DNA-based Molecular Architecture
with Spatially Localized Components

Richard A. Muscat†   Karin Strauss†‡   Luis Ceze†   Georg Seelig†

† University of Washington   ‡ Microsoft Research

ABSTRACT

Performing computation inside living cells offers life-changing applications, from improved medical diagnostics to better cancer therapy to intelligent drugs. Due to its bio-compatibility and ease of engineering, one promising approach for performing in-vivo computation is DNA strand displacement. This paper introduces computer architects to DNA strand displacement "circuits", discusses associated architectural challenges, and proposes a new organization that provides practical composability. In particular, prior approaches rely mostly on stochastic interaction of freely diffusing components. This paper proposes practical *spatial* isolation of components, leading to more easily designed DNA-based circuits. DNA nanotechnology is currently at a turning point, with many proposed applications being realized [20, 9]. We believe that it is time for the computer architecture community to take notice and contribute.

Categories and Subject Descriptors

B. Hardware [**Emerging Technologies**]

General Terms

Design

Keywords

DNA-based In-cell Computation, Spatial Localization

1. INTRODUCTION

Synthetic molecular circuits that work reliably in a complex cellular environment and that can sense and respond to that environment offer the potential for many significant applications. For example, in-vivo computation with molecular circuits could enable novel approaches to medical diagnostics and imaging, selective drug delivery to cancer cells, and even intelligent drugs that lay latent inside cells and release automatically when a disease is first detected.

DNA is an ideal physical substrate for carrying out molecular computation because its simple primitives make it amenable to effective engineering. Furthermore, DNA circuitry is bio-compatible, thus well-suited for intra-cell operation. Using DNA to carry out

computation has been explored for almost two decades, starting when Adleman proposed computing Hamiltonian paths [1] with DNA. Researchers subsequently explored other basic computing primitives with DNA, from Boolean gates [33, 29, 23] to neural networks [24] to chemical reaction networks [32, 3, 4].

As the technology evolves, however, it is only natural to expect architectural questions to arise, as they did with the development of electronic computers. We believe the computer architecture community will find this technology of interest and that it has much to contribute to answering these questions. Several questions we face with electronics today are very relevant to biological computers, as well. For example, both require performing computation with unreliable components, and both require abstractions to manage complexity, simplifying the design of more complex systems.

In particular, we focus on DNA strand displacement systems because of their simplicity and flexibility. The DNA strand displacement mechanism has enabled the construction of many dynamic molecular systems, including motors, sensors, circuits, amplifiers, and reconfigurable nanostructures. These developments are reviewed in the literature [40, 12]. Importantly, the goal of these approaches is not to solve general computational problems faster than electronics could. Instead, current DNA-based computing aims to develop molecular information processing similar to that which occurs *inside cells* and to engineer systems that can *process information that is intrinsically molecular* (such as the information encoded in the concentrations of various cellular molecules). In other words, it is a promising way to embed bio-compatible computing elements into a living cell.

This paper introduces computer architects to the basics of DNA computing, focusing on DNA strand displacement mechanisms. Further, it discusses several architectural challenges associated with computing using DNA and focuses on one of these challenges: the fact that DNA reactions happen in diffusion and therefore lack spatial isolation. Without this property, designers must confront code isolation issues, *i.e.*, each logical gate must be built with a different set of DNA sequences to avoid interference between gates or modules. Finally, we propose a solution to this problem: organizing gates in a practical, spatially isolated fashion, which can be done with the help of DNA origami. In this paper, the spatial organization on a DNA substrate addresses the gate interference problem, simplifying the design process and improving composability of rich DNA strand displacement systems. We describe the design of basic primitives (gates) and evaluate them both analytically and experimentally. Our analytic results show the benefits of spatial isolation on ease of design, especially as circuits grow larger. Our experimental results show evidence that the proposed approach is viable in practice.

2. BACKGROUND

2.1 DNA as a Programmable Material for Nanoscale Engineering

Single-stranded DNA is a linear polymer that consists of four chemically different building blocks, the nucleotides adenine (A), cytosine (C), guanine (G), and thymine (T). A DNA strand is characterized by its linear sequence, which can be abstracted as a string of the symbols A, C, G and T. Two single strands can bind ("hybridize") to form double-stranded DNA (a double helix) if their sequences are complementary, that is, if A in one strand faces T in the other strand while C faces G, as shown in Figure 1. The pairs $A : T$ and $C : G$ are the canonical Watson-Crick base pairs. Two strands with perfectly complementary sequences bind/hybridize to each other more strongly than two strands with partial complementarity. The two ends of a DNA strand are chemically different and are referred to as the 5' and 3' ends, respectively (these names refer to specific carbon atoms within the nucleotide that participate in the attachment to the previous and next nucleotide in the chain, respectively). In a double helix, the two strands have opposite orientation. Single-stranded DNA is floppy; double-stranded DNA is more rigid. The width of a double helix is 2 nm, and the distance between sequential base pairs along the helix is 0.34 nm.

DNA nanotechnology [30, 31, 40, 12] uses DNA as a programmable nanoscale building material because its predictability of interactions is ideal for an engineering approach. DNA nanotechnology chiefly relies on single-stranded molecules that are relatively short (typically less than 100 nucleotides in length). Such "oligonucleotides" of arbitrary sequence can be synthesized chemically and bought from a variety of synthesis companies; in contrast, biological DNA — the carrier of genetic information — mostly occurs as a double-stranded molecule that can be millions of base pairs in length. The use of DNA as an engineering material has been additionally facilitated by the rapidly decreasing cost of oligonucleotide preparation and purification [5].

It is often convenient to abstract contiguous DNA bases into functional DNA *domains* that act as a unit, as Figure 1 also shows (*e.g.*, domains a and b). In most figures, the DNA sequences are not shown explicitly because it is expected that most described systems will work for many choices of domain sequences. Domains are represented in this paper by letters; a starred domain denotes a domain complementary in sequence to the domain without a star (*e.g.*, domain b^* is complementary to domain b), and complementary domains hybridize to each other via Watson-Crick base pairing. In the following discussion, we distinguish between "short" (*e.g.*, domains a and c^*) and "long" domains (*e.g.*, domains b and d^*) based on the stability of a *duplexed* (double-stranded) domain. Short duplexes are unstable at room temperature, while long duplexes are assumed to be stable. In practice, the length and sequence composition required to make a duplex (un)stable depends on temperature and reaction conditions. For typical experimental conditions, *i.e.*, temperatures from 20-40°C and a saline reaction buffer, short domains are usually 4-6 nucleotides, while long domains are 15-25 nucleotides.

2.2 DNA Strand Displacement Reactions: Primitives for Molecular Programming

Strand displacement is the process through which two strands with partial or full complementarity hybridize to each other, displacing one or more pre-hybridized strands in the process. A simple example of a strand displacement reaction is illustrated in Figure 2.

Strand displacement occurs as a series of reactions between single strands and double strands and involves multiple functional domains. A strand displacement reaction is initiated when two complementary *toehold* domains (*i.e.*, short domains), like a and a^* in Figure 2, bind to each other. A single-stranded input $a^*b^*c^*$ binds to the toehold a (because it is exposed) and displaces the previously bound strand ($b^*c^*d^*e^*$) in a base-by-base random walk process, where two domains with identical sequences (b^*) compete to bind with the same complementary long domain (b). The final step is the complete release of the initial binding partner by the new partner, *i.e.*, the separation of toeholds c and c^* and release of strand $b^*c^*d^*e^*$. The rate of a strand displacement reaction is determined by the length of the toehold domain and can be carefully controlled over several orders of magnitude [39, 14, 42], enabling engineering control over the kinetics of synthetic DNA devices.

