Digital Logic with Molecular Reactions

Hua Jiang, Marc D. Riedel, Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota
200 Union St. S.E., Minneapolis, MN 55455
{hua, mriedel, parhi}@umn.edu

***Abstract*—This paper presents a methodology for implementing digital logic with molecular reactions based on a bistable mechanism for representing bits. The value of a bit is not determined by the concentration of a single molecular type; rather, it is the comparison of the concentrations of two complementary types that determines if the bit is "0" or "1". This mechanism is robust: any small perturbation or leakage in the concentrations quickly gets cleared out and the signal value is not affected. Based on this representation for bits, a constituent set of logical components are implemented. These include combinational components – AND, OR, NOR, and XOR – as well as sequential components – D latches and D flip-flops. Using these components, three full-fledged design examples are given: a square-root unit, a binary adder and a linear feedback shift register. DNA-based computation via strand displacement is the target experimental chassis. The designs are validated through simulations of the chemical kinetics. The simulations show that the molecular systems compute digital functions accurately and robustly.**

I. Introduction

Just as electronic systems implement computation in terms of voltage (*energy per unit charge*), molecular systems compute in terms of chemical concentrations (*molecules per unit volume*). Indeed, the field of *molecular computation* strives for molecular implementations of computational processes – that is to say processes that transform input concentrations of molecular types into output concentrations of molecular types [1], [2], [3], [4], [5], [6].

The impetus of the field is not computation *per se*; chemical systems will never be useful for number crunching. Rather the field aims for the design of custom, embedded biological "sensors" and "controllers" – viruses and bacteria that are engineered to perform useful tasks *in situ*, such as cancer detection and drug therapy. Exciting work in this vein includes [7], [8], [9], [10].

There have been several attempts to apply concepts from digital circuit theory to biological engineering. The view that the presence of a type of molecule, such as a protein, corresponds to logical one and its absence corresponds to logical zero, is contained in much of this prior work, either explicitly or implicitly. Numerous types of *genetic gates* have been proposed [11], [12], [13], [14], [15], [16], [17], [18]. Also, dating back to seminal work by Kauffman, gene networks are often modeled as directed graphs in which there is an arrow from one node to another if and only if there is a causal link between the corresponding genes; the node itself is viewed as a Boolean function of its inputs; its state is either "on" or "off" depending on the level of gene expression [19].

Prior work has established general mechanisms for molecular computation [20], [21], [22] as well as specific computational constructs: logical operations such as copying, comparing, incrementing/decrementing as well as programming constructs such as "for" and "while" loops [23]; arithmetic operations such as multiplication, exponentiation and logarithms [24], [25]; and signal processing operations such as filtering [26].

In this paper, we present a novel methodology for implementing digital logic with molecular reactions based on a bistable mechanism for representing bits. The notion of bistability is not new [27]. We apply it in a novel way, with a form of "dual-rail" encoding: the value of a bit is not determined by the concentration of a single molecular type. Rather, it is the comparison of the concentrations of two complementary types that determines if the bit is "0" or "1". This mechanism is robust: any small amount of perturbation or leakage in the concentrations quickly gets cleared out and the signal value is not affected. Based on this bit representation, we present designs for combinational components – AND, OR, NOR, and XOR – as well as for sequential components – D latches and D flip-flops. We illustrate the use of these components with three full-fledged design examples: a square-root unit, a binary adder, and a linear feedback shift register (LFSR) [28]. We validate the designs through simulations of the molecular kinetics, based on ordinary differential equations. The simulations results show that our gates, our square-root unit, our adder, and our LFSR produce nearly perfect digital signal values.

The paper is organized as follows. In Section II, we present some general background information on the computational model and simulation techniques for molecular systems. In Section III, we describe the bistable mechanism for representing binary bits. In Section IV, we discuss the implementation of logic gates. In Section V, we discuss the implementation of D latches and D flip-flops. In Section VI, we present the examples of a binary adder and an LFSR. Finally, in Section VII, we provide concluding remarks.

This work is supported by an NSF Grant, #1117168.

II. Computational Model and Biochemical Background

A. Technology-Independent Model

One of the great successes of integrated circuit design has been in abstracting and scaling the design problem. The physical behavior of transistors is understood in terms of differential equations – say, with models found in tools such as SPICE [29]. However, the design of circuits occurs at more abstract levels – in terms of switches, gates, and modules. Many analogous levels of abstraction exist for biological systems. These range from molecular dynamics, to protein networks, to genetic regulatory networks, to signaling pathways, to complete cellular systems, to multicellular organisms.

We will discuss a particular level of abstraction, analogous in some ways to transistor netlists: molecular reactions. We will examine the abstraction from a design perspective: how can we synthesize molecular reactions that produce specific output concentrations of molecules as *a function* of input concentrations?

