Parallel and Scalable Computation and Spatial Dynamics with DNA-Based Chemical Reaction Networks on a Surface

Lulu Qian[1,2] and Erik Winfree[1,2,3]

[1] Bioengineering,
[2] Computer Science,
[3] Computation and Neural Systems
California Institute of Technology, Pasadena, CA 91125, USA
{luluqian,winfree}@caltech.edu

Abstract. We propose a theoretical framework that uses a novel DNA strand displacement mechanism to implement abstract chemical reaction networks (CRNs) on the surface of a DNA nanostructure, and show that surface CRNs can perform efficient algorithmic computation and create complex spatial dynamics. We argue that programming molecular behaviors with surface CRNs is systematic, parallel and scalable.

1 Introduction

Despite the increasing complexity of synthetic DNA circuits and machinery [48], every step that is made towards building more sophisticated and powerful molecular systems has also revealed new challenges in the scalability and programmability of such systems. For example, in DNA strand displacement circuits, a larger circuit results in a larger number of distinct DNA molecules, and the spurious interactions among these DNA molecules can temporarily disable a fraction of the circuit components and thus slow down the computation and increase the error rate. A number of implementations have been proposed to explore the possibility of using spatial organization that allows circuit components to interact without requiring diffusion, and thus increase the speed of computation and limit spurious interactions to immediate neighbors [5,28]. Interestingly, localized circuits are related to molecular robotics [10]. Spatial organization could also enable running multiple instances of circuits in parallel, each on a different surface but all in the same test tube.

Another challenge is to implement fully general and efficient (space-limited) algorithmic computation that is experimentally feasible and scalable. By this we mean, informally, systems capable of storing and retrieving data from memory, performing logical operations, and executing iterative loops that repeat a processing step. For the purpose of this paper, we are not concerned with whether literally unbounded memory is available, so both a Turing machine and a laptop computer (i.e. a sequential digital logic circuit) would qualify. Efficient (space-limited) computations are possible on both, and a small program can perform a large computation. In contrast, feedforward digital logic circuits can only perform limited computation: they have no memory and during execution from input to output, each logic gate updates exactly once. Even if provided new inputs, a feedforward circuit can only make decisions based on the current input signals, while a Turing machine or sequential circuit can store information in memory and access it at a later point to make more sophisticated decisions based on both current and historical input signals.

In this work, we first explain an abstract model of chemical reaction networks (CRNs) on a surface, in the context of how bimolecular reactions alone can efficiently simulate (space-bounded) Turing machines. Then we introduce the implementation of formal unimolecular and bimolecular reactions on surface, based on a novel DNA mechanism (the three-way initiated four-way strand displacement reaction) that can recognize a DNA signal, pull it off from the surface, and simultaneously load a different signal onto the same location. Following the implementation, we give an example of propagating waves created from a simple set of surface CRNs. Finally, to demonstrate the power and elegance of surface CRNs, we develop systematic approaches for building continually active logic circuits and cellular automata. These approaches are highly efficient and scalable in two ways: First, they compute and generate complex spatial dynamics in parallel. Second, they use a constant number of distinct molecules for vastly different sizes of molecular programs.

2 Abstract Chemical Reaction Networks on a Surface

In the abstract model of surface CRNs, formal chemical species (e.g A, B, C, etc.) are located on a finite two-dimensional grid, and each site has a finite number of neighboring sites. Chemical species at any site can be recognized and converted into an arbitrary different species multiple times, either at a single site through a formal unimolecular reaction (e.g. $A \to B$), or cooperatively with a neighboring site through a formal bimolecular reaction (e.g. $A + B \to C + D$, if A and B are neighbors, then A gets converted to C while simultaneously B gets converted to D). Bimolecular reactions can be applied in any orientation on the surface, but always the first reactant is replaced by the first product, and the second by the second. Thus, $A + B \to C + D$ is effectively the same as $B + A \to D + C$, but is distinct from $A{+}B \to D{+}C$. Importantly, molecules do not move from site to site unless there is an explicit reaction, so by default there is no diffusion. (Diffusion could be "simulated" by including extra reactions such as $A + B \to B + A$ that allow species A and B to randomly walk through each other.) Unlike well-mixed CRNs, both unimolecular and bimolecular reaction rate constants have the same units, per second, because molecules have explicit locations and thus it is not necessary to approximate diffusion and collision probabilities.

Previous works have shown that synthetic DNA molecules can be used to implement arbitrary CRNs in a well-mixed solution [36,7]. As an extension and complement to well-mixed CRNs, we will show that DNA strands can be tethered on the surface of a DNA nanostructure such as DNA origami [32] to implement surface CRNs. Fig. 1a shows the abstract diagram of a small portion of a DNA

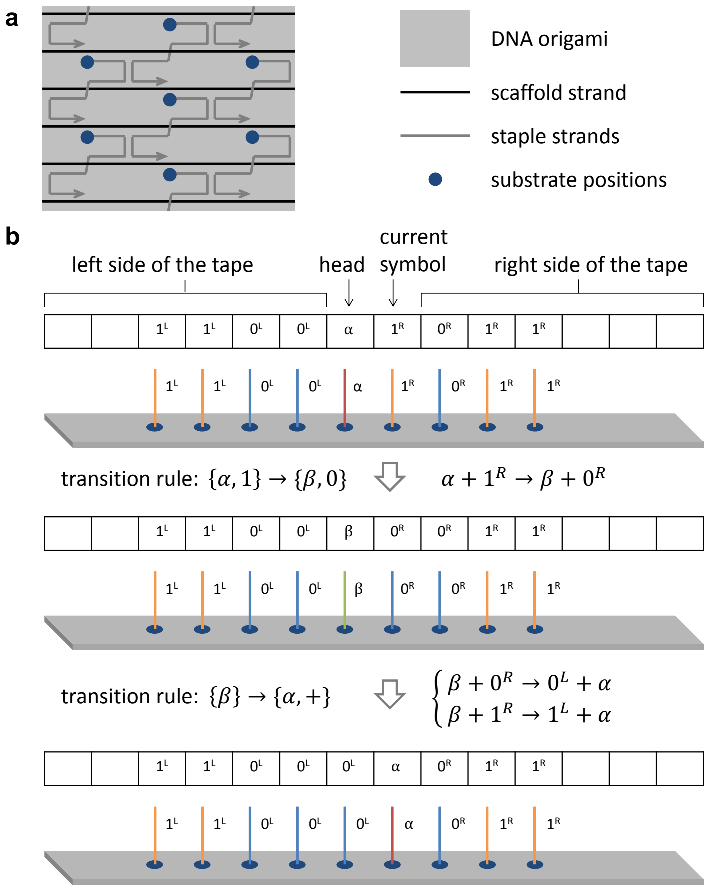

a
DNA origami
scaffold strand
staple strands
substrate positions
b
left side of the tape
head
current symbol
right side of the tape
1^L 1^L 0^L 0^L α 1^R 0^R 1^R 1^R
transition rule: $\{\alpha, 1\} \to \{\beta, 0\}$
$\alpha + 1^R \to \beta + 0^R$
1^L 1^L 0^L 0^L β 0^R 0^R 1^R 1^R
transition rule: $\{\beta\} \to \{\alpha, +\}$
$\beta + 0^R \to 0^L + \alpha$
$\beta + 1^R \to 1^L + \alpha$
1^L 1^L 0^L 0^L 0^L α 0^R 1^R 1^R

Fig. 1. Abstract chemical reaction networks (CRNs) on a surface. **(a)** Abstract diagram of a small portion of a DNA origami. All substrate locations are about 6 nm apart from each other. **(b)** An example of efficient molecular Turing machines implemented with abstract surface CRNs.

origami. Each staple strand can be extended from the 5' or 3' end to provide substrate positions for tethering DNA molecules that represent distinct chemical species. Because all staple strands have distinct sequences, all substrate locations on a origami surface are uniquely addressable. This allows us to locate specific DNA-based chemical species at their initial sites on a two-dimensional hexagonal grid of about 200 sites with about 6 nm distance between any neighboring sites, using a single DNA origami. With DNA origami arrays [26], much larger two-dimensional grids can be created.