2.3 Molecular Signals Can Propagate through Strand Displacement Cascades

Strand displacement releases at least one single-stranded nucleic acid product. In a DNA strand displacement cascade, this released strand can trigger a downstream reaction, as shown in Figure 2. Once $b^*c^*d^*e^*$ is released in the first step, it displaces $d^*e^*f^*g^*$ from cde. Note that the final *output signal* $e^*f^*g^*$ has no domains in common with the *input signal* $a^*b^*c^*$ of the two-stage cascade, so they are completely independent. Functionally, this cascade is a sequence translator since an input with one sequence is deactivated and replaced by an output with a different sequence. Here, we distinguish between the signal $e^*f^*g^*$ and the strand $d^*e^*f^*g^*$. The domain d^* does not participate in any downstream reaction and depends on strand history. Signals with identical sequences other than different history domains act in the same way in downstream reactions. In contrast, the output of the first stage, $c^*d^*e^*$, is an *internal signal* because it depends on both upstream input and downstream output (for example, it shares the toehold domain c^* with the input and the toehold domain e^* with the output).

2.4 DNA Strand-Displacement Circuits

Cascades of strand displacement reactions have been used to experimentally implement feed-forward digital logic circuits capable of combinational logic [29, 23]. As an example, this section uses the designs presented by Seelig *et al.* and Soloveichik *et al.* [29, 32].

Signals in these circuits are single-stranded nucleic acids with the domain structure *short:long:short*. Signals interact with multi-strand DNA complexes ("gates") through strand displacement reactions. In this framework, the two-stage reaction cascade of Figure 2 represents a logical repeater. The input is the signal $a^*b^*c^*$, and the output is $e^*f^*g^*$. Multi-input logic can be implemented through the use of multiple strand gate complexes. The AND gate by Soloveichik *et al.* [32] consists of three partially complementary strands assembled into a complex. Two sequential strand displacement reactions with two different input strands trigger the disassembly of this complex, resulting in the release of an output strand. If only one or none of the inputs is present, no output is released. Figure 3 shows the process. When the first input signal $a^*b^*c^*$ arrives, it displaces $b^*c^*d^*$ and provides a toehold for binding of the second input $d^*e^*f^*$. The latter, in turn, can displace $e^*f^*g^*h^*$, which now frees $f^*g^*h^*$ to serve as input in a downstream reaction. As in the previous example, one additional component, a one-stage translator gate (not shown), is required to obtain complete sequence independence between inputs and outputs.
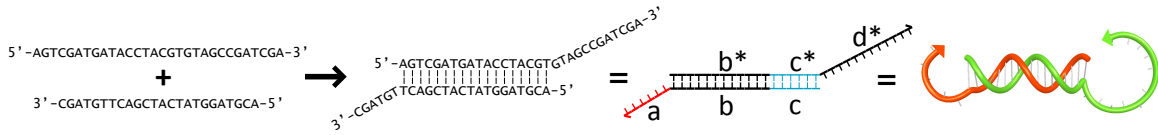
Figure 1: Two DNA strands with complementary sequence hybridize and form double-stranded DNA (or duplex) in a double helix shape. In this example, the two strands are only partially complementary. We use a simplified notation where functional DNA domains are labeled with letters and * indicates complementarity.
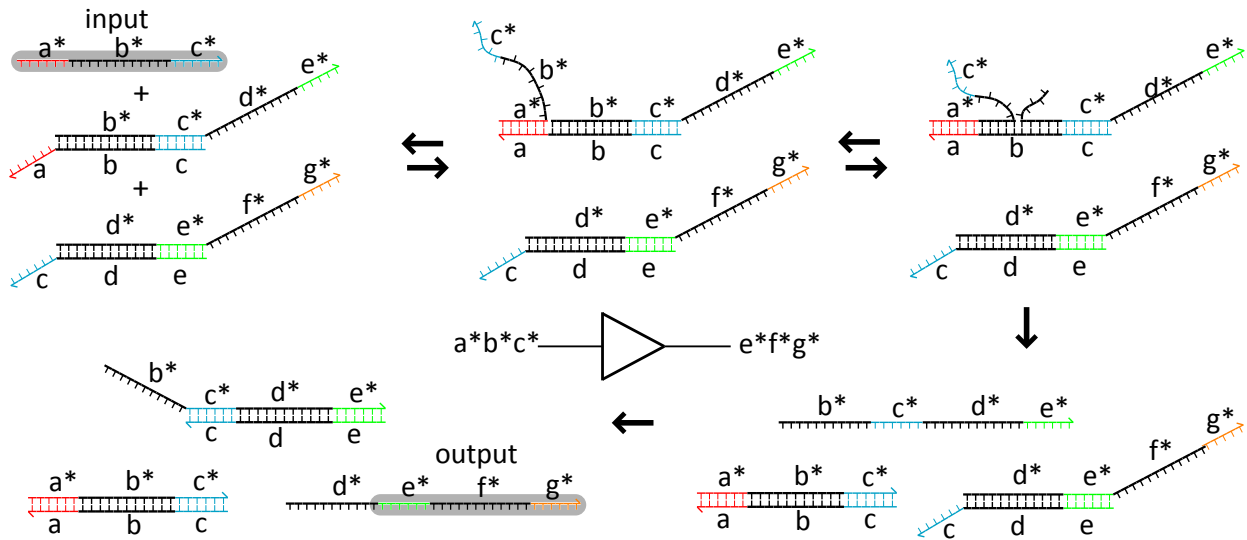

Figure 2: DNA strand displacement is a primitive for molecular information processing. A single-stranded DNA input signal displaces another strand (internal signal) from a duplex. This internal signal then displaces the output signal, freeing it to react with other sequences. Input and output signals have no domains in common.
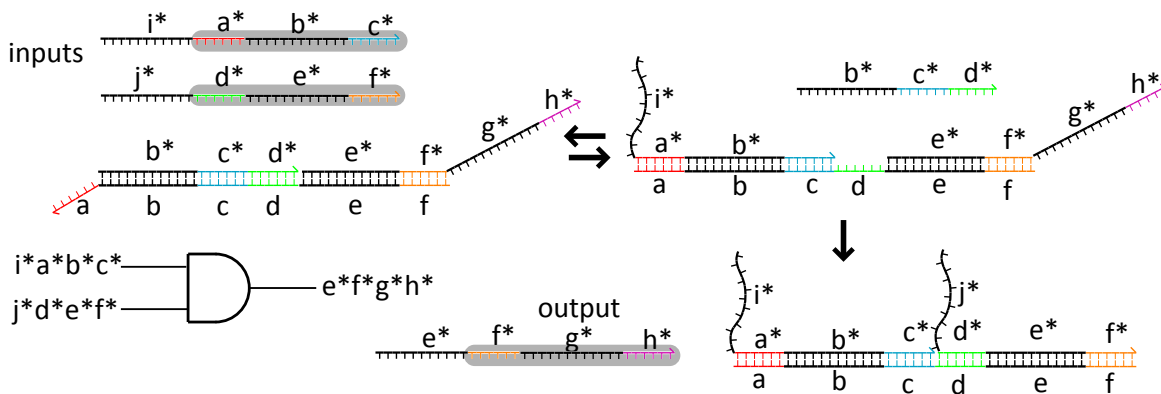

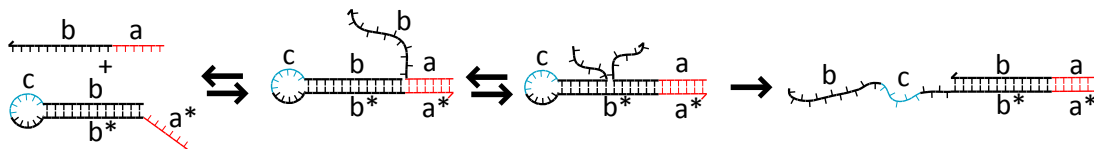**Figure 3:** AND gate in solution by Soloveichik *et al.* [32].

Figure 4: Hairpin being opened by an input and exposing a new output.

3

2.5 DNA Hairpins Enable Co-localization of Inputs and Outputs

A DNA strand that is partially self-complementary can bind to itself and form a *hairpin*. A hairpin has a double-stranded stem and a single-stranded loop that connects the two complementary stem domains. We use this hairpin motif as the basic ingredient for the construction of localized DNA circuits: Hairpins are easily attachable to an origami substrate (defined in Section 2.6), and their structure hides certain domains; this delays the undesirable early start of certain reactions until these domains are exposed by previous reactions.

