Molecular Sensing and Computing Systems

Sayed Ahmad Salehi, Hua Jiang, Marc D. Riedel, *Senior Member, IEEE*, Keshab K. Parhi, *Fellow, IEEE*

Department of Electrical and Computer Engineering
University of Minnesota
200 Union St. S.E., Minneapolis, MN 55455
{saleh022, mriedel, parhi}@umn.edu

(Invited Paper)

***Abstract*—Advances in the field of synthetic biology have been key to demonstration of molecular computing systems in general and DNA in particular. This paper presents an overview of how continuous-time, discrete-time and digital signal processing systems can be implemented using molecular reactions and DNA. In this paper, discrete-time systems refer to sampled signals with continuous signal amplitude. Signals that are sampled in discrete time steps with digital amplitude are referred as digital signals. Delay elements in sampled signals are implemented using molecular reactions in the form of molecular transfer reactions. Completion of all phases of transfer reactions once corresponds to a computation cycle. These molecular systems can be implemented in a fully-synchronous, globally-synchronous locally-asynchronous or fully-asynchronous manner. The paper also presents molecular sensing systems where molecular reactions are used to implement analog-to-digital converters (ADCs) and digita-to-analog converters (DACs). Molecular implementations of digital logic systems are presented. A complete example of addition of two molecules using digital implementation is described where the concentrations of two molecules are converted to digital by two 3-bit ADCs, and the 4-bit output of the digital adder is converted to analog by a 4-bit DAC. This system is demonstrated using both molecular reactions and DNA. A brief comparison of molecular and electronic systems is also presented.**

***Index Terms*—Molecular systems, DNA, analog, digital, logic, signal processing, digital signal processing, analog-to-digital conversion, digital-to-analog conversion**

I. Introduction

The field of Molecular computing in general and DNA computing in particular have advanced remarkably in last 20-25 years. The progress in the broad field of synthetic biology continues to accelerate at a rate faster than Moore's law that refers to doubling in the number of devices on an integrated circuit (IC) chip every 18 months. A similar growth in synthetic biology is referred as Carlson's law [1], [2]. Early publications on molecular computing [3] and DNA computing [4] demonstrated the ability to compute using biological and chemical molecules, as an alternative to computing using silicon ICs [5]. Molecular computing has the potential to revolutionize monitoring concentrations or rates of change of concentrations of proteins and targeted drug delivery.

There is significant interest in synthesis of molecular circuits, *in vitro* or *in vivo*, to understand biological processes or to re-engineer them by implementing particular computations [6]-[11]. This broad field, referred as synthetic biology, has seen remarkable progress since its inception in 2000. Efforts in synthetic biology have culminated in manipulation or even construction of a variety of molecular systems [12]-[15].

Molecular computing does not need to compete with traditional computing. Rather it is meant for activating or inhibiting pathways or monitoring proteins or delivering drugs at a very slow rate. The computation rates in molecular systems are typically 10-15 orders of magnitude slower than traditional computing. For example, monitoring a protein 4 times a day requires a sample period of 21,600 seconds compared to a clock period of 1 ns for a clock speed of 1 GHz. Fortunately, today's DNA circuits can meet these sample rate constraints for simple circuits. As the molecular computing technology evolves, it will be possible to realize molecular circuits that can implement computationally complex operations and/or at faster rates. Furthermore, massive parallelization is an important advantage of chemical and biological systems. For example it is known that more than 10M reactions per second can be performed in a human cell [16]. This power can be exploited for a dense data processing or storage in a cell-sized area.

A molecular system consists of a set of chemical reactions where reactants combine to form products. For example (1) shows a molecular reaction where S and E are reactants, P and E are products and k is the rate constant.

$$S + E \xrightarrow{\mathrm{k}} P + E \tag{1}$$

In this reaction, one molecule of E combines with one molecule of S to produce one molecule of P and one molecule of E. The dynamic behavior of reactant and product concentrations can be modeled using mass-action kinetic model. In this model the ordinary differential equations are used to model the concentration of each species as a function of time [17] [18]. For reaction (1) we have

$$\begin{array}{rcl} \frac{dP}{dt} & = & -\frac{dS}{dt} = kSE \\ \frac{dE}{dt} & = & 0 \end{array} \tag{2}$$

The speed of reaction is proportional to the concentration of inputs and the rate constant.