Turing machines are one of the simplest models for universal computation, and building molecular Turing machines has a been a challenge for over 30 years. Charles Bennett was the first to come up with an implementation that uses hypothetical enzymes [2]. Later, concrete molecular implementations were proposed. The DNA tile self-assembly model was developed and proved to be Turing universal, but it has the distinct disadvantage of storing the entire history of its computation within an array of DNA tiles [33]. Both non-autonomous [34,1,35] and autonomous [44] DNA-based Turing machines were designed; however they required enzymes such as ligase and restriction enzymes for their operation, and their complexity discouraged experimental implementation. Well-mixed CRNs were also shown to be probabilistically Turing universal, but the computation requires molecular counts and therefore volumes that grow exponentially with the amount of memory used [37]. Recently, we developed a stack machine model with DNA polymers [30] that can simulate Turing machines efficiently, but there must be exactly one copy of the polymer DNA molecule for each stack, which introduces significant experimental challenges.

As an example to illustrate the power and generality of surface CRNs, Fig. 1b shows the implementation for a hypothetical molecular Turing machine with a finite tape. Unlike the stack machine implementation with polymer CRNs, our new Turing machine implementation with surface CRNs allows multiple independent Turing machines to operate in parallel within the same test tube, and there is no slow-down as the reaction volume gets larger. However, whereas the stack machine construction has an explicit mechanism for growing an unbounded amount of memory, the Turing machine construction here is limited to the size of the origami surface; in principle, this limit is obviated by unbounded self-assembly of origami [26], ideally configured so that self-assembly is inhibited until triggered [11] by the Turing machine needing more memory.

To review, a Turing machine has a head that moves along a tape and reads or writes symbols on the tape. The function of a Turing machine is decided by a set of transition rules. Each rule updates the state of the head and the symbol near the head based on the current state and symbol information, and moves the head to the left or right on the tape. Here we use an equivalent variant of the standard Turing machine where state change and movement steps are separate [3]. A tethered DNA strand on a DNA origami surface represents a state (such as α or β) if the position corresponds to the head, and represents a symbol (such as 0 or 1) if the position is on the tape. All symbols on the left side of the tape are 0^L or 1^L, and those on the right side are 0^R or 1^R. Each transition rule can be encoded in one or more formal bimolecular reactions on a surface. For example, transition rule $\{\alpha, 1\} \to \{\beta, 0\}$, which reads current state α and current symbol 1 and updates the state to β and the symbol to 0, can be encoded in surface reaction $\alpha + 1^R \to \beta + 0^R$. Transition rule $\{\beta\} \to \{\alpha, +\}$, which reads current state β, updates the state to α, and moves the head to one cell on the right, can be encoded as two surface reactions $\beta + 0^R \to 0^L + \alpha$ and $\beta + 1^R \to 1^L + \alpha$.

3 Implementation of Surface CRNs

It is a significant challenge to implement surface CRNs. Existing DNA implementations of well-mixed CRNs [36,7] use the mechanism of three-way strand displacement [45]. Such implementations can not be used for recognizing and updating a DNA signal on a surface for two reasons: First, in the process of

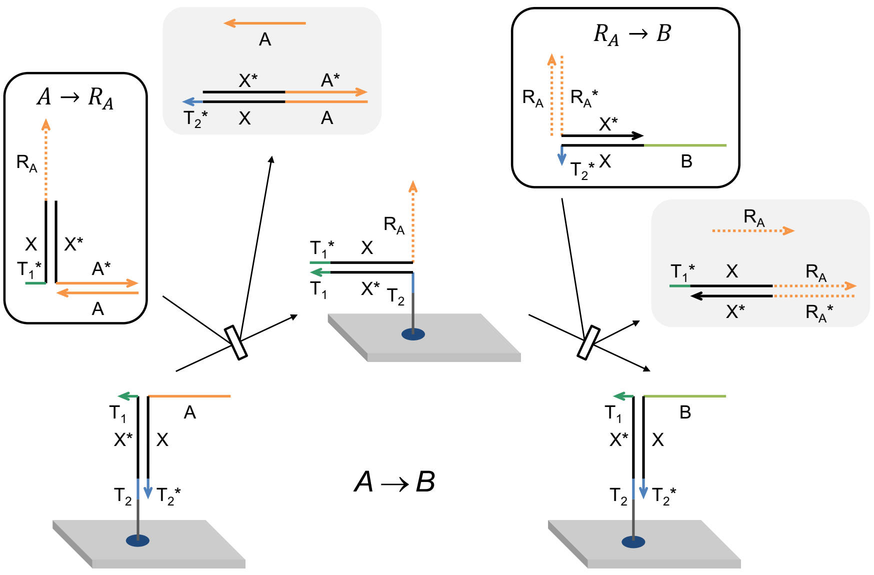


Fig. 2. DNA implementation of formal unimolecular reaction $A \rightarrow B$ on a surface. Fuel molecules are highlighted with outlines and waste molecules are shaded with light grey background; they are free-floating molecules in solutions that are maintained at a constant concentration.

three-way strand displacement, a single-stranded signal that is being recognized will become bound to a complementary strand that is eventually part of a waste molecule. Thus the waste molecule would be stuck on the surface attached to the current signal strand. Second, upon completion of three-way strand displacement, a new signal strand would be released into the solution, instead of staying on the surface as an updated signal.

In contrast, the mechanism of four-way branch migration [40,27] allows the recognition of a double-stranded signal with two adjacent single-stranded toeholds. Upon completion of branch migration, all strands will have swapped their base-paring partners. One strand in the original signal will now become part of a new signal as a result of the formation of a new pair of adjacent single-stranded toeholds, which makes it possible to recognize and update a signal on a surface. Unlike the high specificity of signals that can be encoded in branch migration domains with three-way strand displacement [47], four-way branch migration relies on distinct toehold sequences to represent different signals, and thus it is limited to implementations that require a small number of signal species.

Here, we show that with the help of associative/combinatorial toehold [6,15], three-way strand displacement and four-way branch migration can be integrated into one single step, simultaneously achieving the high specificity of signals and the localized updating of signals through swapping base-paring partners. We call this new mechanism three-way initiated four-way strand displacement, and we use it to implement surface CRNs.