A molecular system consists of a set of chemical reactions, each specifying a rule for how types of molecules combine. For instance,

$$X_1 + X_2 \xrightarrow{k} X_3, \tag{1}$$

specifies that one molecule of X_1 combines with one molecule of X_2 to produce one molecule of X_3. The value k is called the *rate constant*. We model the molecular dynamics in terms of *mass-action kinetics* [30], [31]: reaction rates are proportional to (1) the concentrations of the participating molecular types; and (2) the rate constant. Accordingly, for the reaction above, the rate of change in the concentrations of X_1, X_2 and X_3 is

$$-\frac{d[X_1]}{dt} = -\frac{d[X_2]}{dt} = \frac{d[X_3]}{dt} = k[X_1][X_2], \tag{2}$$

(here $[\cdot]$ denotes concentration). Most prior schemes for molecular computation depend on specific values of the rate constants, which limits the applicability since the rate constants are not constant at all; they depend on factors such as cell volume and temperature. The results of the computation are not robust.

We aim for robust constructs: in our methodology we require only a coarse value for the kinetic constants. Given the coarse value for these constants, the computation is exact. It does not matter how fast the reactions are – only that all reactions fire at relatively similar rates. Accordingly, we will generally omit the kinetic constants from the specification of our reactions.

B. Technology Mapping

Given a specification of an abstract molecular reaction network that implements the requisite computation, the next step is to map it to specific molecular reactions. We describe a mapping to DNA strand-displacement reactions. The reader is referred to [5] for a detailed discussion of this mechanism. Here we illustrate with an example.

Consider the DNA strand-displacement reaction shown in Figure 1. Here a single strand of DNA X_1 replaces the top strand of a double-strand DNA L_i; this generates a double-strand DNA H_j and a single-strand B_j. (This reaction is reversible.) The top strand of H_j can be replaced by single-strand X_2, generating a single-strand O_j. Then O_i replaces two of the top strands of the double-strand T_i, releasing X_3. (Note that the strands L_i, G_i and T_i are "fuel" sources. It is assumed that there is an abundant source of these; the concentrations do not matter.) The signals are the concentrations of X_1, X_2 and X_3. This sequence of strand displacements implements the abstract chemical reaction:

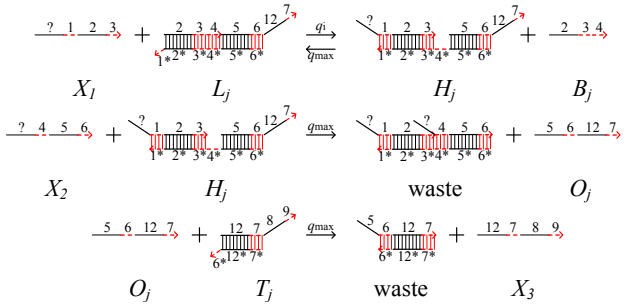

Fig. 1: An example of DNA strand displacement.

$$X_1 + X_2 \xrightarrow{k} X_3,$$

In [5] it is demonstrated that *any* system consisting of bimolecular reactions i.e., reactions with two reactants each, can be mapped to such DNA strand-displacement reactions. All of our designs consist of bimolecular reactions.

C. Simulation & Validation

Given a set of reactions at abstract molecular level, we first map it to DNA strand-displacement reactions of the form shown Figure 1. We then generate system kinetic equations of the mapped DNA system and obtain the transient solution. Such simulations of the chemical kinetics provide a reasonably accurate prediction of the actual *in vitro* behavior [6].

III. Bit Representation

The most straightforward interpretation of binary values in the context of molecular computation is to assign a threshold to the concentration of a designated molecular type [32]. When the concentration exceeds a threshold level, the bit is considered a logical 1; otherwise it is consider a logical 0. Although such a representation is conceptually simple, it requires external mechanisms for comparing the concentration of the designated molecular type with the threshold. Furthermore, it suffers from signal degradation over time: unwanted residue accumulates every time a signal is changed, unless there is some mechanism to clear the signal.

To mitigate these issues, we use a *complementary representation* (reminiscent of a "dual-rail" encoding). For a single bit X, we use two molecular types, X_0 and X_1. The presence of X_0 indicates that X is set to 0; the presence of X_1 indicates that X is set to 1. Clearly, X_0 and X_1 should not be present at the same time or else the value of X would be ambiguous. We use following set of reactions to ensure that this does not happen:

$$\begin{aligned} X_0 + X_1 &\longrightarrow S_X \\ S_X + X_0 &\longrightarrow 3X_0 \\ S_X + X_1 &\longrightarrow 3X_1. \end{aligned} \tag{3}$$

In Reactions 3, a molecule of X_0 combines with a molecule of X_1 to produce a molecule of S_X. This molecule of S_X then combines with a molecule of X_0 or one of X_1, depending on which it meets first. The choice is competitive: both X_0 and X_1 are trying to increase their concentration via the intermediary type S_X; whichever has a higher concentration wins. The concentration of the loser effectively drops to zero. So this mechanism clears out the leakage of molecular types that would otherwise occur when bits are set.

