Digital Logic with Molecular Reactions

Hua Jiang, Aleksandra P. Kharam, Marc D. Riedel and Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota
200 Union St. S.E., Minneapolis, MN 55455
{hua, veden002, 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 bistable bit representation, a constituent set of logical components are implemented. These include combinational components – AND, OR, and XOR – as well as sequential components – D latches and D flip-flops. Using these components, two full-fledged design examples are given: a binary counter and a linear feedback shift register. All the constructs consist of sets of coupled chemical reactions with only coarsely specified rate categories ("fast" and "slow"). Given such categories, the computation is robust regardless of the specific reaction rates. The designs are validated through simulations of the chemical kinetics. The simulations show that the constructs produce nearly perfect digital signal values.**

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 chemical types into output concentrations of chemical types [1], [2], [3], [4], [5], [6].

Yet 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 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].

As one might expect, the success of these endeavors has mostly been attributable to the experimental expertise of the practitioners in specific domains of biology. However, the field has reached a stage where it is beneficial to bring in expertise from computer engineering and from circuit design.

Indeed, 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 and incrementing/decrementing [23]; programming constructs such as "for" and "while" loops [24]; arithmetic operations such as multiplication, exponentiation and logarithms [23], [24]; and signal processing operations such as filtering [25], [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, and XOR – as well as for sequential components – D latches and D flip-flops. We illustrate the use of these components with two full-fledged design examples: a binary counter and a linear feedback shift register (LFSR) [28].

We validate the designs through simulations of the chemical kinetics, based on ordinary differential equations – the equivalent of SPICE simulations for electronic systems [29]. The simulations results show that our gates, our counter, 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 reactions. 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

This work is supported by an NSF EAGER Grant, #CCF0946601.

of D latches and D flip-flops. In Section VI, we present the examples of a binary counter and an LFSR. Finally, in Section VII, we discuss potential applications of our design methodology.

II. Computational Model

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

$$A + B \xrightarrow{\text{k}} 2C \tag{1}$$

specifies that one molecule of A combines with one molecule of B to produce two molecules of C. This reaction fires at a rate proportional to a kinetic constant k. 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 kinetic constants. Accordingly, for the reaction above, the rate of change in the concentrations of A, B and C is

$$-\frac{d[A]}{dt} = -\frac{d[B]}{dt} = \frac{1}{2}\frac{d[C]}{dt} = k[A][B], \tag{2}$$

(here $[\cdot]$ denotes concentration). Concentration is defined to be the quantity of molecules per unit volume. Without loss of generality, we use unitless numbers for concentrations and kinetic constants. These can be readily mapped to values with units in a physical implementation.

Given a coupled set of chemical reactions, one obtains a set of ordinary differential equations. These can readily be solved numerically. Throughout this paper, we present simulation results obtained with MATLAB. For instance, consider the following set of three reactions (known as *Lotka-Volterra* system [32]):

$$\begin{array}{rcl} X & \xrightarrow{\alpha} & 2X \\ X + Y & \xrightarrow{\beta} & 2Y \\ Y & \xrightarrow{\gamma} & \varnothing. \end{array} \tag{3}$$

The kinetic equation for this system is

$$\begin{aligned} \frac{d[X]}{dt} &= \alpha X - \beta XY \\ \frac{d[Y]}{dt} &= \beta XY - \gamma Y. \end{aligned} \tag{4}$$

If initial concentrations of types X and Y are given, their concentrations as a function of time t can be obtained by solving this set of ordinary differential equations. Figure 1 shows how the concentrations of X and Y oscillate over time when $\alpha = 1$, $\beta = 1$, $\gamma = 1$ and the initial concentrations are set to $[X] = 10$ and $[Y] = 10$.

In our methodology, we use only coarse rate categories (“fast” and “slow”) for the kinetic constants. Given such categories, the computation is robust regardless of the specific reaction rates. In particular, it does not matter how fast any “fast” reaction is – only that “fast” reactions are fast relative to “slow” reactions. This is crucial for mapping a set of reactions onto specific chemical substrates. (Throughout this paper, unless denoted as “fast” over the arrow, all reactions are assumed to be in the “slow” category.) All of our designs consist of either unimolecular or bimolecular reactions, i.e., reactions with one or two reactants, respectively. This too is important for mapping to chemical substrates, since the kinetics of reactions with more than two reactants are complex and often physically unrealistic.