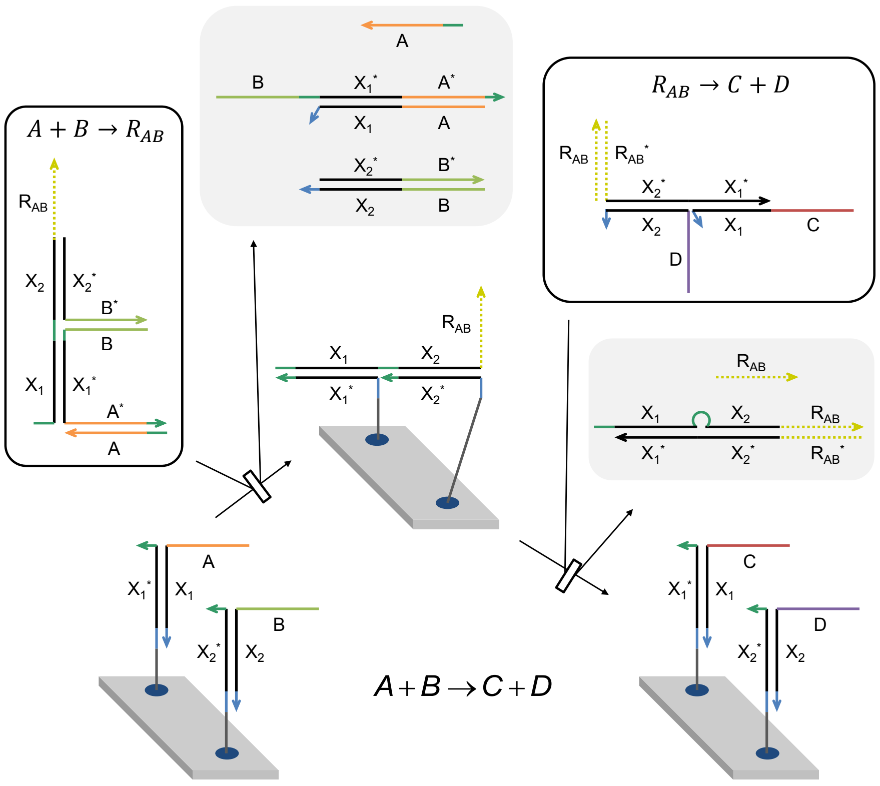


Fig. 3. DNA implementation of formal bimolecular reaction $A + B \to C + D$ on a surface. Fuel molecules are highlighted with outlines and waste molecules are shaded with light grey background, they are free-floating molecules in solution that are maintained at a constant concentration.

As shown in Fig. 2 and appendix Fig. A1, a DNA signal encoding chemical species A on a surface consists of a short single-stranded toehold domain (e.g. T_1, perhaps of 6 nucleotides) and a long single-stranded recognition domain (e.g. A, perhaps of 15 nucleotides) held together by a double-stranded branch migration domain (e.g. X and X^*) followed by a double-stranded toehold domain (e.g. T_2 and T_2^*). Initially, a fuel molecule $A \to R_A$ is free floating in solution. It first binds to signal A through the single-stranded domain T_1^*, three-way branch migration occurs within the double-stranded A and A^* domain, and a single-stranded waste molecule is produced. Simultaneous with the three-way branch migration, four-way branch migration occurs within two double-stranded X and X^* domains. When the double-stranded molecule on the right side is only attached by a short toehold T_2, it will spontaneously fall off the surface and become a waste molecule. At this point, signal A on a surface has been replaced by signal R_A, but with a different orientation. We then use a second fuel molecule $R_A \to B$ to replace R_A with B on the surface and to restore the original orientation of A. Note that the single strand A in fuel molecule $A \to R_A$ and single strand R_A

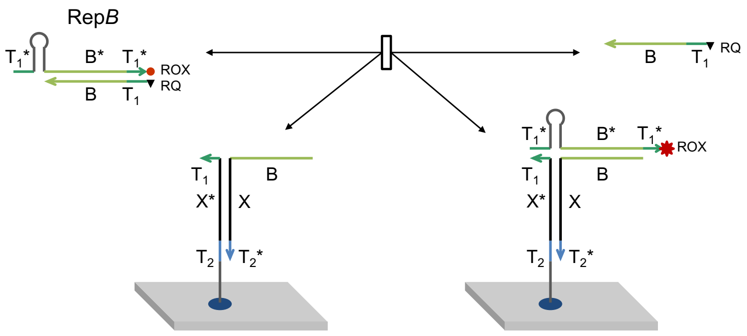


Fig. 4. DNA implementation of a reversible reporting reaction on a surface.

in fuel molecule $R_A \to B$ not only increase the specificity of signal recognition through branch migration, but also protect the fuel molecules from binding to each other in a larger network (e.g. when $A \to B$ and $X \to A$ co-exist).

Doubling the complexity of each fuel molecule, we can now implement the formal bimolecular reaction $A + B \to C + D$ on a surface. As shown in Fig. 3 and appendix Fig. A2, signals A and B are located at neighboring sites on the surface. Fuel molecule $A + B \to R_{AB}$ first undergoes three-way initiated four-way branch migration with signal A on the surface; at the end of this process two short toeholds spontaneously disassociate and neighboring signal B can bind to the intermediate product and undergo a second three-way initiated four-way branch migration reaction to replace both signals A and B with a joint signal R_{AB} on the surface. A second fuel molecule $R_{AB} \to C+D$ then recognizes R_{AB}, and signal D will be placed at the original site of B followed by C being placed at the original site of A.

The keen observer will note that this mechanism requires neighboring sites to make use of distinct branch migration domains X_1 and X_2, rather than universally X. This constraint can be accommodated by using a checkerboard arrangement of X_1 and X_2 sites on the origami and by multiplying the number of fuel species – for example, each unimolecular reaction will need both fuels using X_1 and fuels using X_2 (unless, for some reason, we wish to restrict the unimolecular reaction to just one color of the checkerboard). As it is straightforward to handle, we will henceforth ignore this minor complication.

To experimentally read DNA signals on a surface, we propose a reporting mechanism that reversibly produces a fluorescent signal. As shown in Fig. 4, the free-floating reporter molecule is labeled with a fluorophore and a quencher. A reversible three-way strand displacement reaction separates the fluorophore from the quencher and results in increased fluorescence that can be measured in bulk by a spectrofluorometer. This mechanism, reversibly binding and activating the fluorophore, is compatible with the DNA-PAINT method [22] for super-resolution microscopy, suggesting that dynamic spatial patterns could be observed on a single origami or origami array.

$$\begin{cases} Q + A \xrightarrow{1/s} A + A \\ A \xrightarrow{0.6/s} R \\ R \xrightarrow{0.03/s} Q \\ Q + C \xrightarrow{0.01/s} A + C \end{cases}$$

A	■
R	▨
Q	□
C	□

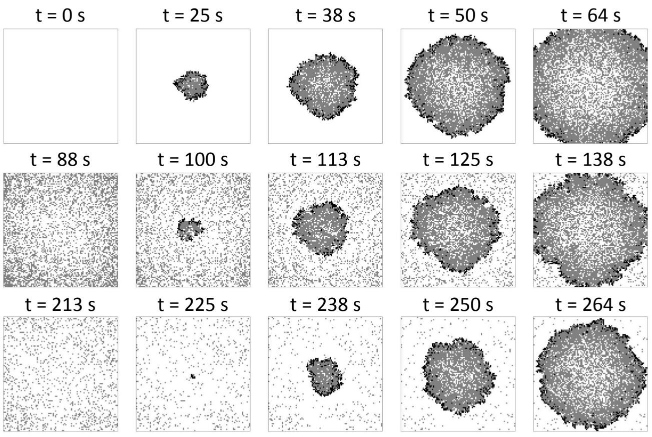


Fig. 5. A nanoscaled pacemaker that triggers a propagating wave. Simulated on a 100 by 100 grid, black pixels indicate signal A, grey pixels indicate signal R, and white pixels indicate signal Q. The signal in the center of the grid is always C.