Whereas the mass-action kinetics model is a natural language for describing biochemical reactions, additional issues need to be considered for achieving more accurate modeling of biochemical systems by ODEs from mass-action model. For

This work is supported by NSF Grants, CCF-1423407, CCF-1117168, and CCF-0946601.

example if reaction (1) represents an enzymatic reaction with E, S and P are enzyme, substrate and product, respectively, the Michaelis-Menten kinetics provides a more accurate modeling of this reaction. Based on this model enzyme reactions are initiated by a binding interaction between the enzyme (E) and the substrate (S) to form a complex (ES), which in turn is converted into a product (P) and the enzyme. This can be represented by the following reactions:

$$S + E \underset{k_r}{\overset{k_f}{\rightleftharpoons}} SE \xrightarrow{\mathrm{k_{cat}}} P + E \tag{3}$$

where k_f, k_r, and k_{cat} denote the rate constants [19] and the double arrows between S and ES represent the fact that enzyme-substrate binding is a reversible process. The ODEs for these reaction can be solved for different assumptions of the system to obtain the dynamic behavior of molecules. For example for a more probable situation, when the enzyme concentration is much less than the substrate concentration, the rate of product formation is given by

$$\frac{dP}{dt} = V_{max}\frac{S}{K_M + S} = k_{cat}E_0\frac{S}{K_M + S}. \tag{4}$$

K_M is Michaelis constant and defined as the substrate concentration at which the reaction rate is at half-maximum and E_0 is the initial concentration of the enzyme. K_M and E_0 are specific for each enzyme and can be obtained by experiment and are available for most of enzymes.

Different $in\ vivo$ or $in\ vitro$ mediums have been used for synthesizing a desired molecular system. For example, for $in\ vivo$ case, promising approaches have used RNA interference(RNAi) and silencing (siRNA) to construct logic gates [20], [21], [22]. For the $in\ vitro$ the DNA strand-displacement [23] is a well established medium for implementing and scaling up molecular systems. In this paper we use DNA strand-displacement as the medium for implementing and simulating our designs.

Computing or signal processing systems can either be analog or discrete-time. In *analog* processing, the input and output correspond to continuous-time signals. In *discrete-time* processing, the continuous-time signal is first sampled using a sampler, then processed in discrete time steps, and finally converted to a continuous-time signal if necessary by some form of interpolation. If the sampled signal in a discrete-time system is also discretized in amplitude, then it is referred to as a *digital* signal. A digital signal processing (DSP) system requires an analog-to-digital converter (ADC), processing of digital signals and finally a digital-to-analog conversion (DAC). Most information processing systems today store, process or transmit digital information. Discrete-time signal processing provides significantly higher accuracy than continuous-time since the delay elements can be realized with high-precision. In [24], it was recognized that the strength of a molecule was significantly degraded in an analog delay line with increase in the order of the system or the number of delays. In contrast, delay lines implemented in a discrete-time molecular or DNA system do not suffer from significant degradation. Digital processing provides even higher robustness and precise control in processing the signal in temporal or spectral domain than discrete-time signals. We differentiate discrete-time as sampled in time but continuous in amplitude and digital as sampled in time and discretized in amplitude.

This paper presents synthesis of molecular computing systems that can be analog, discrete-time or digital. Analog and discrete-time processing of molecular systems have been published before. Synthesizing molecular and DNA reactions to implement continuous-time linear filters was first presented in [25]. Signal processing systems, implemented as either discrete-time or digital, contain delay elements. Delay elements transfer the molecules from their inputs to outputs without altering the concentration every computation cycle. Delay elements were first synthesized using molecular reactions in [26]. These systems can operate either in a fully-synchronous manner [27] using a two-phase clock, or in a locally-asynchronous globally-synchronous manner [26], [28], or in a fully-asynchronous manner [29] and [30]. The goal of this paper is two-fold. First, this paper presents a review of past work on continuous-time and discrete-time processing systems. Second, a new methodology to synthesize molecular ADCs and molecular DACs are presented. Molecular and DNA implementations of a complete digital processing system using ADC, digital computing and DAC are presented. These molecular designs can be scaled up with respect to their complexity. However, due to the resource limitation in living cells, they are more suitable for $in\ vitro$ implementation.

This paper is organized as follows. In Section II, We provide a brief review of continuous-time systems and illustrate molecular implementation of a simple first-order analog filter. Discrete-time signal processing systems using fully-synchronous and locally-asynchronous globally-synchronous manner are reviewed in Section III. Molecular circuits for ADCs, digital logic, DACs, and a complete digital processing system that adds two numbers using 2 ADCs, one digital adder, and 1 DAC are described in Section IV. DNA implementation of the complete digital system and its simulation results are presented in Section V. A comparison of molecular and electronic systems is presented in Section VI. Finally Section VII provides discussion and concluding remarks.