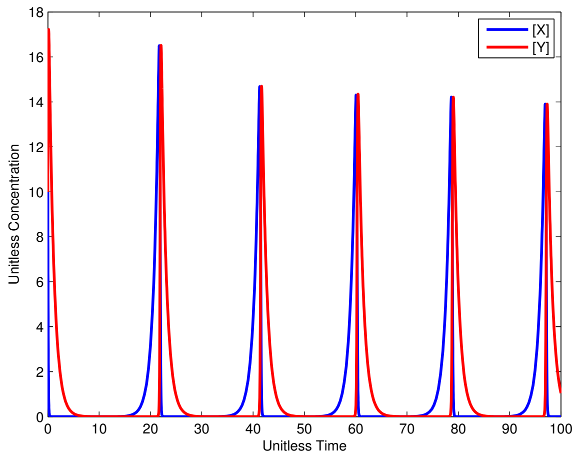


Fig. 1: ODE simulation results of Lotka-Volterra System.

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 [33], [34]. 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:

$$\begin{array}{rcl} X_0 + X_1 & \longrightarrow & S_X \\ S_X + X_0 & \longrightarrow & 3X_0 \\ S_X + X_1 & \longrightarrow & 3X_1. \end{array} \tag{5}$$

In Reactions 5, 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 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 5, consider their kinetic equations:

$$
\begin{aligned}
\frac{d[S_X]}{dt} &= k[X_0][X_1] - k[S_X][X_0] - k[S_X][X_1] \\
\frac{d[X_0]}{dt} &= -k[X_0][X_1] + 2k[S_X][X_0] \\
\frac{d[X_1]}{dt} &= -k[X_0][X_1] + 2k[S_X][X_1].
\end{aligned} \tag{6}
$$

Here, k is the kinetic constant for these reactions (all are in the "slow" category). Suppose the combined initial concentration of X_0 and X_1 is C. Suppose that the initial concentrations of S_X is 0. For a steady-state solution, let $\frac{d[S_X]}{dt} = \frac{d[S_Y]}{dt} = \frac{d[S_Z]}{dt} = 0$. There are, in fact, three steady-state solutions: $\{X_0 = X_1 = \frac{C}{2}\}$, $\{X_0 = 0, X_1 = C\}$, 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; 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:

$$
\begin{array}{rcl}
X_0 + X_1 & \longrightarrow & S_X \\
S_X + X_0 & \longrightarrow & 3X_0 \\
S_X + X_1 & \longrightarrow & 3X_1 \\
Y_0 + Y_1 & \longrightarrow & S_Y \\
S_Y + Y_0 & \longrightarrow & 3Y_0 \\
S_Y + Y_1 & \longrightarrow & 3Y_1 \\
Z_0 + Z_1 & \longrightarrow & S_Z \\
S_Z + Z_0 & \longrightarrow & 3Z_0 \\
S_Z + Z_1 & \longrightarrow & 3Z_1.
\end{array} \tag{7}
$$

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

$$
\begin{array}{rcl}
X_0 + Z_1 & \longrightarrow & X_0 + Z_0 \\
Y_0 + Z_1 & \longrightarrow & Y_0 + Z_0.
\end{array} \tag{8}
$$

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

$$
\begin{array}{rcl}
X_1 + Y_1 & \longrightarrow & X_1 + Y_1 + Z_1' \\
Z_1' & \longrightarrow & \varnothing \\
Z_1' + Z_0 & \longrightarrow & Z_1.
\end{array} \tag{9}
$$

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. 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 7, 8 and 9 implement an AND gate.

Simulation results for the AND gate, obtained by solving ODEs corresponding to the reactions with MATLAB, are shown in Figure 2. 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 $C = 10$ 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 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 $Z_1 = 0$.

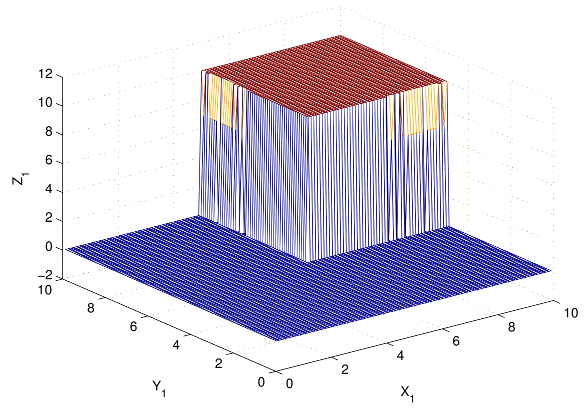


Fig. 2: Simulation results of the kinetic equations for the AND gate.