To further elucidate the behavior of Reactions 3, consider their kinetic equations:

$$\begin{aligned}\frac{d[S_X]}{dt} &= k[X_0][X_1] - k[S_X][X_0] - k[S_X][X_1]\\ \frac{d[X_0]}{dt} &= -k[X_0][X_1] + 2k[S_X][X_0]\\ \frac{d[X_1]}{dt} &= -k[X_0][X_1] + 2k[S_X][X_1].\end{aligned} \tag{4}$$

Suppose the combined initial concentration of X_0 and X_1 is C and that the initial concentrations of S_X is 0. For a steady-state solution, let $\frac{d[S_X]}{dt} = \frac{d[X_0]}{dt} = \frac{d[X_1]}{dt} = 0$. There are, in fact, three steady-state solutions: $\{X_0 = X_1 = \frac{2C}{5}\}$, $\{X_0 = 0, X_1 = C\}$, and $\{X_0 = C, X_1 = 0\}$. The first is unstable. It is a saddle point: any small perturbation that makes the concentrations of X_0 and X_1 unequal leads to one of the other two solutions. These solutions are both stable. We can map Reactions 3 to DNA strand displacements. Kinetics of the DNA system are similar. This bistability forms the basis of our representation of a bit.

IV. Implementing Logic Gates

Given this robust representation of binary bits, we demonstrate how to implement logic gates with molecular reactions. We only consider two-input gates; gates with more than two inputs can be easily implemented by cascading two-input gates.

Suppose the inputs of a gate are X and Y, and the output is Z. These signals are represented by the concentrations of X_0/X_1, Y_0/Y_1, and Z_0/Z_1, respectively. Each one of X, Y, and Z is regulated by its own version of the bit operation reactions:

$$\begin{aligned} X_0 + X_1 &\longrightarrow S_X\\ S_X + X_0 &\longrightarrow 3X_0\\ S_X + X_1 &\longrightarrow 3X_1\\ Y_0 + Y_1 &\longrightarrow S_Y\\ S_Y + Y_0 &\longrightarrow 3Y_0\\ S_Y + Y_1 &\longrightarrow 3Y_1\\ Z_0 + Z_1 &\longrightarrow S_Z\\ S_Z + Z_0 &\longrightarrow 3Z_0\\ S_Z + Z_1 &\longrightarrow 3Z_1.\end{aligned} \tag{5}$$

For each of the four entries in the truth table for the gate, if the value of Z is 1, then molecules of Z_0, if any, should be transferred to Z_1. Similarly, if the value of Z is 0, then molecules of Z_1, if any, should be transferred to Z_0.

A. AND Gate and OR Gate

Let us first consider an AND gate. By definition, either $X = 0$ or $Y = 0$ sets Z to 0, which means that when either X_0 or Y_0 is present, Z_0 should be generated and Z_1 should be cleared out. This is implemented by the reactions

$$\begin{aligned} X_0 + Z_1 &\longrightarrow X_0 + Z_0\\ Y_0 + Z_1 &\longrightarrow Y_0 + Z_0.\end{aligned} \tag{6}$$

Here, X_0 and Y_0 transfer Z_1 to Z_0 but keep their own concentrations unchanged. Z is set to 0 if it has not already been.

Z should be set to 1 only when both $X = 1$ and $Y = 1$. This is implemented by the reactions

$$\begin{aligned} X_1 + Y_1 &\longrightarrow X_1 + Y_1 + Z_1'\\ 2Z_1' &\longrightarrow \varnothing\\ Z_1' + Z_0 &\longrightarrow Z_1.\end{aligned} \tag{7}$$

In the first reaction, X_1 combines with Y_1 to generate Z_1', an indicator that Z should be set to 1. The concentrations of X_1 and Y_1 do not change[1]. Z_1' is transferred to an external sink, denoted by $\varnothing$, in the second reaction. (This could be a waste type whose concentration we do not track.) When molecules of both X_1 and Y_1 are present, these reactions maintain the concentration of Z_1' at an equilibrium level. When one of X_1 and Y_1 is not present, Z_1' gets cleared out. In the last reaction, Z_1' transfers Z_0 to Z_1. Taken together, Reactions 5, 6 and 7 implement an AND gate.

To simulate the AND gate, we map Reactions 5, 6 and 7 to DNA strand-displacement reactions and generate their corresponding ODEs. The results are shown in Figure 2A. The initial concentrations were set as follows: $[Z_1] = [Z_1'] = 0$, and $[S_X] = [S_Y] = [S_Z] = 0$. Note that Z_0 can be set to any nonzero value C; we used $C = 10^{-8} mol/L = 10nM$ in this simulation. We sweep the range of initial values of $[X_1]$ and $[Y_1]$ from 0 to C; similarly, we sweep $[X_0]$ and $[Y_0]$ from C to 0. The resulting steady-state values of $[Z_1]$ for each input combination are recorded. The figure demonstrates that the AND gate works perfectly: when both $[X_1] > \frac{C}{2}$ and $[Y_1] > \frac{C}{2}$, $[Z_1] = C$; otherwise $[Z_1] = 0$.

