molecules. Thus, all molecules for the entire circuit are synthesized and processed in parallel in a single tube.

Once designed and synthesized, will they work? The first question is speed. If the maximum total concentration for reliable DNA gate operation is 100 μM (perhaps optimistic), then $1x$ would be 1.6 nM for a $62,500x$ circuit, and the slowest reaction half-times (the effective gate delay) would be 3 days if $k_0 = 10^4$ /M/s and $k_1 = 10^6$ /M/s. This is too slow. Either one must resign oneself to smaller circuits, for which the concentrations can be higher, or one must find a way to speed up hybridization reactions in dilute complex mixtures. PERT [42, 43] claims to be such a method; in which case, $k_0 = 10^8$ and $k_1 = 10^{10}$ may be possible. This would reduce the 3 days to 30 seconds.

The second question is whether the computation will be correct. For feedforward digital circuits, thresholding and signal restoration – the digital abstraction – is expected to provide some robustness to variations in concentrations, to leak, and to minor crosstalk. However, experimental exploration of seesaw gate circuits will be needed to evaluate the potential for producing reliable function in practice.

In summary, we have proposed a new catalytic DNA gate that appears to be suitable for scaling up to analog and digital circuits incorporating thousands of gates with a reasonable expectation that adequate speed and reliability could be achieved.

However, a limitation of this work is that the circuit constructions we described all function by completely depleting key fuel species, hence each circuit preparation can be used only once. An important goal is to implement sequential circuits containing buffers, flip-flops, resets, and clocks that orchestrate the re-use of circuit elements and can process time-varying input signals. We do not know at this point whether this limitation is essential to the seesaw gate motif, or whether we have just not yet been clever enough to see how to do it. Similarly, we do not yet have a characterization of the class of analog dynamics that can be achieved, although it appears to be a rich space of behaviors. On the practical side, interfaces between DNA circuits and other chemical reactions will be necessary if DNA circuits are to serve as embedded controlers for molecular events.

Acknowledgements

The authors thank Dave Zhang for discussion of the catalytic mechanism, Marc Riedel for providing example netlists from logic synthesis benchmarks, Virgil Griffith for suggesting useful techniques for DNA sequence design, Ho-Lin Chen and Shuki Bruck for suggesting the connection to relay circuits, David Soloveichik for Mathematica code for simulating chemical reaction networks, and Georg Seelig, Bernard Yurke, and everyone else for discussions and support. This work has been supported by NSF grant no. 0728703 and HFSP award no. RGY0074/2006-C.

References

1. Jin Tang and Ronald R. Breaker. Rational design of allosteric ribozymes. *Chem. Biol.*, 4:453–459, 1997.
2. Andrew J. Turberfield, J. C. Mitchell, Bernard Yurke, Allen P. Mills, Jr., M. I. Blakey, and Friedrich C. Simmel. DNA fuel for free-running nanomachines. *Physical Review Letters*, 90(11):118102–1–4, 2003.
3. David Yu Zhang, Andrew J. Turberfield, Bernard Yurke, and Erik Winfree. Engineering entropy-driven reactions and networks catalyzed by DNA. *Science*, 318:1121–1125, 2007.
4. Milan N. Stojanovic, Tiffany Elizabeth Mitchell, and Darko Stefanovic. Deoxyribozyme-based logic gates. *Journal of the American Chemical Society*, 124:3555–3561, 2002.
5. Masami Hagiya, Satsuki Yaegashi, and Keiichiro Takahashi. Computing with hairpins and secondary structures of DNA. In Junghuei Chen, Natasha Jonoska, and Grzegorz Rozenberg, editors, *Nanotechnology: Science and Computation*, pages 293–308, Berlin Heidelberg, 2006. Springer-Verlag.
6. Georg Seelig, David Soloveichik, David Yu Zhang, and Erik Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314:1585–1588, 2006.
7. Robert Penchovsky and Ronald R. Breaker. Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes. *Nat. Biotechnol.*, 23(11):1424–1433, 2005.
8. Joanne Macdonald, Yang Li, Marko Sutovic, Harvey Lederman, Kiran Pendri, Wanhong Lu, Benjamin L. Andrews, Darko Stefanovic, and Milan N. Stojanovic. Medium scale integration of molecular logic gates in an automaton. *Nano Letters*, 6:2598–2603, 2006.
9. Ruslan Yashin, Sergei Rudchenko, and Milan N. Stojanovic. Networking particles over distance using oligonucleotide-based devices. *Journal of the American Chemical Society*, 129:15581–15583, 2007.
10. Nadrian C. Seeman. An overview of structural DNA nanotechnology. *Mol. Biotechnol.*, 37:246–257, 2007.
11. Jonathan Bath and Andrew J. Turberfield. DNA nanomachines. *Nature Nanotechnology*, 2:275–284, 2007.
12. Zev J. Gartner and David R. Liu. The generality of DNA-templated synthesis as a basis for evolving non-natural small molecules. *Journal of the American Chemical Society*, 123:6961–6963, 2001.
13. Kurt V. Gothelf and Thomas H. LaBean. DNA-programmed assembly of nanostructures. *Organic & Biomolecular Chemistry*, 3:4023–4037, 2005.
14. Erik Winfree, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–544, 1998.
15. Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.
16. Peng Yin, Harry M. T. Choi, Colby R. Calvert, and Niles A. Pierce. Programming biomolecular self-assembly pathways. *Nature*, 451:318–322, 2008.
17. Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society (part B)*, 237:37–72, 1953.
18. Anatol M. Zhabotinsky. A history of chemical oscillations and waves. *Chaos*, 4:379–386, 1991.
19. Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight, Jr., Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. Amorphous computing. *Communications of the ACM*, 43:74–82, 2000.
20. David Yu Zhang and Erik Winfree. Control of DNA strand displacement kinetics using toehold exchange. In preparation.
21. Bernard Yurke and Allen P. Mills, Jr. Using DNA to power nanostructures. *Genetic Programming and Evolvable Machines*, 4:111–122, 2003.
22. Robert M. Dirks. *Analysis, design, and construction of nucleic acid devices.* PhD thesis, California Institute of Technology, 2005.
23. Justin S. Bois. *Analysis of interacting nucleic acids in dilute solutions.* PhD thesis, California Institute of Technology, 2007.
24. Claude E. Shannon. A symbolic analysis of relay and switching circuits. Technical Report Master's Thesis, Massachussetts Institute of Technology, 1940.
25. Lulu Qian, Ying Wang, Zhao Zhang, Jian Zhao, Dun Pan, Yi Zhang, Qiang Liu, Chunhai Fan, Jun Hu, and Lin He. Analogic china map constructed by DNA. *Chinese Science Bulletin*, 51:2973–2976, 2006.
26. Shawn M. Douglas, James J. Chou, and William M. Shih. DNA-nanotube-induced alignment of membrane proteins for NMR structure determination. *Proc. Nat. Acad. Sci. USA*, 104:6644–6648, 2007.
27. Donald E. Thomas and Philip R. Moorby. *The Verilog Hardware Description Language.* Kluwer, 1991.
28. Ulirich Golze. *VLSI Chip Design with the Hardware Description Language VERILOG.* Springer-Verlag, 1996.
29. Moe Shahdad, Roger Lipsett, Erich Marschner, Kellye Sheehan, Howard Cohen, Ron Waxman, and Dave Ackley. VHSIC hardware description language. *IEEE Computer*, 18:94–103, 1985.
30. Arwen Brenneman and Anne Condon. Strand design for biomolecular computation. *Theor. Comput. Sci.*, 287:39–58, 2002.