4 Dynamic Spatial Patterns

With just unimolecular and bimolecular surface CRNs, dynamic spatial patterns can be created on a two-dimensional DNA origami surface. For example, with the set of four reactions shown in Fig. 5, a propagating wave pulse can be repeatedly generated. Initially, the site in the center of the grid has signal C ("the pacemaker") and all other sites have signal Q. The only reaction that can take place under this initial condition is $Q + C \rightarrow A + C$, allowing the signal in the center to update one of its neighbors from Q to A. Subsequently, a fast reaction $Q + A \rightarrow A + A$ will occur, and each site with signal A will update its neighbors from Q to A, creating the front of a wave. A slower reaction $A \rightarrow R$ will then convert signals A to R, thereby identifying older parts of the wave and helping establish directionality of the wave propagation. A even slower reaction $R \rightarrow Q$ will restore signals from R to Q after the wavefront has passed. Finally, the slowest reaction $Q + C \rightarrow A + C$ enables a new wave to emerge from the center after the previous wave has faded.

In this example, if all possible reactions execute synchronously (independent of rate constants), the propagating wave will expand deterministically — it is the 3-state Greenberg-Hastings model of excitable media [17]. But discrete CRNs are intrinsically asynchronous, all signals will be updated stochastically, and the edge of the wave will be less well defined, as shown in the simulation in Fig. 5. To obtain reliable wave propagation, rate parameters must be tuned roughly as described above. In the DNA implementation, desired rates can be achieved by varying the lengths of toeholds on fuel molecules encoding the unimolecular and bimolecular reactions (assuming the three-way initiated four-way strand displacement follow roughly the same range of kinetics as three-way strand displacement [46]). As an alternative to tuning rates, it is possible to design (typically larger) surface CRNs that behave exactly as if they were updated synchronously;

we will discuss such an approach later in the paper, in the context of cellular automata.

Rather than eliminate it, the randomness of asynchronous reaction execution can be embraced, and in combination with explicit reactions that simulate two-dimensional diffusion (e.g. $X + e \to e + X$ for all diffusible species X and a special "empty space" signal e), we obtain the entire space of chemical reaction-diffusion systems in the stochastic limit, closely analogous to reactive lattice gas automata models [4]. For example, spiral wave dynamical patterns could be achieved using the six-reaction Oregonator model of the Belousov-Zhabotinsky excitable medium [21].

5 Continuously Active Logic Circuits

Unlike in reaction-diffusion systems, signals in surface CRNs by default will remain in their exact location until a reaction occurs. This feature enables precise geometric control at the single-molecule level, and can be exploited to carry out precise tasks such as digital circuit computation. As shown in Fig. 6b, to construct a two-input OR gate with surface CRNs, three neighboring sites are initially assigned with blank signals $B^{\cup x}$, $B^{\cup y}$ and $B^{\cup z}$, and each of them has another blank neighboring site B to serve as a wire that moves signals around and connects layers of logic gates together (Fig. 6a). Logic OR computation can be performed with six bimolecular reactions. The first four reactions recognize the two input signals 00, 01, 10, or 11 at the x and y sites, update the y site to the correct output signal $0^{\cup k}$ or $1^{\cup k}$, and reset the x site to be blank. The last two reactions move the output signal to the z site and reset the y site to be blank. Similarly, AND gate and NOT gate can be implemented with six and two reactions respectively (Fig. 6cd). Additional straightforward reactions are needed to load the signal from the input wires onto the gate, and to push the output onto its wire (Fig. 6a). These unidirectional reactions ratchet the signals forward, despite the random walks on the wires.

With two additional sets of reactions implementing signal fan-out and crossing wires (Fig. 6ef), arbitrary feedforward logic circuits can be systematically translated into surface CRNs. An example circuit that calculates the square roots of four-bit binary numbers is shown in Fig. 6g. To run the circuit, 0 or 1 signals are initiated at $x_1, \ldots, x_4$, while y_1 and y_2 are in state B. Signals asynchronously propagate through the circuit. Two-input gates must wait until both input signals arrive, before they can produce an output. Crossing wires are designed to ensure that deadlock is impossible. The correct circuit outputs are eventually produced at y_1 and y_2 regardless of the order in which reactions execute.

Unlike the previous well-mixed DNA logic circuits, which deplete some circuit components by the end of each computation and thus are not capable of responding to a new set of input signals [31], these logic circuits with surface CRNs are continuously active. With free-floating fuel molecules in large excess, the signal on each site can be updated multiple times, switching between "0" and "1" and back, as needed. With reversible reporters that read the output signals without consuming them, a changed set of input signals will trigger a cascade of reactions resulting in the update of output signals and associated fluorescence.