II. Molecular Continuous-Time Systems

Molecular implementations of continuous-time or analog systems have been described in many past publications [31]-[34]. Study of analog molecular systems is important since it has been proven that computations in living cells are mostly analog [31]-[33].

Analog computations can be implemented with chemical reaction networks (CRNs) efficiently with respect to the number of reactions and molecular species. For example, as presented for the first time in [33] and [34], implementing a molecular adder via analog computation is simple: we have two input concentrations to be added; both are transferred to the same molecular type by means of two independent reactions. In one application of an *in vivo* analog adder, two inputs may correspond to regulating the expressions of a common protein from two independent genetic promoters [33]. Analog multiplication can be simply implemented by two

molecular reactions [35]:

$$\begin{array}{ccc} x+y & \xrightarrow{k_1} & x+y+z \\ z & \xrightarrow{k_2} & \varnothing \end{array} \tag{5}$$

From mass-action kinetics model we have

$$\frac{dz}{dt} = k_1 xy - k_2 z \tag{6}$$

where x,y, and z are molecular concentrations of their corresponding molecular types. In the steady-state $\frac{dz}{dt} = 0$, thus, $z = \frac{k_1}{k_2}xy$. The output z represents a scaled version of the product xy. Analog implementation of more complex functions such as square roots and logarithmic additions have been presented in [34]. Implementation of linear continuous-time systems with biochemical reactions has been presented in [25]. We briefly describe this method with an example. Each signal, u, is represented by the difference in concentration between two particular molecular types, u^+ and u^-, where u^+ and u^- are defined as:

$$u^+ = \begin{cases} u & if \quad u > 0 \\ 0 & otherwise \end{cases} \tag{7}$$

and

$$u^- = \begin{cases} |u| & if \quad u < 0 \\ 0 & otherwise. \end{cases} \tag{8}$$

Any linear continuous-time system can be implemented using three building blocks: integrator, gain and summation. Using mass-action kinetics model, these blocks can be approximated by a minimal set of chemical reactions, referred as: catalysis, degradation, and annihilation reactions described by (9), (10), and (11), respectively.

$$u^{\pm} \xrightarrow{\gamma} u^{\pm} + y^{\pm} \tag{9}$$

$$u^{\pm} \xrightarrow{\gamma} \varnothing \tag{10}$$

$$u^+ + u^- \xrightarrow{\eta} \varnothing, \tag{11}$$

where γ and $\eta \in R^+$. Reaction (9) is a concise representation of the following two reactions:

$$\begin{array}{ccc} u^+ & \xrightarrow{\gamma} & u^+ + y^+ \\ u^- & \xrightarrow{\gamma} & u^- + y^-. \end{array} \tag{12}$$

This notation is also adopted for other reactions with double superscripts. For each molecular type, an annihilation reaction is necessary to ensure a minimal representation of the molecule. For example, if y is used in a reaction network, the reaction $y^+ + y^- \longrightarrow \varnothing$ should be added.

Integration: Reactions (13) implement integration, $y(t) = \int_0^t u(\tau)d\tau + y(0)$ with $t \in R$:

$$u^{\pm} \xrightarrow{\alpha} u^{\pm} + y^{\pm}, \tag{13}$$

where $\alpha \in R^+$. For these reactions we have

$$\left.\begin{array}{rcl} \frac{dy^+}{dt} & = & \alpha u^+ \\ \frac{dy^-}{dt} & = & \alpha u^- \end{array}\right\} \Rightarrow \frac{dy}{dt} = \frac{dy^+}{dt} - \frac{dy^-}{dt}$$
$$= \alpha u^+ - \alpha u^- = \alpha u \Rightarrow y(t) = \alpha \int_0^t u(\tau)d\tau + y(0). \tag{14}$$

Gain and Summation: The following reactions output a linear combination of the input signals, u_i, with corresponding gain k_i.