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:

$$
\begin{array}{rcl}
X_1 + Z_0 & \longrightarrow & X_1 + Z_1 \\
Y_1 + Z_0 & \longrightarrow & Y_1 + Z_1.
\end{array} \tag{10}
$$

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

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

Simulation results for the OR gate, obtained by solving ODEs corresponding to the reactions with MATLAB, are shown in Figure 3. The initial concentrations are the same as those used for the AND gate.

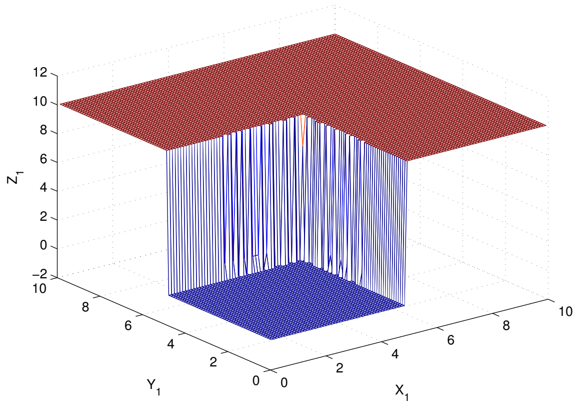


Fig. 3: Simulation results of the kinetic equations for the OR gate.

NAND gate and NOR gate 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 NAND gate only. Together with Reactions 7, the following reactions implement the NAND gate:

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

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 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:

$$\begin{array}{rcl} X_0 + Y_1 & \longrightarrow & X_0 + Y_1 + Z_1' \\ X_1 + Y_0 & \longrightarrow & X_1 + Y_0 + Z_1' \\ Z_1' & \longrightarrow & \varnothing \\ Z_1' + Z_0 & \longrightarrow & Z_1. \end{array} \tag{13}$$

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:

$$\begin{array}{rcl} X_0 + Y_0 & \longrightarrow & X_0 + Y_0 + Z_0' \\ X_1 + Y_1 & \longrightarrow & X_1 + Y_1 + Z_0' \\ Z_0' & \longrightarrow & \varnothing \\ Z_0' + Z_1 & \longrightarrow & Z_0. \end{array} \tag{14}$$

Simulation results for the XOR gate, obtained by solving ODEs corresponding to the reactions with MATLAB, are shown in Figure 4. The initial concentrations are the same as those used for the AND and OR gates.

C. Kinetic Analysis

Reactions 7 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 7 is overcome by the "force" changing it.

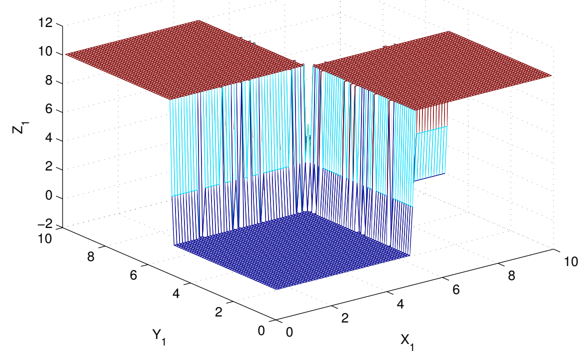


Fig. 4: Simulation results of the kinetic equations for the XOR gate.

Consider an AND gate with $X = 0$, $Y = 1$ and $Z = 1$. Initially, $[X_0] = C$, $[Y_1] = C$ and $[Z_1] = C$, and the concentrations of other types are all 0. As long as $[Z_1] > [Z_0]$, Reactions 7 transfer molecules of Z_0 to Z_1 to preserve $Z = 1$. They compete with Reactions 8, which set $Z = 0$. The reactions related to Z_0 are:

$$\begin{array}{rcl} Z_0 + Z_1 & \xrightarrow{\text{k}} & S_Z \\ S_Z + Z_0 & \xrightarrow{\text{k}} & 3Z_0 \\ X_0 + Z_1 & \xrightarrow{\text{k}} & X_0 + Z_0. \end{array} \tag{15}$$

Again, k is the kinetic constant for slow reactions. The rate of change of Z_0 is

$$\begin{aligned} \frac{d[Z_0]}{dt} &= -k[Z_0][Z_1] + 2k[S_Z][Z_0] + k[X_0][Z_1] \\ &= -k[Z_0][Z_1] + 2k[S_Z][Z_0] + kC[Z_1] \end{aligned} \tag{16}$$

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 and an enable signal, and one output. 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 5 provide a state-locking mechanism. Based on those reactions, we introduce an enabling signal E such that value of the input is transferred to output only when $E = 1$. When $E = 0$, the state-locking mechanism holds the output value.