As shown in Figure 4, a hairpin can participate in a strand displacement reaction [8]. In this example, domain a^* serves as a toehold for binding domain a of the single-stranded input. Domain b of the input opens the stem and unfolds the hairpin. Note that the reaction results in a single-stranded "output" (cb) that remains attached in a complex with the input, but that is now exposed and can further react with other domains. The output has a different toehold than the input (c instead of a), but in this case only the toehold domain of the output c is independent of the input (because b is required to open the hairpin).

2.6 DNA Origami as a Nano-board for Molecular Self-assembly

DNA origami is a DNA self-assembly technique recently developed by Rothemund [26]. In DNA origami, a long single-stranded scaffold DNA is folded into a target shape using short *staple strands*, as shown in Figure 5. Each staple is complementary to two different domains within the scaffold and, depending on the choice of staples, a variety of different shapes can be obtained. The characteristic length of a DNA origami, such as the square sketched in the figure, is on the order of 100 nm. However, the thickness of a one-dimensional origami sheet is only about 2 nm, corresponding to the width of one double helix. Because each staple attaches to a unique address and the distance between sequential base pairs in a double helix is only 0.34 nm, DNA origami enables the placement of molecules with nanoscale precision. Thus, we can leverage this fact to attach hairpins to DNA origami so that individual hairpins, once activated, will react only with other hairpins attached nearby, localizing the circuit. Mammalian cells have already been shown to internalize DNA nanostructures such as tetrahedra (an approximately 7 nm DNA pyramid) [35] and origami [27]. Additionally, origami has been shown to be preserved in mammalian cell lysate [16], so it is unlikely to degrade before the intended circuit has a chance to be activated.

2.7 Manufacturing DNA Systems

As noted, individual arbitrary DNA sequences can be synthesized chemically. However, if the desired product is a construct of several strands, there are additional manufacturing steps. Individual sequences are mixed in solution and annealed, *i.e.*, heated to a high temperature where all bonds that connect strands are disrupted, then gradually cooled. During the cooling process, the designed, stable, double-stranded regions form earlier than partially matched regions, which are less stable interactions. For this reason, different components in a single displacement reaction are annealed separately and then mixed in the same solution; otherwise, they would react at fabrication time, rendering the circuit useless.

3. ARCHITECTURAL ISSUES

There are many parallels between digital electronic circuits and digital DNA strand displacement circuits. There are also several unique architectural challenges that arise in DNA strand displacement circuits. This section presents parallels and discusses challenges.

Lack of spatial isolation. Signal strands and gate complexes all diffuse freely in solution and interact stochastically. The DNA sequence of a signal, rather than its location, determines the gates with which it can interact. Therefore, to compose a circuit with multiple gates, each signal must have its own sequence, or *name*, and be carefully designed and tested to avoid interference with other sequences. In other words, there is no spatial isolation like electronic circuits have, so some form of code isolation is required. An analogy one could draw is making gates "wireless." Imagine a circuit with multiple AND and OR gates that communicate wirelessly; since there is no spatial isolation, one has to "tune" the output of a gate to the input of the next gate in the chain. This way, two gates that have the same function but are in different parts of the circuit can be differentiated during circuit evaluation. We propose a solution to this challenge in Section 4.

Leakage. Perhaps not surprisingly, DNA strand displacement systems have their own version of leakage. The issue stems partly from synthesis errors, which can lead to strands where bases are missing or incorrectly incorporated, and partly from the (unavoidable) random "opening" of base pairs at the end of helices. Under certain conditions, a strand may be displaced even if the input is not present, causing the reaction to spuriously propagate downstream. Conversely, inputs can get stuck without ever releasing an output strand, leading to signal attenuation.

Computation energy and non-reusable gates. Both inputs and gates are consumed as the circuit is evaluated by cascade reactions, so they cannot be reused. This may impose design constraints, such as the lack of feedback loop signals. Importantly, reactions are driven by the formation of additional base pairs (*e.g.*, in the toeholds) between inputs and gates, and the gates thus also act as a diminishing energy source.

Data encoding. Information is encoded not only in sequences but also in the concentration of signals (number of molecules/volume). The concentrations of signal species play a role similar to the voltage applied to a wire in an electronic circuit. A high concentration (above a threshold value) represents the logical "ON" state, while a low concentration (below a threshold value) represents the logical "OFF" state. In previous experiments [29, 41, 42, 23, 24], a typical gate or signal is present at concentrations of 10-100 nM, or in the order of 10^{10}-10^{11} copies in a reaction volume of 100 μl. Absolute numbers may be smaller for a computation that occurs inside a much smaller cellular volume.

Lack of clear hardware/software interface. The two previous challenges result in a new one: there is no clearly defined notion of hardware and software in these systems. Gates and systems come pre-programmed and pre-assembled for the specific computation they are designed to carry out. In addition, information and gates are both composed of the same substrate, *i.e.*, DNA sequences.

Speed of computation. The typical time to evaluate one level of logic can be on the order of minutes to hours. This is most often compatible with the time-scale of target applications: Diagnostic or therapeutic responses do not require nanosecond response times. Further, circuits are inherently asynchronous.

Need for dual-rail logic. In practice, all DNA strand displacement logic circuits built so far use dual-rail logic. This is because
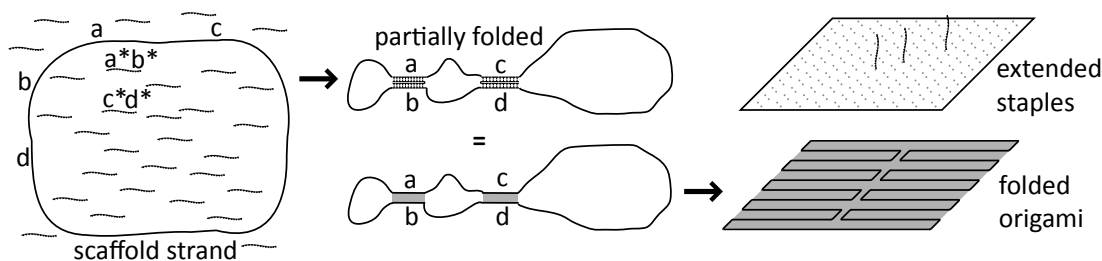
Figure 5: DNA origami is a technique for the self-assembly of molecular structures with nanoscale precision. A long, single-stranded DNA scaffold is mixed with short staple strands.
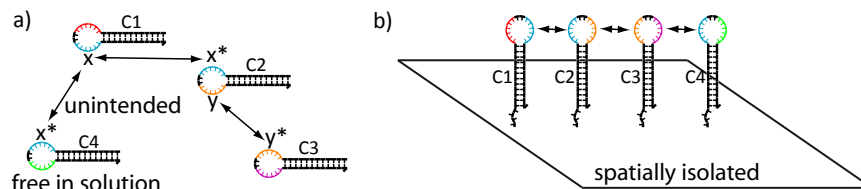
Figure 6: Possible circuit organizations. (a) Non-localized circuit. (b) Localized circuit on DNA origami.

it is not possible to distinguish if a signal is absent due to its being asynchronous, having a value of zero or having a value of one with a single molecule. Dual-rail logic makes NOT gates unnecessary, because a signal can be simply interpreted as its complement by a cascaded gate.

4. AN ARCHITECTURE FOR SPATIALLY ISOLATED DNA GATES

We propose a practical approach to DNA-based computation that takes advantage of spatial isolation and organization. In our approach, DNA structures that implement computational elements, such as logic AND and OR gates, are attached to a "molecular board," typically DNA origami. When a component is activated, it can interact with other components that are in close proximity on the same board, although interaction with components that are farther away is inhibited. A key advantage of this approach is that it minimizes crosstalk. It also lowers code pressure because possible interactions are now constrained simply by proximity, and the same signal sequence can be reused in multiple locations without any risk of interference. Furthermore, a diagnostic circuit using co-localized components may be easier to deliver to cells and is likely better protected from degradation by cellular enzymes [35, 16].