$$\begin{array}{ccc} u_i^{\pm} & \xrightarrow{\gamma k_i} & u_i^{\pm} + y^{\pm} \\ y^{\pm} & \xrightarrow{\gamma} & \varnothing, \end{array} \tag{15}$$

where y represents the output, $k_i, \gamma \in R^+$ for $i \in 1, 2, ..., n$. In the special case $n = 1$, this chemical representation approximates the gain block, $y = k_1 u_1$ for $k \geq 0$. For $n \geq 2$ this chemical representation approximates the summation block, $y = \sum_{i=1}^{n} k_i u_i$ [25]. Suppose $U(s)$ and $Y(s)$ represent the Laplace transforms of input and output, respectively. Any linear I/O system with the transfer function $\frac{Y(s)}{U(s)} = \frac{B(s)}{A(s)}$ can be approximated by using integration, gain, and summation blocks where $B(s) = b_n s^n + b_{n-1}s^{n-1} + ... + b_1 s + b_0$ and $A(s) = s^m + a_{m-1}s^{m-1} + ... + a_1 s + a_0$ and $m \geq n$. Figure 1 illustrates how $\frac{Y(s)}{U(s)}$ can be constructed using these basic building blocks [36], [37].

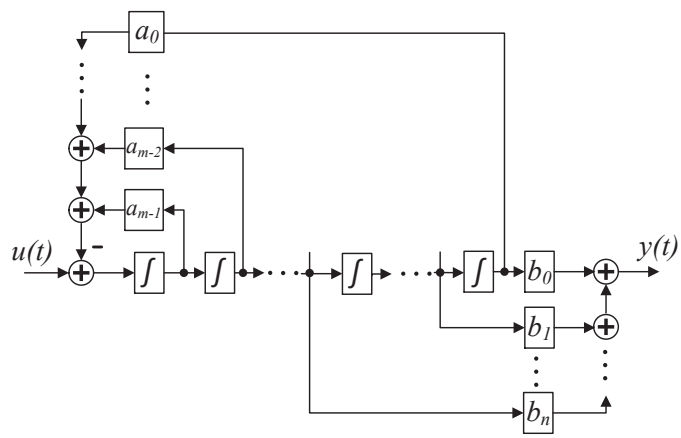


Fig. 1: Constructing linear I/O systems based on transfer function $\frac{Y(s)}{U(s)} = \frac{B(s)}{A(s)}$, using integration, gain, and summation blocks.

A PI controller has been implemented in [25] using these blocks. Here, we illustrate an example molecular implementation of a first-order low-pass continuous-time filter, shown in Figure 2. The transfer function for this filter is $\frac{1}{s+a_0}$. It can be approximated by the following reactions:

$$x(t) = u(t) - a_0 y(t) \rightarrow \left\{ \begin{array}{rcl} y^{\pm} & \xrightarrow{\gamma a_0} & y^{\pm} + x^{\mp} \\ u^{\pm} & \xrightarrow{\gamma} & u^{\pm} + x^{\pm} \\ x^{\pm} & \xrightarrow{\gamma} & \varnothing \\ x^+ + x^- & \xrightarrow{\eta} & \varnothing \\ u^+ + u^- & \xrightarrow{\eta} & \varnothing \end{array} \right. \tag{16}$$

$$\frac{dy}{dt} = x(t) \rightarrow \left\{ \begin{array}{rcl} x^{\pm} & \xrightarrow{\gamma} & x^{\pm} + y^{\pm} \\ y^+ + y^- & \xrightarrow{\eta} & \varnothing \end{array} \right. \tag{17}$$

III. Molecular Discrete-Time Systems

For discrete-time systems the corresponding computations start after the inputs are sampled at specific points in time. In these systems the timings of signal transfers need to be

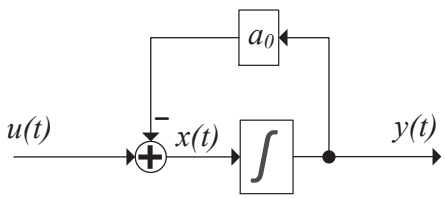


Fig. 2: A first order low-pass continuous-time filter.

synchronized in order to avoid any interference in computations. The concept of a computational cycle in a molecular system is critical. Three different synchronization schemes have been proposed; these include: fully-synchronous, globally-synchronous locally-asynchronous, and fully-asynchronous. Fully synchronous systems are synchronized by a two-phase clock [27], [26]. In a globally-synchronous locally-asynchronous systems, three proteins, referred as Red (R), Green (G) and ($Blue$) are introduced. The transfer of R to G, G to B and B to R completes a computational cycle. The global *RGB clock* provides global synchronization [28], [26]. Fully-asynchronous systems do not make use of any global clock [29], [30]. Typically, RGB clocked systems are the fastest, while the fully-asynchronous systems are the slowest as these involve more phases of transfers. The protein transfer operation is a slow operation and is the bottleneck in molecular systems with respect to sample period. Although fully-synchronous systems require a two-phase clock, this clock is designed from a 4-phase protein transfer mechanism. This paper presents a brief review of the fully-synchronous and the RGB systems. Fully-asynchronous systems are not reviewed in this paper; however, the reader is referred to [29], [30].