We first discuss the enabling signal E. Similar to other bits, E is represented by molecular types E_0 and E_1. The latch operation requires that only one of E_0 and E_1 is present at a time; transitions between these values must be non-overlapping. However, as external inputs, E_0 and E_1 can be overlapped, particularly during the transition from $E = 0$ to $E = 1$ or vice versa. We therefore use two other molecular types, E'_0 and E'_1, as control signals. They are regulated by following reactions:

$$\begin{array}{rcl} E_0 & \longrightarrow & E_0 + E'_0 \\ E_1 & \longrightarrow & E_1 + E'_1 \\ E'_0 & \longrightarrow & \varnothing \\ E'_1 & \longrightarrow & \varnothing \\ E'_0 + E'_1 & \xrightarrow{\text{fast}} & \varnothing \end{array} \tag{17}$$

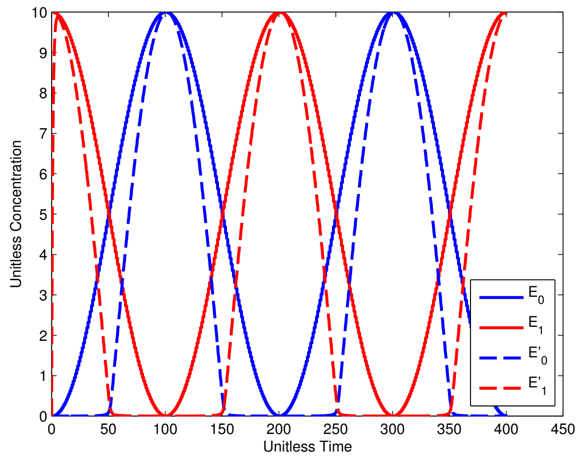
In first two reactions, E_0 and E_1 continuously generate E'_0 and E'_1, respectively. Molecules of E'_0 and E'_1 go to an external sink in the next two reactions, which ensures they exist only when E_0 or E_1 is present, respectively. (The external sink can be a waste type whose concentration we do not track.) The last reaction quickly cancels out equal concentrations of E'_0 and E'_1. Note that this reaction is in the "fast" rate category; this ensures that essentially molecules of only one of E'_0 and E'_1 exist at any instant in time.

The D latch is implemented by following reactions:

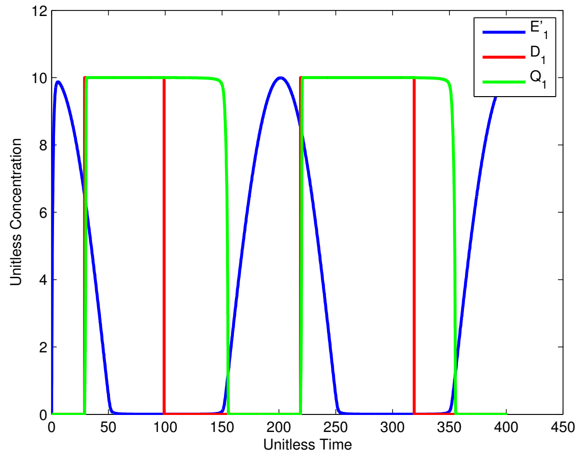
$$\begin{array}{rcl} E'_1 + D_0 & \longrightarrow & E'_1 + D_0 + Q'_0 \\ E'_1 + D_1 & \longrightarrow & E'_1 + D_1 + Q'_1 \\ Q'_0 & \longrightarrow & \varnothing \\ Q'_1 & \longrightarrow & \varnothing \\ Q'_0 + Q_1 & \longrightarrow & Q_0 \\ Q'_1 + Q_0 & \longrightarrow & Q_1. \end{array} \tag{18}$$

With enabling signal E'_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) in presence. Finally, the last two reactions set Q to 0 or 1, in accordance with presence of Q'_0 or Q'_1.