Figure 6 illustrates the concept of spatial isolation. In solution, hairpins C1, C2, C3 and C4 can freely interact with each other when open (Figure 6(a)). Specific, designed interactions between two components are achieved when each component carries a unique, identifying toehold sequence (or code). In this example, component C1 carries a toehold complementary to C2. However, C4 carries the same toehold, so crosstalk may occur if C1 opens and approaches C4 before approaching C2. Spatially organizing the same set of components on an origami substrate (Figure 6(b)) limits interaction to adjacent components; therefore, the duplicated toehold of component C4 can no longer interact with C1. Note that by localizing the reactions, a designer need not be concerned when reusing gate designs (and signal sequences) within the same board, which simplifies the design process and improves composability. Moreover, this makes the design process more closely resemble the design of electronic circuits. Spatial organization also increases the

speed of signal propagation. The rate of chemical reactions is effectively determined by the concentrations of the reactant species, and when the two reactants are co-localized, the effective "concentration" of one reactant around the location of the other is high.

A single DNA origami provides limited surface for placement of logic gates (about 100 hairpins). However, it is possible to arrange, align and bind multiple DNA origamis on a surface through recursive, multi-stage assembly [25, 37]. This provides sufficient precision to extend the computation to larger boards, further lowering code pressure. Another viable alternative is to use a hybrid approach that implements interactions between multiple boards via the release and capture of diffusible signal strands. A benefit of the hybrid approach is the higher tolerance to fabrication defects, erroneous signals or incomplete reactions. In both cases, the origami boards are themselves in diffusion, and there can be (and typically are) multiple copies of each circuit, allowing detections of the output from many identical tiles through their concentration.

We break up computation using our spatially isolated architecture into three basic stages: binding of diffusible inputs to input ports[1]; computational processing of the inputs, which occurs on the board surface using only a small number of hairpin types; and release of an output that can serve as an input to a downstream component, can act as a drug, or can be used for reading computation's result. Section 4.1 identifies the basic building blocks and processes of our spatially isolated circuits. Section 4.2 presents a half-adder circuit as an example of how to scale to larger, more complex circuits from the basic primitives.

4.1 Basic Building Blocks

Input translation. Inputs external to the circuit (external signals) are recognized by special input translation hairpins anchored in the origami (called *anchorages*). For example, hairpin $H(A_0, Y)$ in Figure 7 is designed to recognize input strand A_0 with domains sa_0. Binding of the strand opens the hairpin, producing an output with domains ys (an output repeated throughout the circuit), and

---

[1]Inputs in this paper are shown here at the edges of the board for simplicity, but complex circuits may take advantage of input ports in multiple locations.

mediates a further reaction to produce an input for the adjacent gate. In this example, $H(A_0, Y)$ represents a hairpin that is opened by an input with domain a_0 and releases an output with domain y. This notation system will be used to simplify the upcoming diagrams and better describe hairpin components[2].

Fuel. This free hairpin strand relays a message from one anchored hairpin to the next. In Figure 7, when translation hairpin $H(A_0, Y)$ is activated and opens, it reveals domains ys, which respectively complement the toehold and stem of a fuel $F(Y, X)$. $F(Y, X)$ is a hairpin activated by an input with domain y and output x. Hybridization causes the fuel to open, at which point the fuel's single-stranded part acts as an input for a subsequent hairpin.

It is possible to design a circuit where anchored hairpins communicate directly without fuel, but it is not practical for two major reasons. The first is a geometric issue: if the two hairpins intended to react are anchored, they have fewer degrees of freedom to move and are less likely to position themselves into an interaction-favorable configuration. The second is a manufacturing issue. DNA origamis, along with circuit strands, are manufactured in a single step by annealing all components: strands are first heated to disrupt hybridization and then gradually cooled, allowing the designed, stable duplex regions to form first. However, if circuit components were designed with complementary toeholds as well as stem regions, binding could occur directly between hairpins. With a fuel that is added separately, anchored hairpins can be manufactured without any risk of unintended interactions.

Transmission line. This consists of a series of identical hairpins designed to relay the signal across a pre-designed path. Figure 7 illustrates the process of input translation and transmission. The input A_0 initially hybridizes to the translation gate (1), followed by the binding of the fuel (2), as described previously. The single-stranded region of the fuel, domains sx, acts as an input to the adjacent hairpin, $H(X, Y)$, opening it to reveal domains ys (3). Binding of a new $F(Y, X)$ fuel produces another identical input sx for the next hairpin, and the process repeats along the transmission line until another block is reached (not shown).

Cross-over path. Larger circuits may require transmission lines to cross so that signals can travel to their destination. If they are simply allowed to cross without any safety measures, they may interfere with one another because they use the same type of transmission gate. However, mitigating this problem requires only the temporary substitution of two new domains into one of the transmission lines and one new fuel. The sequences and design can be reused in multiple crossings. Figure 8 shows how a cross-over path is implemented. The diagonal path is a regular transmission line, as explained above. The horizontal path is a modified transmission line. As the horizontal transmission line approaches the crossing, it goes through a translation step via an anchored hairpin $H(X, J)$. This enables transmission using a new fuel $F(J, I)$ and a repeating hairpin, $H(I, J)$. Once past the crossing region, the transmission line goes through a second translation step via another anchored hairpin $H(I, Y)$, at which point it goes back to using the original fuel and anchored hairpins.

AND gate. This consists of two input hairpins[3], a threshold hairpin and an output hairpin. The threshold hairpin is designed to mimic

---

[2]For clarity, gate explanations in this section denote inputs as if they were external inputs, but they could just as well be internal inputs.

[3]Although AND gates with larger numbers of inputs are possible, we do not illustrate them here for simplicity.

the output hairpin, but it produces an output sequence that has no complement in the system. Positioned closer to the input hairpins than the output hairpin, the threshold is more likely to react with the hairpin of the first incoming input. An example is shown in Figure 9. When the first input A_0 arrives (1), it opens $H(A_0, Y)$, followed by fuel $F(Y, X)$ (2), which opens the threshold hairpin $H(X, -)$ (3) since this hairpin is closer. The output from this first stage is inert. When the second input B_0 arrives (4), the signal is transmitted from the input hairpin to the fuel (5), which can freely interact with $H(X, Y)$ since the threshold is already bound to the first signal. Once $H(X, Y)$ is opened (6), the output can be transmitted to further components.

OR gate. This is similar to the AND gate, but it has no threshold hairpin: For OR gates, any of the two inputs should be sufficient to expose the output. Figure 10 illustrates the process. Arrival of either strand A_0 (2a) or B_0 (2b) opens its corresponding input hairpin, which binds to a fuel $F(Y, X)$, transmitting the signal to the next hairpin $H(X, Y)$.

Fan-out. This gate forks an output into multiple gates so a single output can be reused as input for more than one downstream gate. The operating principle is to have a fan-out hairpin per desired output. An input comes in, reacts with a fan-out hairpin, and then detaches itself, becoming available for interaction with another fan-out. Figure 11 illustrates a configuration with a fan-out of two and details the reaction with one of the fan-out hairpins. An open fuel strand from a previous transmission step (1) opens the fan-out hairpin $H(X, ZY)$ (2), freeing the domains zsy. Domains zsy then bind to fuel $F(ZY, Y)$, specific to the fan-out hairpins, opening it and exposing sys (3). Finally, sys displaces the attached input through competition of the s domain, leaving a single-stranded ys hanging off the fan-out hairpin (4), which can now open a fuel $F(Y, X)$ that opens the next hairpin $H(X, Y)$. This also frees the input to interact with another fan-out hairpin (5 — compare to 1).

Producing outputs. Once the circuit has finished its computation, it needs mechanisms to communicate its results or to actuate in the "external world". External communication (with humans) is typically provided via fluorescence, discussed in Section 5. Communication with other circuits and simple actuation is achieved by releasing certain sequences. More complex actuation requires these sequences to trigger the operation of other types of DNA machines (*e.g.*, a DNA-based mechanism that releases a drug).

4.2 From Gates to Circuits: A Half-adder Example

It is possible to build more complex circuits from the primitives described above. We now illustrate how a half-adder can be built with localized strand displacement. Even though other circuits may be more relevant to in-cell computation, we chose the half-adder example because it is a circuit familiar to computer designers. This simplifies our discussion of concepts and methodology.