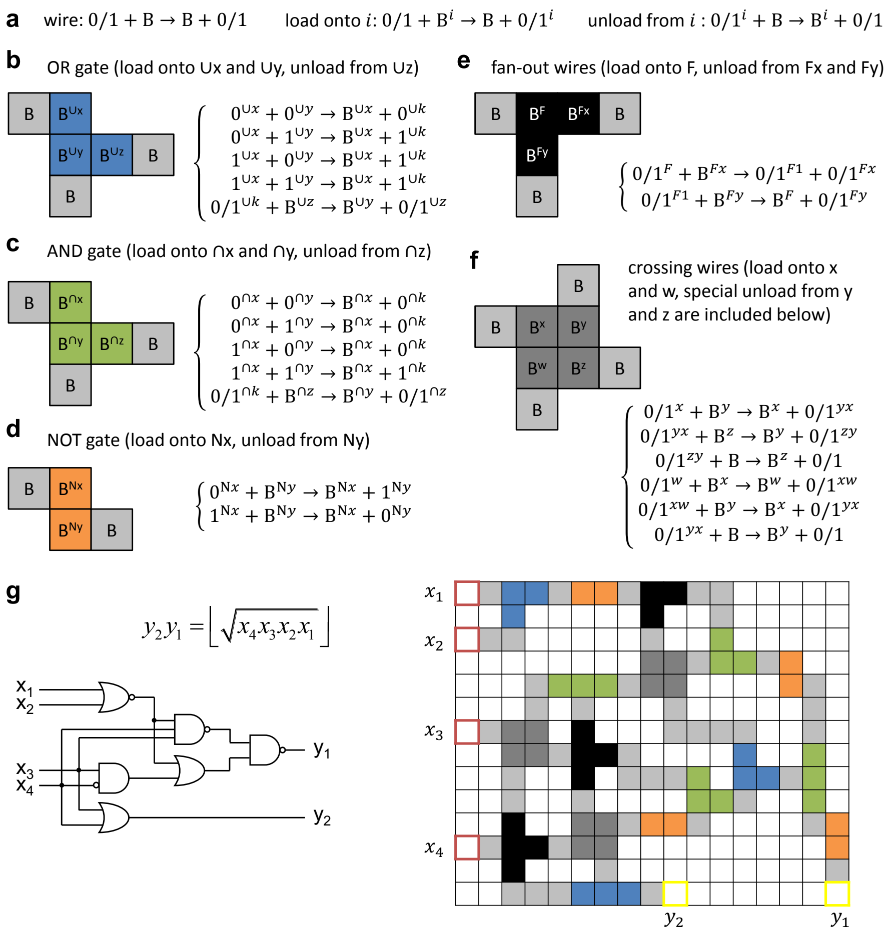

a wire: 0/1 + B → B + 0/1 load onto i: 0/1 + B^i → B + 0/1^i unload from i : 0/1^i + B → B^i + 0/1
b OR gate (load onto ∪x and ∪y, unload from ∪z)
$0^{\cup x} + 0^{\cup y} \to B^{\cup x} + 0^{\cup k}$
$0^{\cup x} + 1^{\cup y} \to B^{\cup x} + 1^{\cup k}$
$1^{\cup x} + 0^{\cup y} \to B^{\cup x} + 1^{\cup k}$
$1^{\cup x} + 1^{\cup y} \to B^{\cup x} + 1^{\cup k}$
$0/1^{\cup k} + B^{\cup z} \to B^{\cup y} + 0/1^{\cup z}$
c AND gate (load onto ∩x and ∩y, unload from ∩z)
$0^{\cap x} + 0^{\cap y} \to B^{\cap x} + 0^{\cap k}$
$0^{\cap x} + 1^{\cap y} \to B^{\cap x} + 0^{\cap k}$
$1^{\cap x} + 0^{\cap y} \to B^{\cap x} + 0^{\cap k}$
$1^{\cap x} + 1^{\cap y} \to B^{\cap x} + 1^{\cap k}$
$0/1^{\cap k} + B^{\cap z} \to B^{\cap y} + 0/1^{\cap z}$
d NOT gate (load onto Nx, unload from Ny)
$0^{Nx} + B^{Ny} \to B^{Nx} + 1^{Ny}$
$1^{Nx} + B^{Ny} \to B^{Nx} + 0^{Ny}$
e fan-out wires (load onto F, unload from Fx and Fy)
$0/1^{F} + B^{Fx} \to 0/1^{F1} + 0/1^{Fx}$
$0/1^{F1} + B^{Fy} \to B^{F} + 0/1^{Fy}$
f crossing wires (load onto x and w, special unload from y and z are included below)
$0/1^{x} + B^{y} \to B^{x} + 0/1^{yx}$
$0/1^{yx} + B^{z} \to B^{y} + 0/1^{zy}$
$0/1^{zy} + B \to B^{z} + 0/1$
$0/1^{w} + B^{x} \to B^{w} + 0/1^{xw}$
$0/1^{xw} + B^{y} \to B^{x} + 0/1^{yx}$
$0/1^{yx} + B \to B^{y} + 0/1$
g $y_2y_1 = \lfloor \sqrt{x_4x_3x_2x_1} \rfloor$
x_1 x_2 x_3 x_4 y_1 y_2

Fig. 6. Continuously active logic circuits. **(a)** wire, loading and unloading, **(b)** OR gate, **(c)** AND gate, **(d)** NOT gate, **(e)** fan-out wires and **(f)** crossing wires implemented with surface CRNs. "0/1" is shorthand for two rules of the same form, one with all instances "0" and the other with all instances "1". **(g)** A four-bit square root circuit implemented with surface CRNs. The three-input AND gate is implemented with 2 two-input AND gates. The fan-out of three is implemented similarly as (but distinctly from) the fan-out of two, with an extra state $F2$ of site B^F.

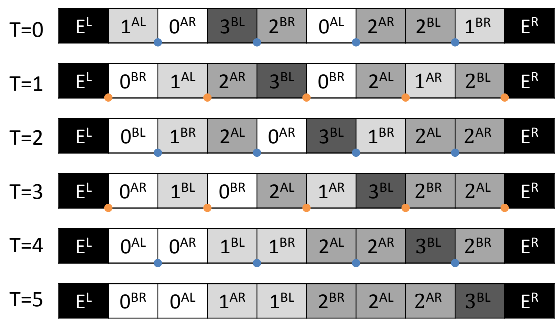
An additional benefit of the continuously active logic circuit architecture using surface CRNs is that iterative sequential circuits can be implemented using the same mechanisms. For example, if three NOT gates are wired together in a ring (a canonical oscillatory circuit), and a single 0 signal is placed on one of the wires, then the signal will travel around and around the ring, flipping from 0 to 1 and back as it goes. More usefully, to iterate a function $f(x_1, x_2, \ldots, x_n) = (x'_1, x'_2, \ldots, x'_n)$, the outputs of a circuit computing $f()$ merely need to be routed back to the input, controlled with a synchronizing signal that ensures that the new inputs are not activated until all of the previous outputs have been computed and collected. Thus, in principle, arbitrary finite state machines, and even standard central processing unit (CPU) designs, can be efficiently implemented using surface CRNs on a large enough origami array. Such designs are closely related to the implementation of delay-insensitive circuits in asynchronous cellular automata [25].

Logic circuits with surface CRNs should be more scalable than previous DNA-based logic circuits. Any feedforward or sequential logic circuit can be implemented with the same set of signal molecules on the surface and fuel molecules in solution, regardless of the circuit size. A different circuit will correspond to a different layout of signals on the surface, and a larger circuit simply requires a larger grid on DNA origami. For example, all three OR gates in the square root circuit (Fig. 6g) at different locations on the surface will interact with the same set of fuel molecules (Fig. 6b) to perform the desired computation, and all three OR gates can operate at the same time. Assuming the concentrations of all fuel molecules stay constant, which can be approximated by using a small amount of DNA origami (and hence signal molecules on its surface) and a large excess of fuel molecules in solution, the speed of each logic operation should stay the same in larger circuits. This is in contrast to well-mixed DNA circuits, where the maximum total DNA concentration limit requires that larger circuits operate at lower signal concentrations, resulting in a per-operation slowdown linear in the number of gates (c.f. SI section S15 of [31]). Finally, because each origami can contain a different circuit and/or be initialized with different input, billions of independent circuit computations can execute within a single test tube, in contrast to well-mixed bulk-phase reactions where only a single circuit is executed.

6 Cellular Automata

Compared to Turing machines where only the single site representing the head is updated at a time, cellular automata take full advantage of parallel computation and can efficiently generate complex patterns that evolve over time. A cellular automaton has a grid of cells, each with an initial state. Based on the current state of itself and its neighbors, each cell can be updated to a new state. A set of transition rules determine how the cells are updated, and these rules are applied to all cells in the grid simultaneously. Interesting biological processes such as the behavior of muscle cells, cardiac function, animal coat markings, and plant ecology have been simulated by cellular automata [13]. Even some of the simplest one-dimensional cellular automata with two states (0 and 1) are rich enough to support universal computation [8]. DNA tile self-assembly has been used to successfully implement cellular automata [33], but only by constructing a static pattern representing the cellular automata's space-time history. A previous proposal to implement one-dimensional cellular automata dynamics on one-dimensional structures [43] used a variety of enzymes and may not be experimentally feasible.