Simulation results for the D latch are shown in Figure 5. The concentrations of Q_0 and Q_1 are obtained from ODE simulations of kinetic equations corresponding to Reactions 17, 18, together with the bistable Reactions 5 for Q. The initial conditions for the system are set to $E_0 = D_0 = Q_0 = C$, $E_1 = D_1 = Q_1 = 0$, $E'_0 = E'_1 = Q'_0 = Q'_1 = 0$ and $S_Q = 0$. To test the effectiveness of Reactions 17, E_0 and E_1 are set to be overlapping sinusoidal signals. The "fast" kinetic constant is set to be 100 times higher than the "slow" constant. Figure 5a shows that E'_0 and E'_1 are non-overlapping version of E_0 and E_1. Figure 5b illustrates the latching behavior. Q follows D only when E'_1 is present.



(a) The enabling signals.



(b) Performance of the latch.

Fig. 5: Simulation results of the kinetic equations for the D latch.

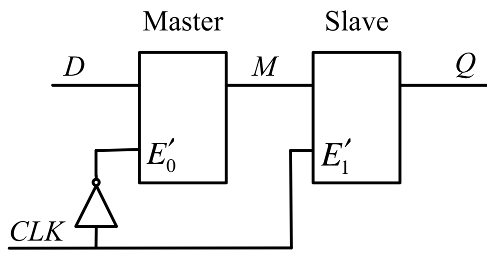


Fig. 6: Master-slave structure of D a flip-flop.

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 6. 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:

$$
\begin{array}{rcl}
E_0' + D_0 & \longrightarrow & E_0' + D_0 + M_0' \\
E_0' + D_1 & \longrightarrow & E_0' + D_1 + M_1' \\
M_0' & \longrightarrow & \varnothing \\
M_1' & \longrightarrow & \varnothing \\
M_0' + M_1 & \longrightarrow & M_0 \\
M_1' + M_0 & \longrightarrow & M_1 \\
E_1' + M_0 & \longrightarrow & E_1' + M_0 + Q_0' \\
E_1' + M_1 & \longrightarrow & E_1' + M_1 + Q_1' \\
Q_0' & \longrightarrow & \varnothing \\
Q_1' & \longrightarrow & \varnothing \\
Q_0' + Q_1 & \longrightarrow & Q_0 \\
Q_1' + Q_0 & \longrightarrow & Q_1.
\end{array}
\tag{19}
$$

The following reactions implement the enabling signal:

$$
\begin{array}{rcl}
CLK_0 & \longrightarrow & CLK_0 + E_0' \\
CLK_1 & \longrightarrow & CLK_1 + E_1' \\
E_0' & \longrightarrow & \varnothing \\
E_1' & \longrightarrow & \varnothing \\
E_0' + E_1' & \xrightarrow{\text{fast}} & \varnothing.
\end{array}
\tag{20}
$$

We also include the bistable bit operation reactions for M and Q. In the set of Reactions 19, the first six reactions implement the master latch, which is enabled by E_0'. The slave latch, enabled by E_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.

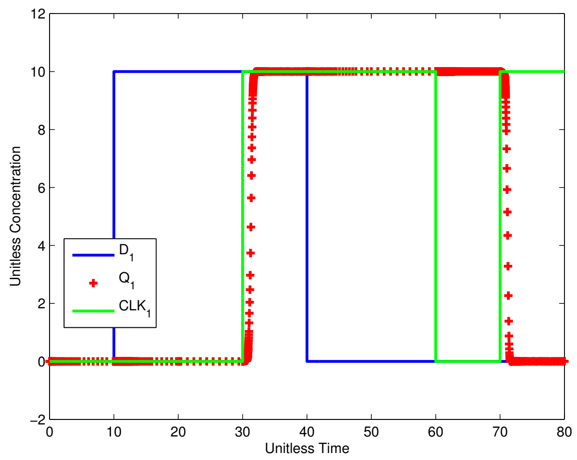


Fig. 7: Simulation results of the kinetic equations for the D flip-flop.

The simulation results are shown in Figure 7. We randomly generate the CLK signal and the input D. Clearly, the output Q follows the value of D only at rising edges of the CLK signal.[1]

[1]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. We point the reader to prior work [35].

VI. Examples

In this section, we demonstrate two full-fledged examples of digital designs implemented with molecular reactions: a three-bit counter and a linear feedback shift register (LFSR).