The process is similar to designing circuits with electronic gates. First, the logical circuit is designed and physically laid out. Figure 12 shows multiple representations of a half-adder circuit. Representation (a) is its final layout on a DNA origami board. Note that about half of the board is unused and could host additional components. Representation (b) is the equivalent dual-rail logic circuit. Only one pair of (A_0, B_0), (A_0, B_1), (A_1, B_0), (A_1, B_1) will be present, so only one of the AND gates activates and generates the corresponding outputs. Representation (c) also shows the circuit at the level of anchored and fuel hairpins, with corresponding input
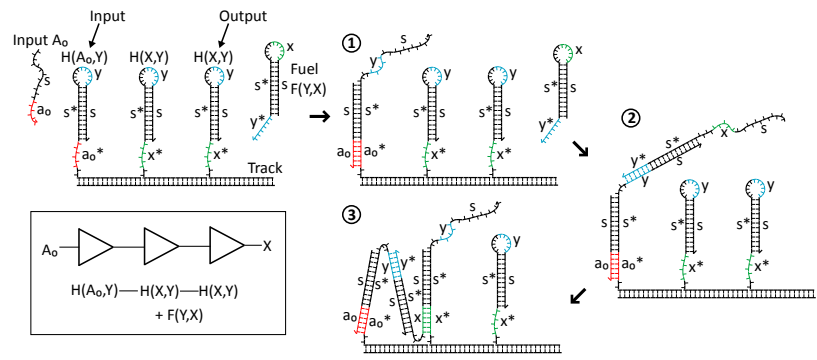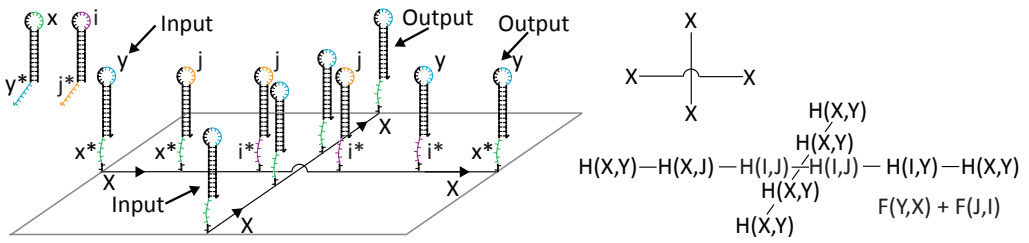
Figure 7: Signal transmission over two hairpins.

Figure 8: Signal cross-over in localized DNA circuits.
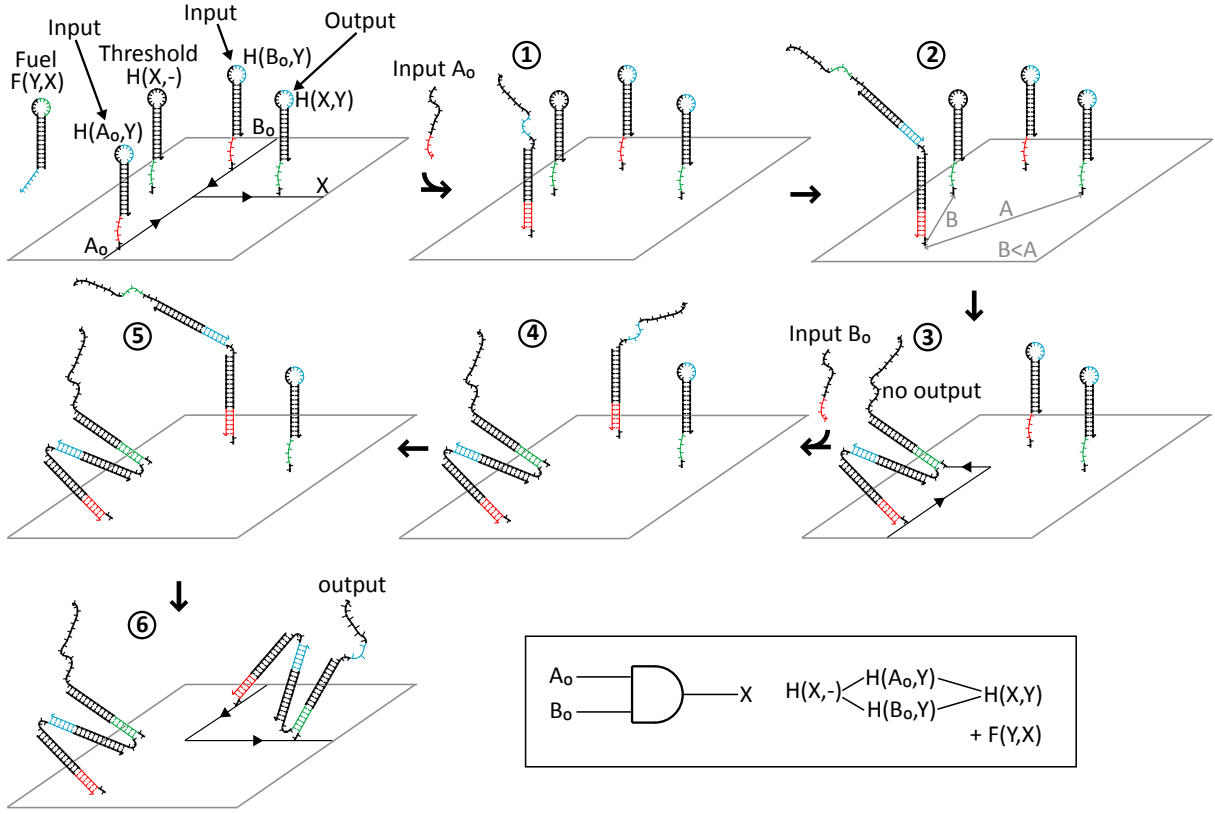

Figure 9: AND logic with two inputs, A_0 and B_0 (top left diagram, diagonal opposites in origami), and one output, X (top left diagram, right side).

and output signals in each of the hairpins. From left to right, the first three hairpin columns represent the AND gates (threshold, inputs and output hairpins). The fourth column represents the fan-out hairpins, at which point a switch to a new pair of domains is used to prevent backward transmission of the signal; otherwise, the output of a fan-out hairpin could interact with untriggered hairpins within reach in the third column. From that point on, output signals are either readily available in the next hairpin or transmitted farther down. Representation (d) shows two dual-rail half-adders combined into a full dual-rail adder.

5. EVALUATION

This section assesses: (1) the complexity reduction and performance improvement of localized circuits compared to non-localized ones, and (2) whether our proposed design works well in practice.

5.1 Assessing Complexity Reduction and Performance Improvement

We compare circuits built from our localized components to equivalent non-localized circuits based on the gates of Soloveichik *et al.* [32]. We expect that results would be similar for other gate designs [29, 23, 4]. We use two main criteria for this comparison, namely, the number of different sequences required to encode all signals and circuit components, and the typical time it takes to complete a computation. Reducing the number of required sequences is beneficial to modularity because it practically eliminates concerns over unanticipated interference between the multiple distinct sequences used in the circuit. In non-localized circuits, every sequence may interact with every other sequence, so the complexity of verifying whether the circuit is interference-free grows with the square of the number of sequences used. In localized circuits, sequences may interact only with other sequences anchored nearby and with molecules in solution, such as fuels, inputs and outputs. Thus, the verification complexity of localized circuits is not affected by the size of the circuit itself and grows only with the square of the number of inputs and outputs (fuel sequences are reused across many gates, so only a few types are required regardless of circuit size). As in electronic circuits, the computation time in strand displacement circuits is primarily determined by the depth of the circuit [28], here meaning the longest path of sequential strand displacement reactions that needs to occur for a computation to complete.

In counting sequences, we distinguish between inputs that are externally supplied and signals that are internal to the circuit. Furthermore, we count only the number of unique signals needed to design each circuit, not the number of physical DNA strands, because only the signal sequences are unique. All other strand sequences are determined by the sequences of these signals. Signals in the non-localized circuit have the domain structure *short:long:short*; signals in the localized circuit have the domain structure *short:long*, except for one signal produced during the fan-out stage that also has the structure *short:long:short*.