The reactions for the OR gate are similar to those for the AND gate. Either $X = 1$ or $Y = 1$ sets Z to 1. This entails having both X_1 and Y_1 transfer Z_0 to Z_1:

$$\begin{aligned} X_1 + Z_0 &\longrightarrow X_1 + Z_1\\ Y_1 + Z_0 &\longrightarrow Y_1 + Z_1.\end{aligned} \tag{8}$$

When both $X = 0$ and $Y = 0$, molecules of Z_1 are transferred to Z_0:

$$\begin{aligned} X_0 + Y_0 &\longrightarrow X_0 + Y_0 + Z_0'\\ 2Z_0' &\longrightarrow \varnothing\\ Z_0' + Z_1 &\longrightarrow Z_0.\end{aligned} \tag{9}$$

Simulation results for the OR gate are shown in Figure 2B. The simulation method and initial concentrations are the same as those used for the AND gate.

NAND and NOR gates can be implemented by effecting the transfers between Z_0 and Z_1 in the opposite directions of those of the AND and OR gates. We illustrate for the NOR gate only. Together with Reactions 5, the following reactions

[1]This reaction looks against molecular conservation. In DNA implementation, however, Z_1' is generated from external fuels.

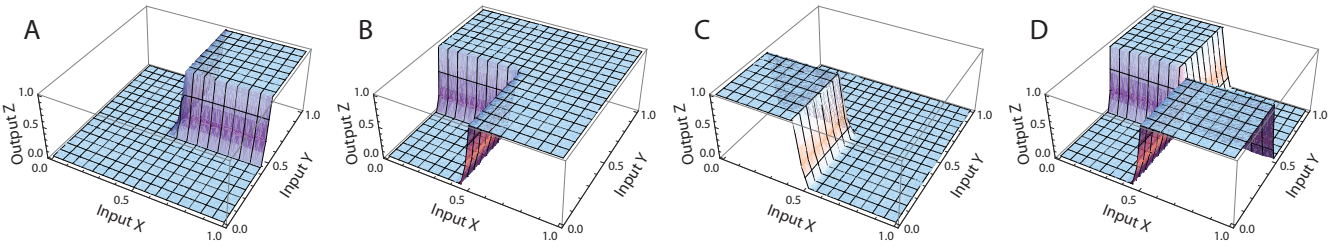

Fig. 2: Simulation results of the kinetic equations for the logic gates. Concentration values are normalized.

implement a NOR gate:

$$\begin{array}{rcl} X_1 + Z_1 & \longrightarrow & X_1 + Z_0 \\ Y_1 + Z_1 & \longrightarrow & Y_1 + Z_0. \\ X_0 + Y_0 & \longrightarrow & X_0 + Y_0 + Z_1' \\ 2Z_1' & \longrightarrow & \varnothing \\ Z_1' + Z_0 & \longrightarrow & Z_1. \end{array} \tag{10}$$

Simulation results for the NOR gate are shown in Figure 2C.

B. XOR gate

We could, of course, implement an exclusive-OR (XOR) gate with say NAND gates or NOR gates. Instead, we present a direct implementation. For an XOR gate, $Z = 1$ when $X \neq Y$. Therefore, molecules of Z_0 are transferred to Z_1 when both X_0 and Y_1 are present, or when both X_1 and Y_0 are present:

$$\begin{array}{rcl} X_0 + Y_1 & \longrightarrow & X_0 + Y_1 + Z_1' \\ X_1 + Y_0 & \longrightarrow & X_1 + Y_0 + Z_1' \\ 2Z_1' & \longrightarrow & \varnothing \\ Z_1' + Z_0 & \longrightarrow & Z_1. \end{array} \tag{11}$$

Similarly, when both X_0 and Y_0 are present, or when both X_1 and Y_1 are present, molecules of Z_1 are transferred to Z_0:

$$\begin{array}{rcl} X_0 + Y_0 & \longrightarrow & X_0 + Y_0 + Z_0' \\ X_1 + Y_1 & \longrightarrow & X_1 + Y_1 + Z_0' \\ 2Z_0' & \longrightarrow & \varnothing \\ Z_0' + Z_1 & \longrightarrow & Z_0. \end{array} \tag{12}$$

Simulation results for the XOR gate are shown in Figure 2D.

C. Kinetic Analysis

Reactions 5 strive to retain the previous value of Z. However, when the inputs of a gate change and molecules of one of Z_0 or Z_1 are transferred to the other, the "force" to keep the previous value set by Reactions 5 is overcome by the "force" changing it.

Consider an AND gate with $X = 0$, $Y = 1$ and $Z = 1$. Initially, $[X_0] = [Y_1] = [Z_1] = C$, and the concentrations of other types are all 0. As long as $[Z_1] > [Z_0]$, Reactions 5 transfer molecules of Z_0 to Z_1 to preserve $Z = 1$. They compete with Reactions 6, which set $Z = 0$. Active reactions related to Z_0 are:

$$\begin{array}{rcl} Z_0 + Z_1 & \longrightarrow & S_Z \\ S_Z + Z_0 & \longrightarrow & 3Z_0 \\ X_0 + Z_1 & \longrightarrow & X_0 + Z_0. \end{array} \tag{13}$$

The rate of change of $[Z_0]$ is

$$\begin{aligned} \frac{d[Z_0]}{dt} &= -k[Z_0][Z_1] + 2k[S_Z][Z_0] + k[X_0][Z_1] \\ &= -k[Z_0][Z_1] + 2k[S_Z][Z_0] + kC[Z_1] \end{aligned} \tag{14}$$

Since $[Z_0] \leq C$, $\frac{d[Z_0]}{dt} \geq 0$. $\frac{d[Z_0]}{dt} = 0$ only when $[Z_0] = C$. This means Z_1 will be continuously transferred to Z_0, until $[Z_0]$ reaches C. The "force" of an input bit changing an output bit overcomes the "force" preserving the previous value. Similar reasoning can be applied to the other cases of the AND gate, and for the other gates.