All reactions in the discrete-time system are implemented using only two coarse rate categories for the reaction rate constants, i.e., k_{fast} and k_{slow}. Given reactions with any such set of rates, the computation is correct. It does not matter how fast the fast reactions are or how slow the slow reactions are - only that all fast reactions fire relatively faster than slow reactions. We illustrate both schemes with a simple example, a moving-average filter. In fact, it is a first-order discrete-time low-pass filter. The circuit diagram for the filter is shown in Figure 3. It produces an output value that is one-half the current input value plus one-half the previous value. Given a time-varying input signal X, the output signal Y is a moving average, i.e., a smoother version of the input signal. Since there is no feedback in the system, it is called a *finite impulse response* (FIR) filter [38].

A. Fully-Synchronous Framework

In this framework a global clock signal synchronizes signal transfers in the system. For a molecular clock, reactions are chosen that produce sustained oscillations in terms of chemical concentrations. With such oscillations, a low concentration corresponds to a logical value of zero; a high concentration corresponds to a logical value of one. Techniques for generating chemical oscillations are well established in the literature. Classic examples include the Lotka-Volterra, the

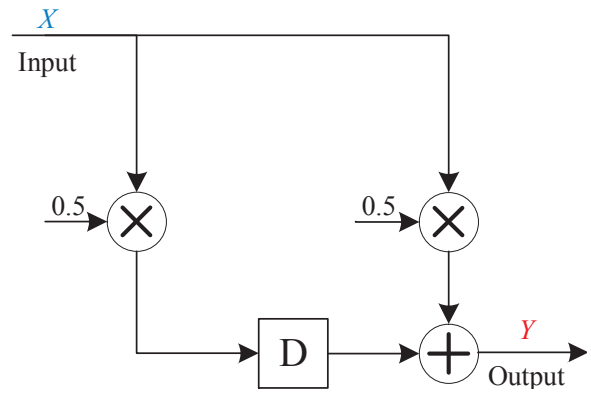


Fig. 3: Block diagram for the moving-average filter [26].

Brusselator, and the Arsenite-Iodate-Chlorite systems [39], [40]. Unfortunately, none of these schemes is quite suitable for synchronous sequential computation: the required clock signal should be symmetrical, with abrupt transitions between the phases. A new design was proposed in [26] and [27] for multi-phase chemical oscillator. For a 4-phase oscillator the phases can be represented by molecular types *R, G, B, V*. First consider the reactions

$$\begin{aligned} 2S_r &\xrightarrow{\mathrm{k_{slow}}} r + 2S_r \\ 2S_g &\xrightarrow{\mathrm{k_{slow}}} g + 2S_g \\ 2S_b &\xrightarrow{\mathrm{k_{slow}}} b + 2S_b \\ 2S_v &\xrightarrow{\mathrm{k_{slow}}} v + 2S_v \end{aligned} \tag{18}$$

$$\begin{aligned} R + r &\xrightarrow{\mathrm{k_{fast}}} R \\ G + g &\xrightarrow{\mathrm{k_{fast}}} G \\ B + b &\xrightarrow{\mathrm{k_{fast}}} B \\ V + v &\xrightarrow{\mathrm{k_{fast}}} V. \end{aligned} \tag{19}$$

In reactions (18), the molecular types *r, g, b, v* are generated slowly and constantly, from source types S_r, S_g, S_b, S_v, whose concentrations do not change with the reactions. In reactions (19), the types *R, G, B, V* quickly consume the types *r, g, b, v*, respectively. Call *R, G, B, V* the phase signals and *r, g, b, v* the absence indicators. The latter are only present in the absence of the former. The reactions

$$\begin{aligned} R + v &\xrightarrow{\mathrm{k_{slow}}} G \\ G + r &\xrightarrow{\mathrm{k_{slow}}} B \\ B + g &\xrightarrow{\mathrm{k_{slow}}} V \\ V + b &\xrightarrow{\mathrm{k_{slow}}} R \end{aligned} \tag{20}$$

transfer one phase signal to another, in the absence of the previous one. The essential aspect is that, within the *R, G, B, V* sequence, the full quantity of the preceding type is transferred to the current type before the transfer to the succeeding type begins. To achieve sustained oscillation, we introduce positive

feedback. This is provided by the reactions

$$
\begin{aligned}
2G &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_G \\
R + I_G &\xrightarrow{k_{slow}} 3G \\
2B &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_B \\
G + I_B &\xrightarrow{k_{slow}} 3B \\
2V &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_V \\
B + I_V &\xrightarrow{k_{slow}} 3V \\
2R &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_R \\
V + I_R &\xrightarrow{k_{slow}} 3R
\end{aligned}
\tag{21}
$$