Table 1 shows the number of domains required to implement a variety of circuits. As a first example, we consider a linear transmission line. The simplest case corresponds to a single logical repeater gate, which is the best case proxy of a circuit with depth one. Likewise, chains of transmission gates are the simplest possible circuits of depth n. Thus, we evaluate both a single transmission gate and a chain of n transmission gates. In solution, a single repeater can be implemented by the two-stage reaction cascade shown in Figure 2. The design of a repeater thus requires the design of three signal strands: the input, the output, and the internal signal that connects the two stages of the cascade. In a cascade of

n repeaters in solution, each additional repeater requires the design of two new signals. Both the number of sequences and the typical completion time thus scale linearly with the number of repeaters in the sequence.

	Trns(1)	Trns(n)	AND(1)	AND(n)	Half-addr	Full-addr	N-bit addr
Non-localized	3	2n+1	4	3n+1	12	24	12+16n
Localized	3	3	4	n+3	13	15	15+6n

Table 1: Number of domains required to implement various circuits.

In the localized circuit, a repeater similarly requires three distinct signals: the external input as, an internal signal ys (that enables fuel capture), and an intermediate output sequence xs. The sequence s is determined by the input, and only the toehold sequences change. Unlike the diffusive case, the same two sequences (ys and xs) are used in every additional layer, so the number of sequences needed is independent of the cascade's length. Computation time scales linearly with the number of sequential repeaters, as observed for the diffusion case. Note, however, that co-localization of components can dramatically enhance reaction speed simply by increasing the effective concentration of one reactant at the site of the other (*e.g.*, see Chandran *et al.* [6] for a more detailed discussion). In our design, fuel capture should be the rate-limiting step, while the activation of an output by an open fuel should be fast. Thus, compared to the case where all communication is diffusive, each step in the reaction can potentially be accelerated by a constant factor.

Next, we compare the non-localized AND gate by Soloveichik *et al.* [32] with the localized gate introduced in Section 4.1, again both individually and in chains of n gates. The non-localized AND gate requires four independent signals: two inputs, one output, and one internal signal strand that ensures independence of inputs and outputs. Every additional layer in a linear cascade requires three additional sequences: one input, one output and one internal signal. The second input is provided by the upstream gate. As shown in Figure 9, an AND gate using localized components requires four different signals, two signals corresponding to external inputs and two internal signals. All information about the logic is encoded in the spatial arrangement of the hairpins. Every additional layer requires one additional external input but no new internal sequences. In the non-localized gate, input binding and output release are implemented as three sequential strand displacement reactions: The second input can bind only if the first input is present. In contrast, in the localized gate, binding of the two inputs is independent and can occur in parallel. Its latency is proportional to only two sequential strand displacement reactions. This inherent parallelism can enhance reaction speed by a constant for each AND gate.

We next compare working circuits: a half-adder, a full adder, and an n-bit full adder. If we use dual-rail logic and choose a half-adder design based on the truth table, we can implement a half adder with four AND gates followed by fan-out to produce two output signals (carry and sum) from each gate. In the non-localized case, this requires a total of 16 unique signals: 4 inputs, 8 internal signals and 4 outputs. The localized case requires 13 signals: 4 inputs, 4 outputs and 5 internal signals for logic gates, fan-out and crossover (see Figure 12). A full adder accepts one additional external input, which, in a dual-rail circuit, requires two more signals. Then, we need four additional AND gates, four fan-out units and a dual-rail OR gate that connects the two half-adders. For the localized full adder (Figure 12, (d)), sequences can be reused for all 8 AND and OR gates, fan-out gates and additional "wire" crossings, bringing
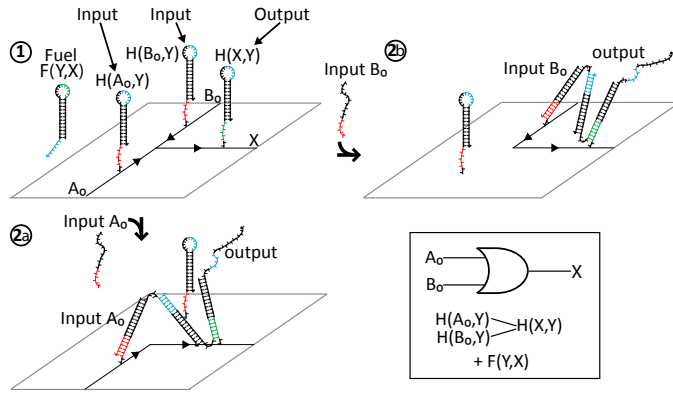
Figure 10: OR logic with two inputs, A_0 and B_0 (top left diagram, diagonal opposites in origami), and one output X (top left diagram, right side).
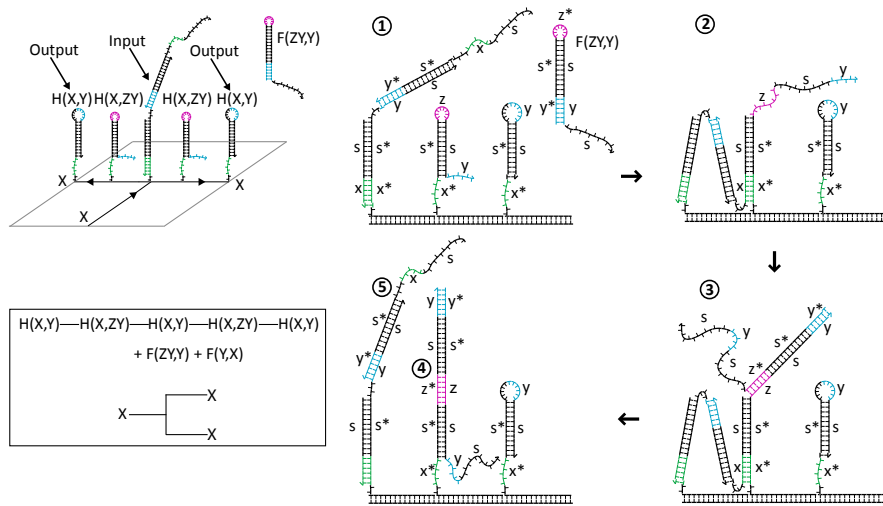

Figure 11: Fan-out gate with one input (top left diagram, middle hairpin) and two outputs (top left diagram, left and right ends).
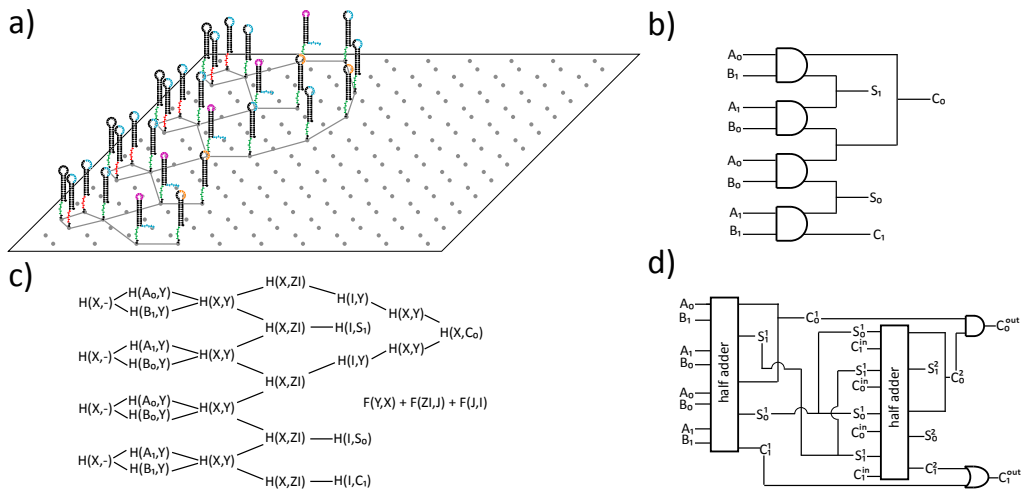

Figure 12: Multiple representations of a half-adder circuit on an origami tile ((a) through (c)). As shown in (d), another half-adder and simple logic can be added to make a full adder.

the total number of distinct sequences to 15. In contrast, for the non-localized case, we need 8 additional internal signals to implement the AND gate and fan-out as well as 2 additional signals for the OR gate. In an n-bit adder, each additional bit contributes 4 new input and 2 output sequences. The non-localized case requires 10 new internal sequences (and thus 16 total) for each adder; for the localized case, the number of sequences grows only because of external inputs and additional outputs (and thus only 6 new sequences per additional bit).