V. Implementing D Flip-Flop

In this section, we discuss the implementation of the key block for sequential logic, namely a D flip-flop. We start by implementing a D latch and then we implement a D flip-flip using a master-slave configuration of D latches.

A. D Latch

A D latch has two inputs, the latch input D and an enable signal, and one output Q, as shown in Figure 3A. When the enable signal is 1, the latch output is equal to the latch input. When the enable signal is 0, the latch holds the last input value that it saw before the enable signal was still 1. We could, of course, implement such a D latch with cross-coupled NOR gates. Instead, we present a direct implementation based upon our bistable construct for binary values. Indeed, Reactions 3 provide a state-locking mechanism. Based on those reactions, we introduce an enabling signal CLK such that value of the input is transferred to output only when $CLK = 1$. When $CLK = 0$, the state-locking mechanism holds the output value. Similar to other bits, CLK is represented by molecular types CLK_0 and CLK_1.

The D latch is implemented by following reactions:

$$\begin{array}{rcl} CLK_1 + D_0 & \longrightarrow & CLK_1 + D_0 + Q_0' \\ CLK_1 + D_1 & \longrightarrow & CLK_1 + D_1 + Q_1' \\ 2Q_0' & \longrightarrow & \varnothing \\ 2Q_1' & \longrightarrow & \varnothing \\ Q_0' + Q_1 & \longrightarrow & Q_0 \\ Q_1' + Q_0 & \longrightarrow & Q_1. \end{array} \tag{15}$$

With enabling signal CLK_1, the first two reactions generate Q_0' or Q_1', with presence of input signals D_0 or D_1, respectively. The next two reactions ensure that molecules of Q_0' (Q_1') do not accumulate when there are no molecules of D_0 (D_1) present. Finally, the last two reactions set Q to 0 or to 1 if Q_0' is present or if Q_1' is present, respectively.

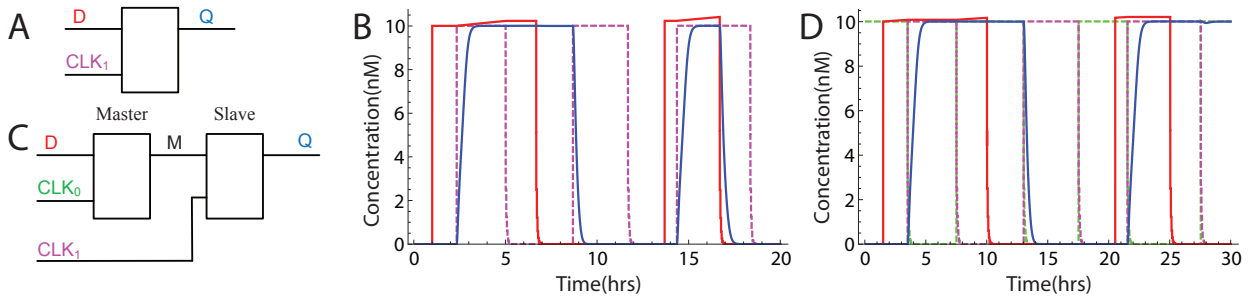

Fig. 3: Block diagrams and transient simulation results of D latch and D flip-flop. Signals are color coded.

Transient simulation results for DNA strand-displacement reactions of the D latch are shown in Figure 3B. The initial conditions for the system are set to $[CLK_0] = [D_0] = [Q_0] = C$, $[CLK_1] = [D_1] = [Q_1] = 0$, $[Q'_0] = [Q'_1] = 0$ and $[S_Q] = 0$. Q follows D only when CLK_1 is present.

B. D Flip-Flop

Unlike a latch, a flip-flop reacts to changes in its enabling signal. If the enabling signal is clock, then the flip-flop only grabs its input on the rising edge of the clock, that is to say when the clock signal changes from 0 to 1. We implement a D flip-flop with a master-slave configuration of D latches, as shown in Figure 3C. In this configuration, the signal D goes through two D latches in series. When $CLK = 0$, the master latch is enabled and the value of D passes through it. Meanwhile, the slave latch retains its previous value. When CLK turns to 1, the master latch is switched off and retains its previous value. At the same time, slave latch is enabled and the value from the master latch passes through. This mechanism is implemented by the following reactions:

$$\begin{aligned}
CLK_0 + D_0 &\longrightarrow CLK_0 + D_0 + M'_0 \\
CLK_0 + D_1 &\longrightarrow CLK_0 + D_1 + M'_1 \\
2M'_0 &\longrightarrow \varnothing \\
2M'_1 &\longrightarrow \varnothing \\
M'_0 + M_1 &\longrightarrow M_0 \\
M'_1 + M_0 &\longrightarrow M_1 \\
CLK_1 + M_0 &\longrightarrow CLK_1 + M_0 + Q'_0 \\
CLK_1 + M_1 &\longrightarrow CLK_1 + M_1 + Q'_1 \\
2Q'_0 &\longrightarrow \varnothing \\
2Q'_1 &\longrightarrow \varnothing \\
Q'_0 + Q_1 &\longrightarrow Q_0 \\
Q'_1 + Q_0 &\longrightarrow Q_1.
\end{aligned} \tag{16}$$

We also include the bistable bit operation reactions for M and Q. In the set of Reactions 16, the first six reactions implement the master latch, which is enabled by CLK_0. The slave latch, enabled by CLK_1, takes M_0 and M_1, the output of the master latch, as its input signals. It is implemented by the last six reactions.