A. A Three-Bit Counter

We implement the three-bit counter with our constructs for logic gates and for D flip-flops (The design presented here can be easily extended to an n-bit counter, for values of $n > 3$). The schematic of the counter is shown in Figure 8. Molecular

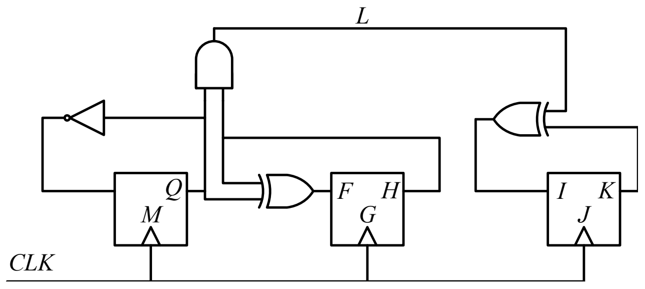


Fig. 8: Schematic of the counter.

types representing signals are labeled in the figure. Here, Q represents the least significant bit; H represents the second bit; K represents the most significant bit. F and I are the input signal types for the last two flip-flops. The first flip-flop does not required a designated input type, as we will discuss later. M, G and J represent the internal signal types of each flip-flop. Finally, L is the output type of the AND gate.

The reactions for the three flip-flops are:

$$
\begin{array}{rcl}
E_0' + Q_0 & \longrightarrow & E_0' + D_0 + M_1' \\
E_0' + Q_1 & \longrightarrow & E_0' + D_1 + M_0' \\
M_0' & \longrightarrow & \varnothing \\
M_1' & \longrightarrow & \varnothing \\
M_0' + M_1 & \longrightarrow & M_0 \\
M_1' + M_0 & \longrightarrow & M_1. \\
E_1' + M_0 & \longrightarrow & E_1' + M_0 + Q_0' \\
E_1' + M_1 & \longrightarrow & E_1' + M_1 + Q_1' \\
Q_0' & \longrightarrow & \varnothing \\
Q_1' & \longrightarrow & \varnothing \\
Q_0' + Q_1 & \longrightarrow & Q_0 \\
Q_1' + Q_0 & \longrightarrow & Q_1
\end{array}
\tag{21}
$$

$$
\begin{array}{rcl}
E_0' + F_0 & \longrightarrow & E_0' + F_0 + G_0' \\
E_0' + F_1 & \longrightarrow & E_0' + F_1 + G_1' \\
G_0' & \longrightarrow & \varnothing \\
G_1' & \longrightarrow & \varnothing \\
G_0' + G_1 & \longrightarrow & G_0 \\
G_1' + G_0 & \longrightarrow & G_1. \\
E_1' + G_0 & \longrightarrow & E_1' + G_0 + H_0' \\
E_1' + G_1 & \longrightarrow & E_1' + G_1 + H_1' \\
H_0' & \longrightarrow & \varnothing \\
H_1' & \longrightarrow & \varnothing \\
H_0' + H_1 & \longrightarrow & H_0 \\
H_1' + H_0 & \longrightarrow & H_1
\end{array}
\tag{22}
$$

and

$$\begin{array}{rcl} E_0' + I_0 & \longrightarrow & E_0' + I_0 + J_0' \\ E_0' + I_1 & \longrightarrow & E_0' + I_1 + J_1' \\ J_0' & \longrightarrow & \varnothing \\ J_1' & \longrightarrow & \varnothing \\ J_0' + J_1 & \longrightarrow & J_0 \\ J_1' + J_0 & \longrightarrow & J_1. \\ E_1' + J_0 & \longrightarrow & E_1' + J_0 + K_0' \\ E_1' + J_1 & \longrightarrow & E_1' + J_1 + K_1' \\ K_0' & \longrightarrow & \varnothing \\ K_1' & \longrightarrow & \varnothing \\ K_0' + K_1 & \longrightarrow & K_0 \\ K_1' + K_0 & \longrightarrow & K_1. \end{array} \tag{23}$$

Reactions 22 and 23 implement standard D flip-flops transferring F to H and I to K. In Reactions 21, the output Q is directly fed back to the input; it sets the internal signal M to the opposite value when E_0' is present. This implicitly implements an NOT gate and it eliminates the requirement of a designated input type.