The trade-off is likely less favorable in terms of circuit speed. In order to connect inputs and outputs in the localized case, we may need considerable amounts of "wiring," *i.e.*, transmission lines that are not necessary in the diffusive case. This suggests that an optimal approach combines localized circuit elements that fit conveniently on a single origami, *e.g.*, a single adder, with diffusive signals that enable communication between origamis. We intend to explore this trade-off in future work.

5.2 Assessing Practicality

Even though simulation tools for strand displacement systems are becoming increasingly sophisticated [19, 13], experimental validation is still necessary. For example, there is currently no simulation tool capable of reliably simulating hairpin-based systems. This also means that many systems parameters — such as the lengths of various domains — are best determined through experimental testing. Furthermore, all experimental strand displacement systems exhibit some degree of undesirable side reactions, which can be due to errors in the DNA synthesis, imperfect complex assembly, or bad sequence design. The main challenge in evaluating any DNA system is properly determining system parameters (in this case domain lengths and hairpin spacing) and characterizing components (kinetic rates and yield), which requires many repeats under varying experimental conditions and design iterations, each taking on the order of hours to run.

Figure 13 presents experimental data for one of our proposed design elements, a three-step transmission line that includes a translation step. The input strand is designed to open the first hairpin, triggering fuel binding and subsequent opening of the second and third hairpins, similar to Figure 7. The final step is to detect that the reaction has happened. This is accomplished by using a fluorophore probe, a duplex that initially has a fluorophore attached to the end of one of its strands and a quencher attached to the end of the other strand, adjacent to the fluorophore. The proximity of a quencher prevents fluorescence. When the transmission line releases its output, it displaces the strand attached to the quencher and activates the probe to produce a detectable fluorescence signal. The solution used in the experiment has multiple copies of the transmission line, so fluorescence gradually increases over time as more probes are activated, as shown in Figure 13.

The three hairpins and the strand that connects them were mixed at equal concentration in solution (a buffer, TAE, consisting of Tris base, Acetate acid, EDTA and 12.5 nM magnesium (Mg^{2+}) that improves hybridization by maintaining a pH and salt concentration similar to physiological conditions). They were annealed to form the complete transmission line. Separately, the two strands that form the probe were annealed in a similar manner. The fuel was annealed separately, as well. Four separate experiments were performed: in each, the transmission lines were mixed with an equal quantity of probe and then mixed with: no further strands ($Track$ curve); input only ($Track + Input$); fuel only ($Track + Fuel$); or a combination of both fuel and input ($Track + Fuel + Input$). Each of these elements represents a separate experiment and is denoted by a separate curve on Figure 13 (right). The transmission

line and probe do not interact in isolation, which is confirmed by the fact that the addition of either an individual input or fuel results in little increase in fluorescent signal. A combination of both input and fuel increases fluorescence, which indicates that the signal has been propagated from one end of the track to the other because that is the only way to open the probe. Interaction between localized components in the field of synthetic DNA motors has demonstrated that the rate of interaction between two localized components is much higher than interactions between freely diffusing components [2, 17]. Given the similar length scales, we attribute the signal to interactions within the same track as opposed to interaction between different tracks.

5.3 Discussion

Performance. Computation time for biochemical circuits can be shown to scale linearly with circuit depth. The delay is shorter for co-localized components compared to diffusive ones. Simulation of the interaction rate between two linked objects shows a speedup of several orders of magnitude. This becomes more apparent with lower circuit concentrations, as the localized interactions are independent of concentration. Our experiments show that non-localized gates are slower and have lower yield compared to localized gates. Provided fuel is abundant, computation rate scales linearly with number of layers. In practice, it is not yet clear how much delay can be further reduced for localized DNA components, but it is safe to say that computation times will be in minutes to hours, which is mostly appropriate for the kinds of applications they target.

In-cell environment. The environment of a cell can potentially be hostile to DNA-based circuits. Transitions are the main worry: many molecules in cells bind to DNA, which may inhibit toehold formation or branch migration, equally affecting designed and leakage interactions. Separate studies show that DNA structures are stable in cell lysate (solution produced when cells are destroyed by disrupting their cell membranes) and within cells. A different nucleic acid (locked nucleic acid, a form of modified RNA) resistant to nucleases could be used. Dilution due to cell division should also be considered, but the division is long enough ($\sim$1 day for mammalian cells in culture) for several sequential reactions to complete.

6. RELATED WORK

Several groups have used strand displacement to engineer feed-forward Boolean logic circuits of increasing complexity [33, 29, 23]. Chandran *et al.* [6] have proposed a design for strand displacement-based Boolean logic circuits with spatially localized components. The paper contains an interesting discussion of the potential speed-up that could be achieved in a circuit where all components are localized. However, unlike the present work, the authors do not attempt to reduce code pressure through the repetition of sequence domains. Outputs produced in one reaction step need to physically reach the next gate with the proper orientation to serve as input, a key issue not fully addressed. Furthermore, no experimental verification of the proposed mechanisms is discussed.

Multiple groups showed that DNA nanostructures can serve as nanoboards for the controlled placement of proteins, gold nanoparticles, carbon nanotubes, quantum dots and various other components with interesting chemical, optical, and electronic properties (reviewed in the literature [20, 34]). In particular, Pistol *et al.* used a DNA lattice to arrange a nanoscale computational circuit with optical components [22, 21]. In contrast, our proposal organizes DNA-based information processing mechanisms in a DNA origami.
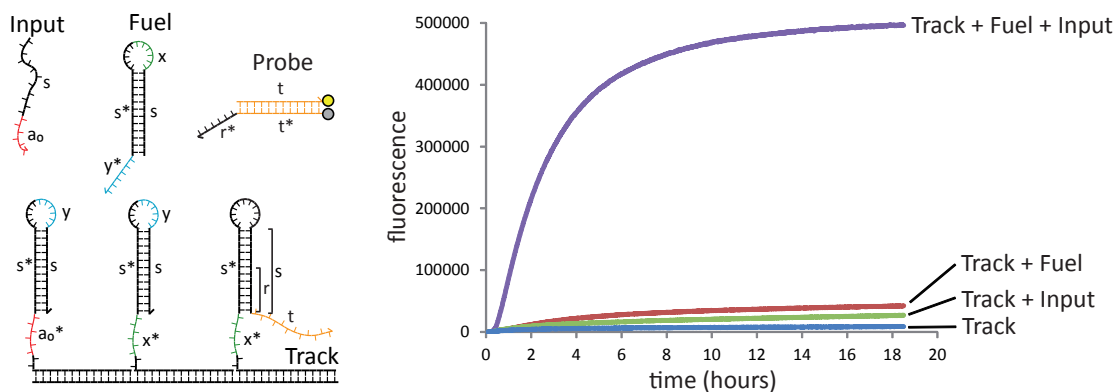
Figure 13: Three-anchorage transmission line (left). An input strand is designed to initiate signal transmission along the track. Activation of a fluorescent probe by the final anchorage allows detection of the transmitted signal (right). When mixed with fuel, input or no strands at all, minimal output is detected. A combination of both input and fuel results in a signal increase, as designed.

Pierce and collaborators used DNA hairpin motifs similar to the one used in the present work for the programmed self-assembly of linear and branched polymer structures [8, 38]. Autonomous propagation of a DNA signal along a track has been achieved in the area of molecular motors, where a DNA "cargo" is moved between track locations. Movement is powered by the turnover of a DNA fuel hairpin and strand displacement [10, 2, 18, 17] or the cleaving of a DNA backbone [15, 36]. Such systems rely on interactions between localized track components to keep the cargo moving along the same track in a single direction. However, the nature of using a single cargo means no fan-out may be achieved. All of these papers focus on molecular motion and not on computation. Still, they provide further support for the experimental approach outlined in this paper.