An example of cellular automata with surface CRNs is shown in Fig. 7. It is an implementation of a one-dimensional block cellular automaton that sorts


T=0 E^L 1^{AL} 0^{AR} 3^{BL} 2^{BR} 0^{AL} 2^{AR} 2^{BL} 1^{BR} E^R
T=1 E^L 0^{BR} 1^{AL} 2^{AR} 3^{BL} 0^{BR} 2^{AL} 1^{AR} 2^{BL} E^R
T=2 E^L 0^{BL} 1^{BR} 2^{AL} 0^{AR} 3^{BL} 1^{BR} 2^{AL} 2^{AR} E^R
T=3 E^L 0^{AR} 1^{BL} 0^{BR} 2^{AL} 1^{AR} 3^{BL} 2^{BR} 2^{AL} E^R
T=4 E^L 0^{AL} 0^{AR} 1^{BL} 1^{BR} 2^{AL} 2^{AR} 3^{BL} 2^{BR} E^R
T=5 E^L 0^{BR} 0^{AL} 1^{AR} 1^{BL} 2^{BR} 2^{AL} 2^{AR} 3^{BL} E^R

each transition rule

$$\{x, y\} \rightarrow \{x^*, y^*\}$$

is implemented with:

$$\begin{cases} x^{AL} + y^{AR} \rightarrow x^{*BR} + y^{*AL} \\ x^{BL} + y^{BR} \rightarrow x^{*AR} + y^{*BL} \end{cases}$$

edge conditions for each state x are implemented with:

$$\begin{cases} \mathrm{E}^L + x^{AR} \rightarrow \mathrm{E}^L + x^{AL} \\ \mathrm{E}^L + x^{BR} \rightarrow \mathrm{E}^L + x^{BL} \\ x^{AL} + \mathrm{E}^R \rightarrow x^{BR} + \mathrm{E}^R \\ x^{BL} + \mathrm{E}^R \rightarrow x^{AR} + \mathrm{E}^R \end{cases}$$

transition rules for sorting numbers between 0 to 3:

$\{0,0\} \rightarrow \{0,0\}$	$\{0,1\} \rightarrow \{0,1\}$	$\{0,2\} \rightarrow \{0,2\}$	$\{0,3\} \rightarrow \{0,3\}$
$\{1,0\} \rightarrow \{0,1\}$	$\{1,1\} \rightarrow \{1,1\}$	$\{1,2\} \rightarrow \{1,2\}$	$\{1,3\} \rightarrow \{1,3\}$
$\{2,0\} \rightarrow \{0,2\}$	$\{2,1\} \rightarrow \{1,2\}$	$\{2,2\} \rightarrow \{2,2\}$	$\{2,3\} \rightarrow \{2,3\}$
$\{3,0\} \rightarrow \{0,3\}$	$\{3,1\} \rightarrow \{1,3\}$	$\{3,2\} \rightarrow \{2,3\}$	$\{3,3\} \rightarrow \{3,3\}$

Fig. 7. A one-dimensional block cellular automaton that sorts numbers. The behavior with synchronous updates is shown, with blue and orange dots indicating pairing on alternate time steps. Superscripts indicate extra information that the asynchronous surface CRN must track to ensure correct behavior. Transition rules in red performs sorting and transition rules in black maintain the updates in alternating order.

single-digit numbers. A blocked (aka partitioning) cellular automata executes by dividing the array into pairs of neighboring sites, synchronously applying a look-up-table of rules for how to replace each pair by a new pair of values, shifts the pairing by one, and repeats. Unlike the cellular automata, our model of surface CRNs is intrinsically non-oriented and asynchronous. To allow the recognition of orientations and to synchronize the update of all sites, each signal molecule on a DNA origami surface (e.g. 1^{AL} and 0^{AR}) is designed to include both information about a number (e.g. 0, 1, 2, or 3) and information about block pairing (AL, AR, BL or BR, where AL indicates the left side of an "A" pair, AR indicates the right side of an "A" pair, and so on). Each transition rule $\{x, y\} \rightarrow \{x^\star, y^\star\}$ that reads the current states of two neighboring cells and updates them with new states can be implemented with two fuel molecules encoding $x^{AL} + y^{AR} \rightarrow x^{\star BR} + y^{\star AL}$ and $x^{BL} + y^{BR} \rightarrow x^{\star AR} + y^{\star BL}$. This ensures that the ways of pairing up signals alternate after each update, so all signals will be compared with neighbors on both sides; further, it ensures that a site won't be updated again until its neighbor has caught up. Transition rules such as $\{1,0\} \rightarrow \{0,1\}$, $\{2,1\} \rightarrow \{1,2\}$, and $\{3,2\} \rightarrow \{2,3\}$ ensure that smaller numbers will be sorted to the left and larger numbers will be sorted to the right. Left edge signal E^L, right edge signal E^R and corresponding fuel molecules are used to complete the function of sorting.

The approach illustrated here for simulating synchronous one-dimensional block cellular automata with asynchronous one-dimensional surface CRNs can be generalized to provide simulations of arbitrary synchronous two-dimensional block cellular automata [39] and even traditional cellular automata with von Neumann and Moore neighborhood update functions [23]. The construction uses the same basic principles but is more elaborate, and is not presented here.

7 Discussion

The key mechanism that we proposed for implementing surface CRNs is a novel strand displacement primitive that we call the three-way initiated four-way strand displacement reaction. This reaction is more complex than any other DNA strand displacement primitives that have been demonstrated so far, and it is important to understand the kinetics and robustness of this reaction. We argue that because the toehold binding step makes use of both a regular toehold plus a co-axial stacking bond, it is immediately followed by three-way branch migration (without the delay one might expect from a remote toehold [16]). Presumably right after the initiation of three-way branch migration, the four-way branch migration will simultaneously take place. Thus the reaction should be completed roughly as fast as a single-step three-way or four-way strand displacement reaction.

It is also important to evaluate the potential leak reactions in our surface CRNs implementation. Any leak reactions that could occur between signal molecules on a surface would have to involve two double-stranded domains without toeholds, which is unlikely. Any leak reactions that could occur between fuel molecules in solution will only produce waste molecules in solution and not affect the state of signal molecules on surfaces. The most likely leak would be between a fuel molecule and a mismatching surface-bound signal: although single short toeholds are not very effective for initiating 4-way branch migration [9], it is possible that at some rate 4-way branch migration could initiate even without the 3-way branch migration taking place. Overall, however, the potential leak reactions should be less significant than other types of strand displacement systems. Also note that, because the number of distinct fuel molecules is constant with any size of the logic circuits with surface CRNs, the number of leak reactions will not scale with the complexity of the circuits. For cellular automata, a small set of transition rules can be used to create very complex behaviors programmed by various initial conditions, and in these cases the leak reactions will not scale with the size of programs either.

The implementation of formal bimolecular reactions on a surface raises a couple of concerns, including the step that requires two toeholds to temporarily disassociate when other part of the molecules are still held together, and the step that requires initiation of four-way branch migration with a toehold on one side and a bulge on another. We believe it should be possible to simplify and optimize this implementation.

It is inevitable that experimental implementation of surface CRNs will have occasional errors, and therefore robustness and fault-tolerance will be important issues to address in future work. Successful approaches may depend upon the type of surface CRN being implemented. For example, for digital circuits on DNA origami, classical and modern techniques for faulty digital circuits [41,19] provide solutions if the faults are transient bit-flips or permanent device failure. But DNA surface CRNs may have other types of faults, such as arbitrary signal errors (not just 0-to-1 or 1-to-0) and even signal loss, origami-to-origami exchange, or stuck intermediate reaction steps. Asynchronous cellular automata models are closer models; although error correction is more difficult there, techniques have been developed both in one and two dimensions [14,42]. Finding practical and effective methods for molecular programming remains an important challenge.

Even if errors at the molecular level (leak reactions, stuck intermediates, defective molecules, etc.) are negligible, the complexity of understanding DNA strand displacement system implementations at the domain level already calls for simulation tools and logical verification techniques. An important step will be expanding the capability of software like Visual DSD [29] to handle DNA complexes with arbitrary secondary structure as well as localization on surfaces – this would also enable simulation of a large variety of molecular robotics implementations. Even when the full set of domain-level reactions have been worked out for DNA strand displacement systems, logical errors may be difficult to identify manually. Automated approaches for verifying the correctness of well-mixed DNA strand displacement systems are being developed [24] and these may provide a starting point for verifying surface-based systems.