The AND gate and two XOR gates are implemented by the following reactions:

$$\begin{array}{rcl} H_0 + L_1 & \longrightarrow & H_0 + L_0 \\ Q_0 + L_1 & \longrightarrow & Q_0 + L_0 \\ H_1 + Q_1 & \longrightarrow & H_1 + Q_1 + L_1' \\ L_1' & \longrightarrow & \varnothing \\ L_1' + L_0 & \longrightarrow & L_1 \end{array} \tag{24}$$

and

$$\begin{array}{rcl} H_0 + Q_1 & \longrightarrow & H_0 + Q_1 + F_1' \\ H_1 + Q_0 & \longrightarrow & H_1 + Q_0 + F_1' \\ H_0 + Q_0 & \longrightarrow & H_0 + Q_0 + F_0' \\ H_1 + Q_1 & \longrightarrow & H_1 + Q_1 + F_0' \\ F_0' & \longrightarrow & \varnothing \\ F_1' & \longrightarrow & \varnothing \\ F_1' + F_0 & \longrightarrow & F_1 \\ F_0' + F_1 & \longrightarrow & F_0 \\ K_0 + L_1 & \longrightarrow & K_0 + L_1 + I_1' \\ K_1 + L_0 & \longrightarrow & K_1 + L_0 + I_1' \\ K_0 + L_0 & \longrightarrow & K_0 + L_0 + I_0' \\ K_1 + L_1 & \longrightarrow & K_1 + L_1 + I_0' \\ I_0' & \longrightarrow & \varnothing \\ I_1' & \longrightarrow & \varnothing \\ I_1' + I_0 & \longrightarrow & I_1 \\ I_0' + I_1 & \longrightarrow & I_0. \end{array} \tag{25}$$

Reactions 20–25, together with the bistable bit operations, implement the three-bit counter.

Simulation results, obtained by solving ODEs corresponding to the reactions with MATLAB, are shown in Figure 9. We see that the counter counts from “000” to “111” in eight cycles.

B. A Linear Feedback Shift Register

We demonstrate the design of a degree-4 LFSR. The schematic of such an LFSR is shown in Figure 10. Its feedback polynomial is $x^4 + x^3 + 1$. The four D flip-flops

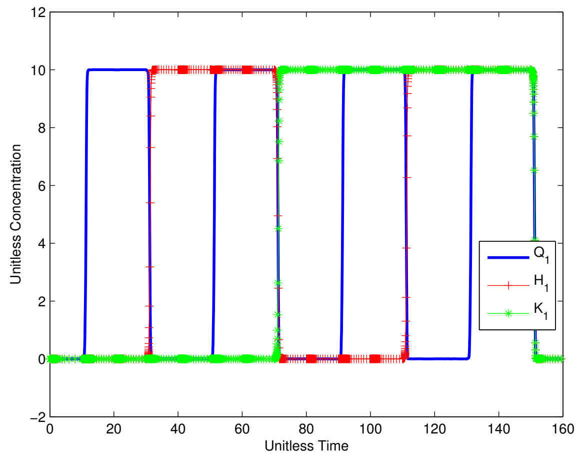


Fig. 9: Simulation results of the kinetic equations for the 3-bit counter.

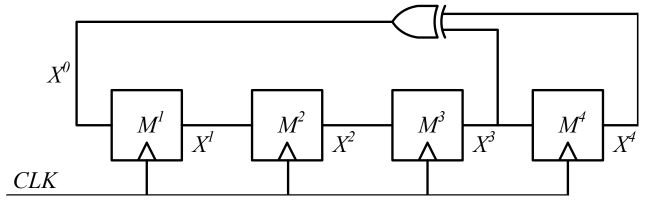


Fig. 10: Schematic of the LFSR.

are implemented by reactions:

$$\begin{array}{rcl} E_0' + X_0^{n-1} & \longrightarrow & E_0' + X_0^{n-1} + M_0^{n\prime} \\ E_0' + X_1^{n-1} & \longrightarrow & E_0' + X_1^{n-1} + M_1^{n\prime} \\ M_0^{n\prime} & \longrightarrow & \varnothing \\ M_1^{n\prime} & \longrightarrow & \varnothing \\ M_0^{n\prime} + M_1^n & \longrightarrow & M_0^n \\ M_1^{n\prime} + M_0^n & \longrightarrow & M_1^n. \\ E_1' + M_0^n & \longrightarrow & E_1' + M_0^n + X_0^{n\prime} \\ E_1' + M_1^n & \longrightarrow & E_1' + M_1^n + X_1^{n\prime} \\ X_0^{n\prime} & \longrightarrow & \varnothing \\ X_1^{n\prime} & \longrightarrow & \varnothing \\ X_0^{n\prime} + X_1^n & \longrightarrow & X_0^n \\ X_1^{n\prime} + X_0^n & \longrightarrow & X_1^n \end{array} \tag{26}$$

where $n = 1, 2, 3, 4$. The XOR gate is implemented by the following reactions:

$$\begin{array}{rcl} X_0^3 + X_1^4 & \longrightarrow & X_0^3 + X_1^4 + X_1^{0\prime} \\ X_1^3 + X_0^4 & \longrightarrow & X_1^3 + X_0^4 + X_1^{0\prime} \\ X_0^3 + X_0^4 & \longrightarrow & X_0^3 + X_0^4 + X_0^{0\prime} \\ X_1^3 + X_1^4 & \longrightarrow & X_1^3 + X_1^4 + X_0^{0\prime} \\ X_1^{0\prime} & \longrightarrow & \varnothing \\ X_0^{0\prime} & \longrightarrow & \varnothing \\ X_1^{0\prime} + X_0^0 & \longrightarrow & X_1^0 \\ X_0^{0\prime} + X_1^0 & \longrightarrow & X_0^0. \end{array} \tag{27}$$

Simulation results, obtained by solving ODEs corresponding to the reactions with MATLAB, are shown in Figure 11. We set initial values to $x_1x_2x_3x_4$ =“1111”. All possible states, except “0000”, are visited in 15 clock cycles.

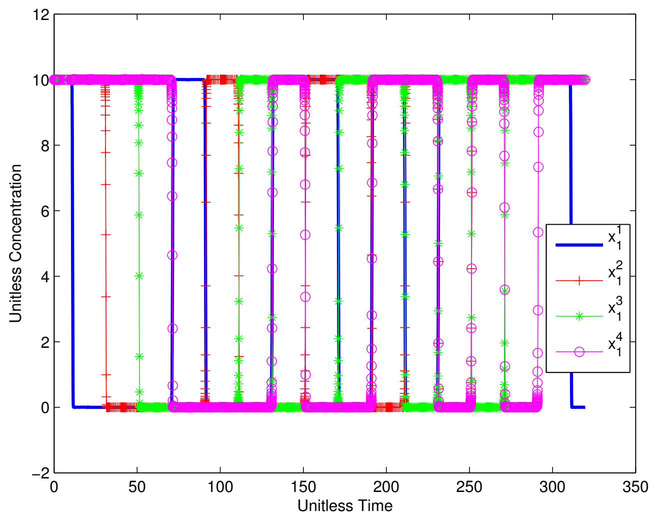


Fig. 11: Simulation results of the kinetic equations for the LFSR.

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; 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 chemical reactions 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, including those describing binary counters [33], [36], the bit transitions in our designs are remarkably crisp. Errors do not accumulate across cycles in sequential computation. As a result, our constructs such as the counter and the LFSR can run indefinitely. Significantly, our constructs 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 specific chemical substrates.

We are exploring the mechanism of DNA-strand displacement as an experimental chassis [6]. The kinetics of arbitrary chemical reactions can be *emulated* with such strand displacements [5]. Reaction rates are controlled by designing sequences with different binding strengths. The binding strengths are controlled by the length and sequence composition of "toehold" sequences of DNA. With the right choice of toehold sequences, reaction rates differing by as much as 10^6 can be achieved. 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*, 2010 (in press).

[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.

[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," vol. 124, no. 14, 2002, pp. 3555–3561.

[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 biochemical computational modules," in *Proceedings of the Pacific Symposium on Biocomputing*, 2011.

[24] A. Shea, B. Fett, M. D. Riedel, and K. Parhi, "Writing and compiling code into biochemistry," in *Proceedings of the Pacific Symposium on Biocomputing*, 2010, pp. 456–464.

[25] H. Jiang, A. P. Kharam, M. D. Riedel, and K. K. Parhi, "A synthesis flow for digital signal processing with biomolecular reactions," in *IEEE International Conference on Computer-Aided Design*, 2010, pp. 417–424.

[26] M. Samoilov, A. Arkin, and J. Ross, "Signal processing by simple chemical systems," *Journal of Physical Chemistry A*, vol. 106, no. 43, pp. 10 205–10 221, 2002.

[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] I. R. Epstein and J. A. Pojman, *An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos*. Oxford Univ Press, 1998.

[33] A. Kharam, H. Jiang, M. D. Riedel, and K. Parhi, "Binary counting with chemical reactions," in *Pacific Symposium on Biocomputing*, 2011.

[34] L. Qian and E. Winfree, "A simple DNA gate motif for synthesizing large-scale circuits," *Journal of the Royal Society Interface*, February 2011.

[35] H. Jiang, M. D. Riedel, and K. K. Parhi, "Synchronous sequential computation with molecular reactions," in *Design Automation Conference*, 2011.

[36] Withheld.