Our long-term goal is to develop circuits that can be used for diagnostic and therapeutic applications in living cells. Many of the ingredients for such an approach are already available: DNA hairpins have recently been used to enable multiplexed RNA imaging with sub-cellular resolution [7]. Origami-based biosensor arrays [11] are being developed, and delivery of complex nanostructures to cells has been demonstrated [16]. The technology introduced in this paper provides the computation that can connect the sensing of disease markers to a targeted therapeutic response.

7. CONCLUSIONS

This paper introduced DNA strand displacement computing to the computer architecture community, discussed major associated architectural challenges, and proposed practical localized DNA circuits. We evaluated the benefits of localization and showed it to be a key improvement that enables composability and simplifies the design of larger circuits. With this work, we have taken another important step in the long journey toward in-cell computation. The field is exciting and new, and fascinating challenges lie ahead. We hope this paper motivates other computer architects to apply their expertise toward tackling these challenges.

Acknowledgments

References

[1] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021, 1994.

[2] J. Bath, S. Green, K. Allen, and A. Turberfield. Mechanism for a directional, processive, and reversible DNA motor. *Small*, 5(13):1513–1516, 2009.

[3] L. Cardelli. Strand algebras for DNA computing. *DNA Computing and Molecular Programming*, pages 12–24, 2009.

[4] L. Cardelli. Two-domain DNA strand displacement. *Mathematical Structures in Computer Science*, 23(02):247–271, 2013.

[5] R. Carlson. The changing economics of DNA synthesis. *Nature Biotechnology*, 27(12):1091–1094, 2009.

[6] H. Chandran, N. Gopalkrishnan, A. Phillips, and J. Reif. Localized hybridization circuits. *DNA Computing and Molecular Programming*, pages 64–83, 2011.

[7] H. Choi, J. Chang, L. Trinh, J. Padilla, S. Fraser, and N. Pierce. Programmable in situ amplification for multiplexed imaging of mRNA expression. *Nature Biotechnology*, 28(11):1208–1212, 2010.

[8] R. Dirks and N. Pierce. Triggered amplification by hybridization chain reaction. *Proceedings of the National Academy of Sciences of the United States of America*, 101(43):15275, 2004.

[9] S. Douglas, I. Bachelet, and G. Church. A logic-gated nanorobot for targeted transport of molecular payloads. *Science Signalling*, 335(6070):831, 2012.

[10] S. Green, J. Bath, and A. Turberfield. Coordinated chemomechanical cycles: a mechanism for autonomous molecular motion. *Physical Review Letters*, 101(23):238101, 2008.

[11] Y. Ke, S. Lindsay, Y. Chang, Y. Liu, and H. Yan. Self-assembled water-soluble nucleic acid probe tiles for label-free RNA hybridization assays. *Science*, 319(5860):180, 2008.

[12] Y. Krishnan and F. Simmel. Nucleic acid based molecular devices. *Angewandte Chemie International Edition*, 50(14):3124–3156, 2011.

[13] M. Lakin, S. Youssef, L. Cardelli, and A. Phillips. Abstractions for DNA circuit design. *Journal of The Royal Society Interface*, 9(68):470–486, 2012.

[14] Q. Li, G. Luan, Q. Guo, and J. Liang. A new class of homogeneous nucleic acid probes based on specific displacement hybridization. *Nucleic Acids Research*, 30(2):e5–e5, 2002.

[15] K. Lund, A. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M. Stojanovic, N. Walter, et al. Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, 2010.

[16] Q. Mei, X. Wei, F. Su, Y. Liu, C. Youngbull, R. Johnson, S. Lindsay,

H. Yan, and D. Meldrum. Stability of DNA origami nanoarrays in cell lysate. *Nano Letters*, 11(4):1477–1482, 2011.

[17] R. Muscat, J. Bath, and A. Turberfield. A programmable molecular robot. *Nano Letters*, 11(3):982, 2011.

[18] T. Omabegho, R. Sha, and N. Seeman. A bipedal DNA brownian motor with coordinated legs. *Science*, 324(5923):67, 2009.

[19] A. Phillips and L. Cardelli. A programming language for composable DNA circuits. *Journal of the Royal Society Interface*, 6(Suppl 4):S419–S436, 2009.

[20] A. Pinheiro, D. Han, W. Shih, and H. Yan. Challenges and opportunities for structural DNA nanotechnology. *Nature Nanotechnology*, 6(12):763–772, 2011.

[21] C. Pistol, W. Chongchitmate, C. Dwyer, and A. Lebeck. Architectural implications of nanoscale-integrated sensing and computing. *Micro, IEEE*, 30(1):110–120, 2010.

[22] C. Pistol, C. Dwyer, and A. Lebeck. Nanoscale optical computing using resonance energy transfer logic. *Micro, IEEE*, 28(6):7–18, 2008.

[23] L. Qian and E. Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196, 2011.

[24] L. Qian, E. Winfree, and J. Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011.

[25] A. Rajendran, M. Endo, Y. Katsuda, K. Hidaka, and H. Sugiyama. Programmed two-dimensional self-assembly of multiple DNA origami jigsaw pieces. *ACS Nano*, 5(1):665–671, 2011.

[26] P. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006.

[27] V. J. Schüller, S. Heidegger, N. Sandholzer, P. Nickels, N. Suhartha, S. Endres, C. Bourquin, and T. Liedl. Cellular immunostimulation by CpG-sequence-coated DNA origami structures. *ACS Nano*, 5(12):9696–9702, 2011.

[28] G. Seelig and D. Soloveichik. Time-complexity of multilayered DNA strand displacement circuits. *DNA Computing and Molecular Programming*, pages 144–153, 2009.

[29] G. Seelig, D. Soloveichik, D. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585, 2006.

[30] N. Seeman. Nanomaterials based on DNA. *Annual Review of Biochemistry*, 79:65–87, 2010.

[31] W. Shih and C. Lin. Knitting complex weaves with DNA origami. *Current Opinion in Structural Biology*, 20(3):276–282, 2010.

[32] D. Soloveichik, G. Seelig, and E. Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393, 2010.

[33] K. Takahashi, S. Yaegashi, A. Kameda, and M. Hagiya. Chain reaction systems based on loop dissociation of DNA. *DNA Computing: 11th International Workshop on DNA Computing, DNA11, Lecture Notes in Computer Science*, 3892:347–358, 2006.

[34] C. Teller and I. Willner. Organizing protein–DNA hybrids as nanostructures with programmed functionalities. *Trends in Biotechnology*, 28(12):619–628, 2010.

[35] A. Walsh, H. Yin, C. Erben, W. M.J., and A. Turberfield. DNA cage delivery to mammalian cells. *ACS Nano*, 5(7):5427–5432, 2011.

[36] S. Wickham, M. Endo, Y. Katsuda, K. Hidaka, J. Bath, H. Sugiyama, and A. Turberfield. Direct observation of stepwise movement of a synthetic molecular transporter. *Nature Nanotechnology*, 6(3):166–169, 2011.

[37] S. Woo and P. Rothemund. Programmable molecular recognition based on the geometry of DNA nanostructures. *Nature Chemistry*, 3:620–627, 2011.

[38] P. Yin, H. Choi, C. Calvert, and N. Pierce. Programming biomolecular self-assembly pathways. *Nature*, 451(7176):318–322, 2008.

[39] B. Yurke and A. Mills. Using DNA to power nanostructures. *Genetic Programming and Evolvable Machines*, 4(2):111–122, 2003.

[40] D. Zhang and G. Seelig. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature Chemistry*, 3(2):103–113, 2011.

[41] D. Zhang, A. Turberfield, B. Yurke, and E. Winfree. Engineering entropy-driven reactions and networks catalyzed by DNA. *Science*, 318(5853):1121, 2007.

[42] D. Zhang and E. Winfree. Control of DNA strand displacement kinetics using toehold exchange. *Journal of the American Chemical Society*, 131(47):17303–17314, 2009.