The transient simulation results are shown in Figure 3D. They are obtained by simulating DNA strand-displacement reactions mapped from Reactions 16. Clearly, the output Q follows the value of D only at rising edges of the CLK signal.[2]

VI. Examples

In this section, we demonstrate three full-fledged examples of digital designs implemented with molecular reactions: a digit-serial square-root unit, a binary adder and a sequential linear feedback shift register (LFSR).

A. A Square-Root Unit

We implement a square-root unit [33] with our constructs for logic gates (The design presented here is with a 4-bit output). It consists of 14 controlled add subtract cells (CAS). a_7 is the most significant bit of input; a_0 is the least significant bit. q_3 is the most significant bit of output; q_0 is the least significant bit. There are eight gates in each CAS. Each gate is implemented by the corresponding molecular reactions discussed in previous section. The unit computes square-root in a systematic way and can be easily extended to n-bit output, for values of $n > 4$.

Figure 4 shows the schematic and simulation results of the square-root unit. In Figure 4(c), the input values for the system are $a_7a_6 \cdots a_0 = 10101001$, $a_7a_6 \cdots a_0 = 01100001$, $a_7a_6 \cdots a_0 = 01111111$, respectively. The outputs correctly show the square root of the input values. We see that the less significant bits reach the correct levels more slowly than more significant bits, because the values of the former depend on the values of the latter.

B. A Binary Adder

We implement a four-bit adder with our constructs for logic gates (The design presented here can be easily extended to an n-bit adder, for values of $n > 4$). The block diagram of the adder is shown in Figure 5A. It consists of three full adder (FA) components and one half adder (HA) component. S_3 is the most significant bit of output; S_0 is the least significant bit. Schematics of full adder and half adder are shown in Figure 5B. There are five gates in full adder and two gates in half adder. Each gate is implemented by corresponding molecular reactions discussed in Section IV. Due to space limit, we do not list them here.

Transient solutions of three different input combinations are shown in Figure 5C. They are obtained by solving DNA

[2]A discussion of how to generate proper "clock" signals is beyond the scope of this paper. A variety of molecular oscillators have been proposed in the literature; these could readily be used for this purpose.

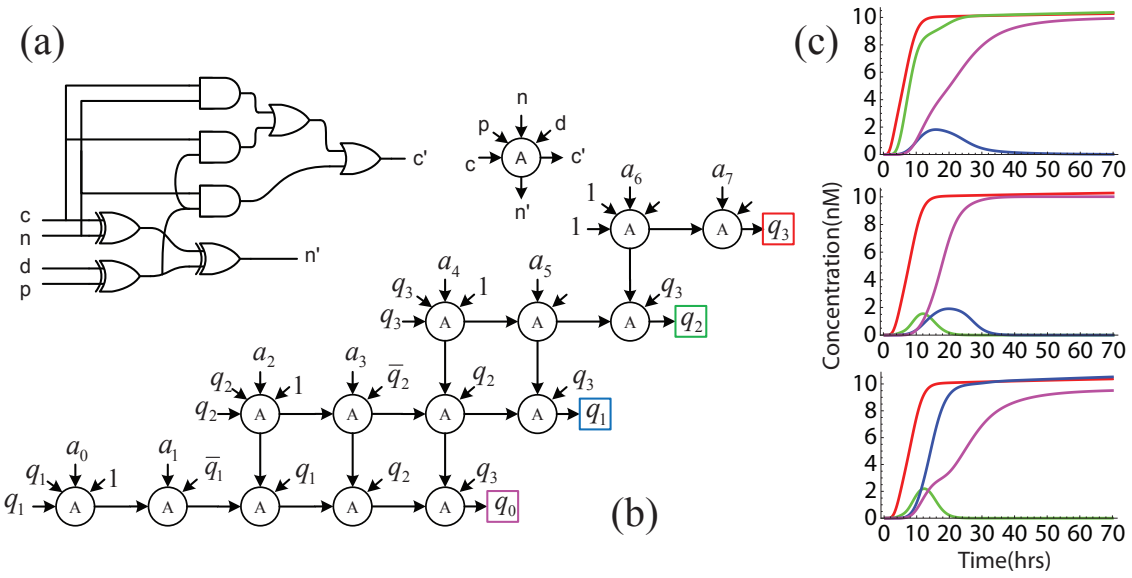

Fig. 4: A square-root unit consisting of 14 controlled add subtract cells (CAS). Maximum strand displacement rate constant $q_{max} = 10^6 M^{-1}s^{-1}$; initial concentration of auxiliary species $C_{max} = 10\mu M$. (a) Schematic of a CAS. (b) Schematic of the unit. (c) Transient behaviors of the unit with inputs set to $a_7a_6\cdots a_0 = 10101001$, $a_7a_6\cdots a_0 = 01100001$, $a_7a_6\cdots a_0 = 01111111$, respectively.

strand-displacement reactions. The input combinations are "0111"+"1110", "0111"+"0101", and "1111"+"0001". We see that the output signals converge to the correct values.