If these considerable challenges can be overcome, the resulting control over spatially-organized molecular systems could provide important new capabilities. As a systematic implementation for an extremely general class of systems, our approach leverages the effort spent characterizing and debugging a small set of reaction mechanisms to construct a wide range of system behaviors, ranging from continuously active algorithmic computation with memory, to pattern formation and spatial dynamics. Applications could include synthetic molecular systems that regulate biochemical pathways in response to continuously changing environmental signals, therapeutic molecular devices [12] that efficiently produce treatment decisions not only based on biomarkers that are currently present, but also those that indicate historical conditions of the target cell, or molecular-scale devices and instruments that precisely regulate nanoscale components (such as plasmonic elements [38] or chemical moieties [18,20]) to achieve measurement or synthesis tasks.

Acknowledgments. We thank Ho-Lin Chen and David Yu Zhang for insightful discussions and suggestions. LQ and EW were supported by the Molecular Programming Project under NSF expedition in computing (0832824 and 1317694). LQ was supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund (1010684), a Faculty Early Career Development Award from NSF (1351081), and a Okawa Foundation Research Grant (0656180).

References

1. Beaver, D.: A universal molecular computer. DNA Based Computers, DIMACS 27, 29–36 (1996)

2. Bennett, C.H.: The thermodynamics of computation – a review. International Journal of Theoretical Physics 21, 905–940 (1982)
3. Bennett, C.H.: Logical reversibility of computation. IBM Journal of Research and Development 17, 525–532 (1973)
4. Boon, J.P., Dab, D., Kapral, R., Lawniczak, A.: Lattice gas automata for reactive systems. Physics Reports 273, 55–147 (1996)
5. Chandran, H., Gopalkrishnan, N., Phillips, A., Reif, J.: Localized hybridization circuits. In: Cardelli, L., Shih, W. (eds.) DNA 17. LNCS, vol. 6937, pp. 64–83. Springer, Heidelberg (2011)
6. Chen, X.: Expanding the rule set of DNA circuitry with associative toehold activation. Journal of the American Chemical Society 134, 263–271 (2011)
7. Chen, Y.J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., Seelig, G.: Programmable chemical controllers made from DNA. Nature Nanotechnology 8, 755–762 (2013)
8. Cook, M.: Universality in elementary cellular automata. Complex Systems 15, 1–40 (2004)
9. Dabby, N.L.: Synthetic molecular machines for active self-assembly: prototype algorithms, designs, and experimental study. Ph.D. thesis, California Institute of Technology (2013)
10. Dannenberg, F., Kwiatkowska, M., Thachuk, C., Turberfield, A.J.: DNA walker circuits: Computational potential, design, and verification. In: Soloveichik, D., Yurke, B. (eds.) DNA 2013. LNCS, vol. 8141, pp. 31–45. Springer, Heidelberg (2013)
11. Dirks, R.M., Pierce, N.A.: Triggered amplification by hybridization chain reaction. Proceedings of the National Academy of Sciences 101, 15275–15278 (2004)
12. Douglas, S.M., Bachelet, I., Church, G.M.: A logic-gated nanorobot for targeted transport of molecular payloads. Science 335, 831–834 (2012)
13. Ermentrout, G.B., Edelstein-Keshet, L.: Cellular automata approaches to biological modeling. Journal of Theoretical Biology 160, 97–133 (1993)
14. Gács, P.: Reliable cellular automata with self-organization. Journal of Statistical Physics 103, 45–267 (2001)
15. Genot, A.J., Bath, J., Turberfield, A.J.: Combinatorial displacement of DNA strands: application to matrix multiplication and weighted sums. Angewandte Chemie International Edition 52, 1189–1192 (2013)
16. Genot, A.J., Zhang, D.Y., Bath, J., Turberfield, A.J.: Remote toehold: a mechanism for flexible control of DNA hybridization kinetics. Journal of the American Chemical Society 133, 2177–2182 (2011)
17. Greenberg, J.M., Hastings, S.: Spatial patterns for discrete models of diffusion in excitable media. SIAM Journal on Applied Mathematics 34(3), 515–523 (1978)
18. Gu, H., Chao, J., Xiao, S.J., Seeman, N.C.: A proximity-based programmable DNA nanoscale assembly line. Nature 465, 202–205 (2010)
19. Han, J., Jonker, P.: A defect-and fault-tolerant architecture for nanocomputers. Nanotechnology 14, 224 (2003)
20. He, Y., Liu, D.R.: Autonomous multistep organic synthesis in a single isothermal solution mediated by a DNA walker. Nature Nanotechnology 5, 778–782 (2010)
21. Jahnke, W., Winfree, A.T.: A survey of spiral-wave behaviors in the Oregonator model. International Journal of Bifurcation and Chaos 1, 445–466 (1991)
22. Jungmann, R., Steinhauer, C., Scheible, M., Kuzyk, A., Tinnefeld, P., Simmel, F.C.: Single-molecule kinetics and super-resolution microscopy by fluorescence imaging of transient binding on DNA origami. Nano Letters 10, 4756–4761 (2010)

23. Kari, J.: Theory of cellular automata: A survey. Theoretical Computer Science 334, 3–33 (2005)
24. Lakin, M.R., Phillips, A., Stefanovic, D.: Modular verification of DNA strand displacement networks via serializability analysis. In: Soloveichik, D., Yurke, B. (eds.) DNA 19. LNCS, vol. 8141, pp. 133–146. Springer, Heidelberg (2013)
25. Lee, J., Adachi, S., Peper, F., Mashiko, S.: Delay-insensitive computation in asynchronous cellular automata. Journal of Computer and System Sciences 70, 201–220 (2005)
26. Liu, W., Zhong, H., Wang, R., Seeman, N.C.: Crystalline two-dimensional DNA-origami arrays. Angewandte Chemie 123, 278–281 (2011)
27. Muscat, R.A., Bath, J., Turberfield, A.J.: A programmable molecular robot. Nano Letters 11, 982–987 (2011)
28. Muscat, R.A., Strauss, K., Ceze, L., Seelig, G.: DNA-based molecular architecture with spatially localized components. In: Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA), pp. 177–188 (2013)
29. Phillips, A., Cardelli, L.: A programming language for composable DNA circuits. Journal of The Royal Society Interface 6, S419–S436 (2009)
30. Qian, L., Soloveichik, D., Winfree, E.: Efficient turing-universal computation with DNA polymers. In: Sakakibara, Y., Mi, Y. (eds.) DNA 16. LNCS, vol. 6518, pp. 123–140. Springer, Heidelberg (2011)
31. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science 332, 1196–1201 (2011)
32. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006)
33. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2, e424 (2004)
34. Rothemund, P.W.K.: A DNA and restriction enzyme implementation of Turing machines. DNA Based Computers, DIMACS 27, 75–119 (1996)
35. Smith, W.D.: DNA computers in vitro and in vivo. DNA Based Computers, DIMACS 27, 121–185 (1996)
36. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences 107, 5393–5398 (2010)
37. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Natural Computing 7(4), 615–633 (2008)
38. Tan, S.J., Campolongo, M.J., Luo, D., Cheng, W.: Building plasmonic nanostructures with DNA. Nature Nanotechnology 6, 268–276 (2011)
39. Toffoli, T., Margolus, N.: Cellular automata machines: a new environment for modeling. MIT Press (1987)
40. Venkataraman, S., Dirks, R.M., Rothemund, P.W.K., Winfree, E., Pierce, N.A.: An autonomous polymerization motor powered by DNA hybridization. Nature Nanotechnology 2, 490–494 (2007)
41. Von Neumann, J.: Probabilistic logics and the synthesis of reliable organisms from unreliable components. Automata Studies 34, 43–98 (1956)
42. Wang, W.: An asynchronous two-dimensional self-correcting cellular automaton. In: Proceedings of 32nd Annual Symposium on Foundations of Computer Science, pp. 278–285. IEEE (1991)
43. Yin, P., Sahu, S., Turberfield, A.J., Reif, J.H.: Design of autonomous DNA cellular automata. In: Carbone, A., Pierce, N.A. (eds.) DNA 11. LNCS, vol. 3892, pp. 399–416. Springer, Heidelberg (2006)