31. Morgan A. Bishop, Arkadii G. D'Yachkov, Anthony J. Macula, Thomas E. Renz, and Vyacheslav V. Rykov. Free energy gap and statistical thermodynamic fidelity of DNA codes. *Journal of Computational Biology*, 14:1088–1104, 2007.
32. Kalim U. Mir. A restricted genetic alphabet for DNA computing. In Laura F. Landweber and Eric B. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS*, pages 243–246, Providence, RI, 1998. American Mathematical Society.
33. Ravinderjit S. Braich, Nickolas Chelyapov, Cliff Johnson, Paul W. K. Rothemund, and Leonard M. Adleman. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science*, 296:499–502, 2002.
34. Dirk Faulhammer, Anthony R. Cukras, Richard J. Lipton, and Laura F. Landweber. Molecular computation: RNA solutions to chess problems. *Proc. Nat. Acad. Sci. USA*, 97(3):1385–1389, 2000.
35. Igor G. Panyutin and Peggy Hsieh. Formation of a single base mismatch impedes spontaneous DNA branch migration. *Journal of Molecular Biology*, 230:413–424, 1993.
36. Igor G. Panyutin and Peggy Hsieh. Kinetics of spontaneous DNA branch migration. *Proc. Nat. Acad. Sci. USA*, 91:2021–2025, 1994.
37. Ming-Yang Kao, Manan Sanghi, and Robert Schweller. Randomized fast design of short DNA words. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, pages 1275–1286, 2005.
38. Oliver D. King. Bounds for DNA codes with constant GC-content. *Electronic Journal of Combinatorics*, 10:R33, 2003.
39. Agilent Technologies. SurePrint technology (web page). http://www.chem.agilent.com/scripts/generic.asp?lpage=557&indcol=N&prodcol=Y#custom.
40. Agilent Technologies. Multi-pack gene expression microarrays (web page). http://www.chem.agilent.com/scripts/generic.asp?lpage=51683&indcol=Y&prodcol=Y.
41. NimbleGen Systems, Inc. Array synthesis (web page). http://www.nimblegen.com/technology/manufacture.html.
42. David E. Kohne, Stuart A. Levison, and Michael J. Byers. Room temperature method for increasing the rate of DNA reassociation by many thousandfold: The phenol emulsion reassociation technique. *Biochemistry*, 16:5329–5341, 1977.
43. Arach Goldar and Jean-Louis Sikorav. DNA renaturation at the water-phenol interface. *Eur. Phys. J. E.*, 14:211–239, 2004.