C. A Linear Feedback Shift Register

We also demonstrate the design of a sequential degree-3 LFSR. Its schematic is shown in Figure 5D. It consists of three D flip-flops and one XOR gate.

We synthesize the components of the LFSR according to Reactions 11 and 16. Transient simulation results are shown in Figure 5E. We set initial values to $A = B = C =$"111". All possible states, except "000", are visited in 7 clock cycles. The LFSR works as expected.

VII. Remarks

Although pertaining to biology, the contributions of this paper are not experimental nor empirical; rather they are constructive and conceptual. The premise is that one can design a set of molecular reactions which are further mapped to DNA strand-displacement reactions; such reactions translate into a set of coupled differential equations modeling the rate of change of concentrations according to chemical kinetics; simulating the differential equations provides an accurate characterizing of how the system would behave. The challenge is how to design the set of reactions to implement specific forms of computation.

We are the first to design robust digital logic with molecular reactions. Compared to previous attempts, the bit transitions in our designs are remarkably crisp. Errors do not accumulate across cycles in sequential computation. Significantly, our designs do not depend on specific reaction rates; the computation is accurate for a wide range of rates. This is crucial for mapping the design to DNA substrates.

DNA strand displacement is a well-developed experimental chassis. A detailed discussion of the mapping to this substrate and the corresponding experimental procedures are beyond the scope of this paper. We point the reader to [5]. Our contribution can be positioned as the "front-end" of the design flow; the DNA assembler and experimental chassis described by these authors constitute the "back-end".

References

[1] L. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 11, pp. 1021–1024, 1994.

[2] L. Qian, D. Soloveichik, and E. Winfree, "Efficient turing-universal computation with DNA polymers," in *International Conference on DNA Computing and Molecular Programming*, 2010.

[3] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree, "Enzyme-free nucleic acid logic circuits," in *Science*, vol. 314, 2006, pp. 1585–1588.

[4] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with finite stochastic chemical reaction networks," *Natural Computing*, vol. 7, no. 4, 2008.

[5] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," *Proceedings of the National Academy of Sciences*, vol. 107, no. 12, pp. 5393–5398, 2010.

[6] B. Yurke, A. J. Turberfield, A. P. Mills, Jr, F. C. Simmel, and J. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, pp. 605–608, 2000.

[7] J. C. Anderson, E. J. Clarke, A. P. Arkin, and C. A. Voigt, "Environmentally controlled invasion of cancer cells by engineered bacteria," *Journal of Molecular Biology*, vol. 355, no. 4, pp. 619–627, 2006.

[8] D. Ro, E. Paradise, M. Ouellet, K. Fisher, K. Newman, J. Ndungu, K. Ho, R. Eachus, T. Ham, M. Chang, S. Withers, Y. Shiba, R. Sarpong, , and J. Keasling, "Production of the antimalarial drug precursor artemisinic acid in engineered yeast," *Nature*, vol. 440, pp. 940–943, 2006.

[9] S. Venkataramana, R. M. Dirks, C. T. Ueda, and N. A. Pierce, "Selective cell death mediated by small conditional RNAs," *Proceedings of the National Academy of Sciences*, vol. 107, no. 39, pp. 16 777–16 782, 2010.

[10] D. M. Widmaier, D. Tullman-Ercek, E. A. Mirsky, R. Hill, S. Govindarajan, J. Minshull, and C. A. Voigt, "Engineering the Salmonella type III secretion system to export spider silk monomers," *Molecular Systems Biology*, vol. 5, no. 309, pp. 1–9, 2009.