44. Yin, P., Turberfield, A.J., Sahu, S., Reif, J.H.: Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 10. LNCS, vol. 3384, pp. 426–444. Springer, Heidelberg (2005)
45. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature 406, 605–608 (2000)
46. Zhang, D.Y., Winfree, E.: Control of DNA strand displacement kinetics using toehold exchange. Journal of the American Chemical Society 131, 17303–17314 (2009)
47. Zhang, D.Y., Chen, S.X., Yin, P.: Optimizing the specificity of nucleic acid hybridization. Nature Chemistry 4, 208–214 (2012)
48. Zhang, D.Y., Seelig, G.: Dynamic DNA nanotechnology using strand-displacement reactions. Nature Chemistry 3, 103–113 (2011)

Appendix

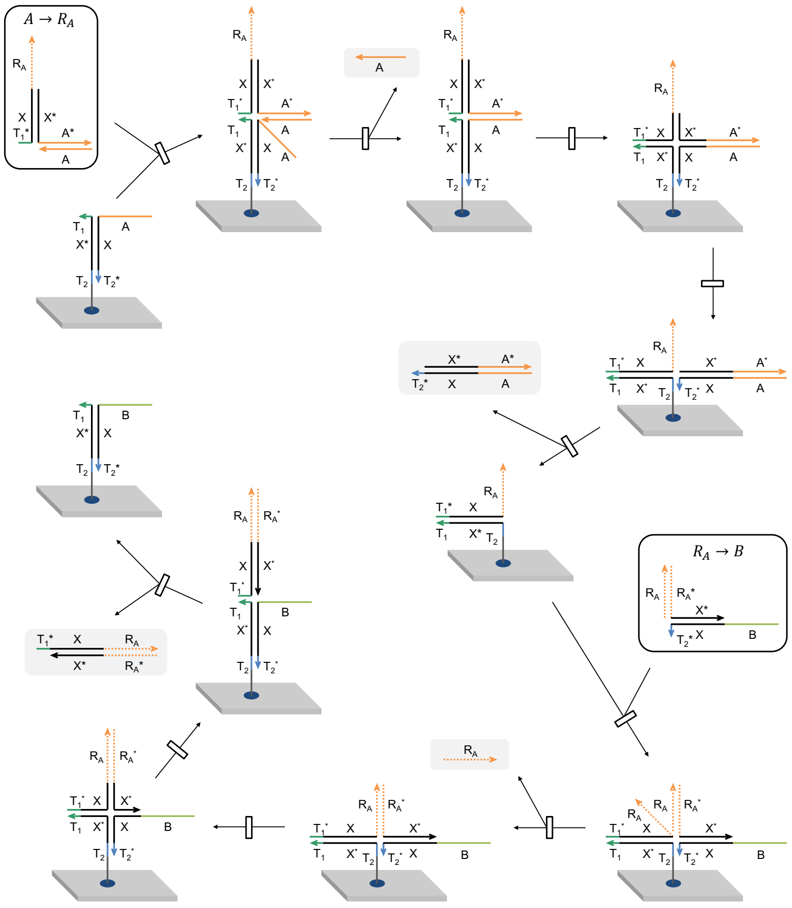


Fig. 8. Detailed mechanism of formal unimolecular reaction $A \rightarrow B$ on a surface

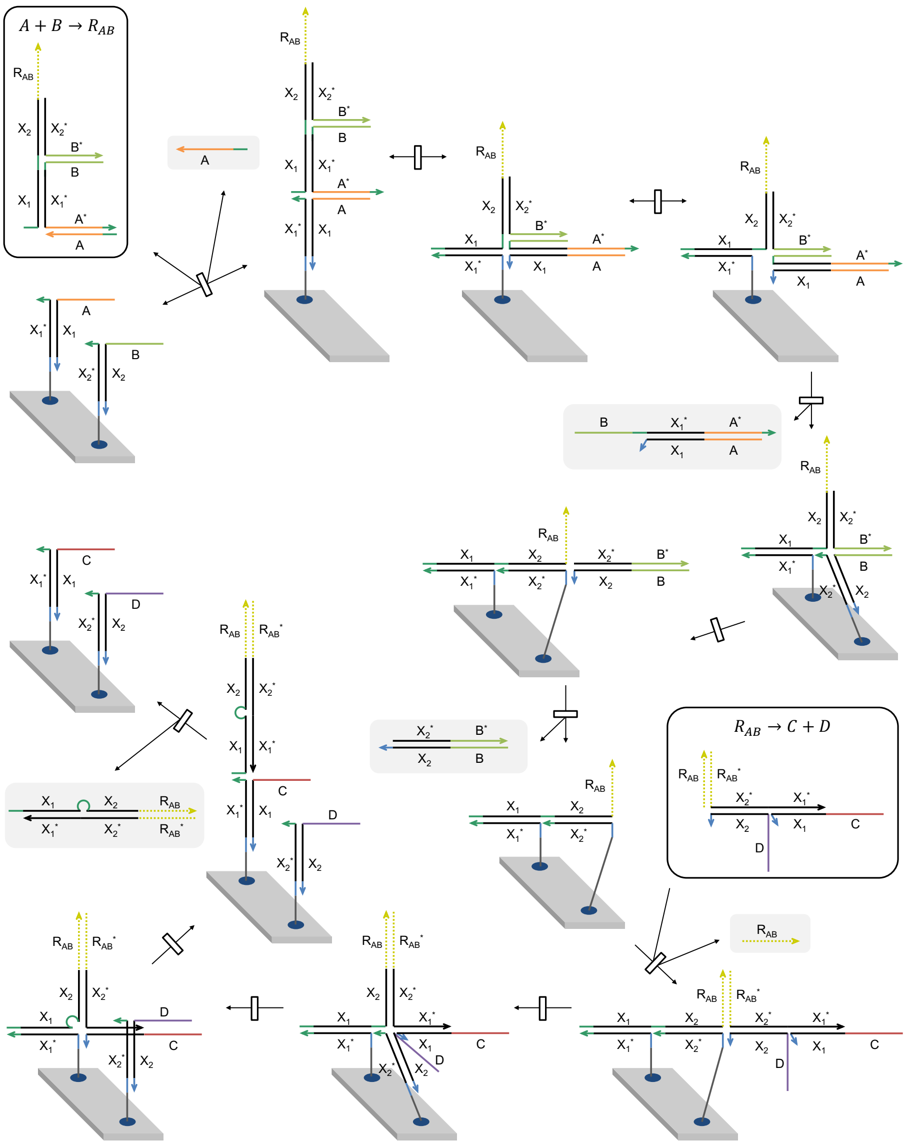


Fig. 9. Detailed mechanism of formal bimolecular reaction $A+B \rightarrow C+D$ on a surface