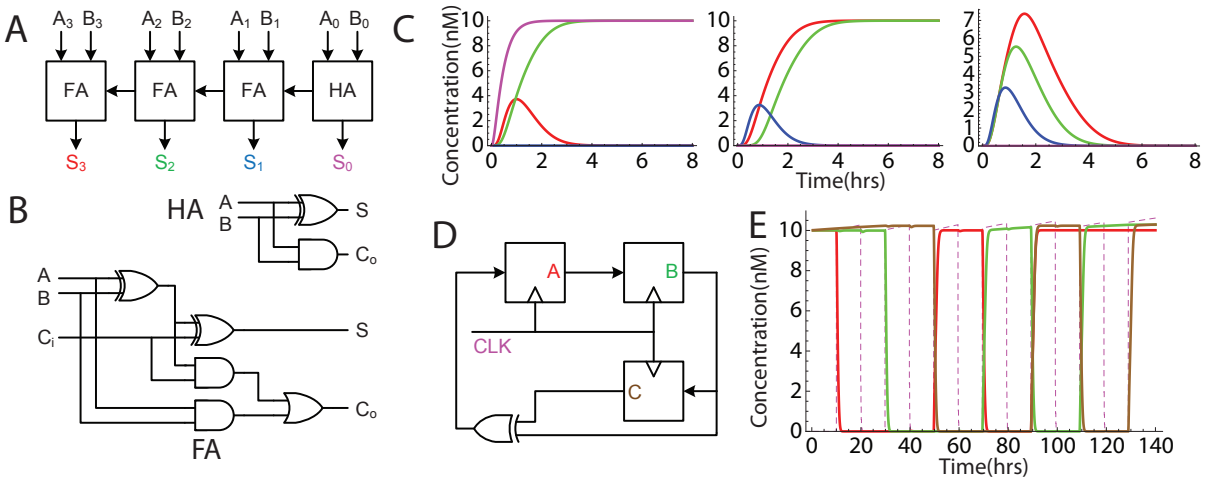

Fig. 5: Schematics and transient simulation results of the adder and LFSR. Maximum strand displacement rate constant $q_{max} = 10^6 M^{-1} s^{-1}$; initial concentration of auxiliary species $C_{max} = 10 \mu M$. Signals are color coded.

[11] J. C. Anderson, C. A. Voigt, and A. P. Arkin, "A genetic AND gate based on translation control," *Molecular Systems Biology*, vol. 3, no. 133, 2007.

[12] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro, "An autonomous molecular computer for logical control of gene expression," *Nature*, vol. 429, no. 6990, pp. 423–429, 2004.

[13] K. Ramalingam, J. R. Tomshine, J. A. Maynard, and Y. N. Kaznessis, "Forward engineering of synthetic bio-logical and gates," *Biochemical Engineering Journal*, vol. 47, no. 1–3, pp. 38–47, 2009.

[14] M. N. Stojanovic, T. E. Mitchell, and D. Stefanovic, "Deoxyribozyme-based logic gates," *Journal of the American Chemical Society*, vol. 124, pp. 3555–3561, 2002.

[15] R. Weiss, G. E. Homsy, and T. F. Knight, "Toward in vivo digital circuits," in *DIMACS Workshop on Evolution as Computation*, 1999, pp. 1–18.

[16] M. N. Win and C. D. Smolke, "A modular and extensible RNA-based gene-regulatory platform for engineering cellular function," *Proceedings of the National Academy of Sciences*, vol. 104, no. 36, p. 14283, 2007.

[17] M. N. Win, J. Liang, and C. D. Smolke, "Frameworks for programming biological function through RNA parts and devices," *Chemistry & Biology*, vol. 16, pp. 298–310, 2009.

[18] Y. Yokobayashi, R. Weiss, and F. H. Arnold, "Directed evolution of a genetic circuit," *Proceedings of the National Academy of Sciences*, vol. 99, no. 26, pp. 16 587–16 591, 2002.

[19] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[20] A. Arkin and J. Ross, "Computational functions in biochemical reaction networks," *Biophysical Journal*, vol. 67, no. 2, pp. 560 – 578, 1994.

[21] R. Weiss, "Cellular computation and communications using engineering genetic regulatory networks," Ph.D. dissertation, MIT, 2003.

[22] M. N. Win and C. D. Smolke, "higher-order cellular information processing with synthetic RNA devices," *Science*, vol. 322, no. 5900, pp. 456–460, 2008.

[23] P. Senum and M. D. Riedel, "Rate-independent constructs for chemical computation," *PLoS ONE*, vol. 6, no. 6, 2011.

[24] B. Fett, J. Bruck, and M. D. Riedel, "Synthesizing stochasticity in biochemical systems," in *Design Automation Conference*, 2007, pp. 640–645.

[25] B. Fett and M. D. Riedel, "Module locking in biochemical systhesis," in *International Conference on Computer-Aided Design*, 2008, pp. 758–764.

[26] H. Jiang, S. Salehi, M. Riedel, and K. Parhi, "Discrete-time signal processing with DNA," *ACS Synthetic Biology*, vol. 2, no. 5, pp. 245–254, 2013.

[27] A. Goldbeter and D. E. Koshland, "An amplified sensitivity arising from covalent modification in biological systems," *Proceedings of the National Academy of Sciences*, vol. 78, no. 11, pp. 6840–6844, 1961.

[28] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, pp. 122–127, 1969.

[29] L. Nagel and D. Pederson, "Simulation program with integrated circuit emphasis," in *Midwest Symposium on Circuit Theory*, 1973.

[30] P. Érdi and J. Tóth, *Mathematical Models of Chemical Reactions: Theory and Applications of Deterministic and Stochastic Models*. Manchester University Press, 1989.

[31] F. Horn and R. Jackson, "General mass action kinetics," *Archive for Rational Mechanics and Analysis*, vol. 47, pp. 81–116, 1972.

[32] L. Qian and E. Winfree, "A simple dna gate motif for synthesizing large-scale circuits," *Journal of The Royal Society Interface*, 2011.

[33] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 4, pp. 358–375, 1991.