Consider the first two reactions. Two molecules of G combine with one molecule of R to produce three molecules of G. The first step in this process is reversible: two molecules of G can combine, but in the absence of any molecules of R, the combined form will dissociate back into G. So, in the absence of R, the quantity of G will not change much. In the presence of R, the sequence of reactions will proceed, producing one molecule of G for each molecule of R that is consumed. Due to the first reaction $2G \xrightarrow{k_{slow}} I_G$, the transfer will occur at a rate that is super-linear in the quantity of G; this speeds up the transfer and so provides positive feedback. Suppose that the initial quantity of R is set to some non-zero amount and the initial quantity of the other types is set to zero. We will get an oscillation among the quantities of R, G, B, and V.

One requirement for a clock in synchronous computation is that different clock phases should not overlap. A two-phase clock is used for synchronous structures: concentrations of molecular types representing clock phase 0 and clock phase 1 should not be present at the same time. To this end, two nonadjacent phases, say *R* and *B* in a four-phase *RGBV* oscillator, are chosen as the clock phases. The scheme for chemical oscillation works well. Figure 4 shows the concentrations of R and B as a function of time, obtained through differential equation simulations of the Reactions (18), (19), (20), and (21). It may be noted that the two phases *R* and *B* are essentially non-overlapping.

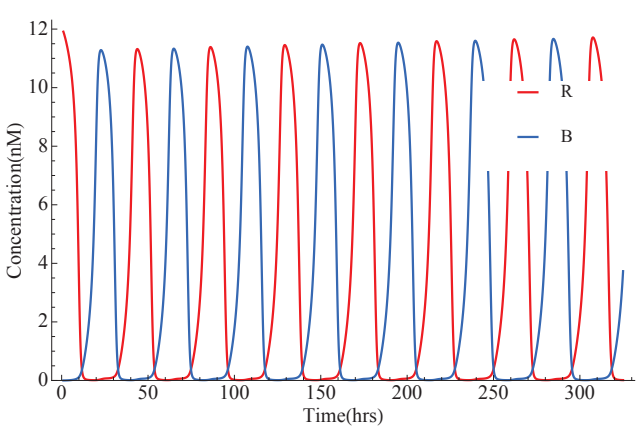


Fig. 4: simulation results for R and B phases of a four-phase oscillator [26].

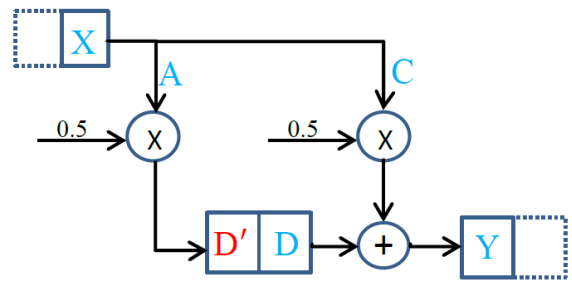


Fig. 5: Block diagram for synchronous implementation of the moving-average filter [26].

S1	S2
$B + X \xrightarrow{k_{slow}} A + C + B$	$R + D' \xrightarrow{k_{slow}} D + R$
$2\,A \xrightarrow{k_{fast}} D'$	
$2\,C \xrightarrow{k_{fast}} Y$	
$B + D \xrightarrow{k_{slow}} Y + B$	

Fig. 6: Set of molecular reactions for the synchronous implementation of the moving-average filter [26].

The delay and computation elements for the moving average filter in Figure 5 are implemented by the reactions in Figure 6. As Figure 5 shows each delay element, D, is modeled by two molecular types, D and D'. In the presence of B, the input signal X is transferred to molecular types A and C; these are both reduced to half and transferred to D' and Y, respectively. In the presence of R, D' is transferred to D. Therefore, in the following phase B, half of the new sample adds with the half of the previous sample stored in D.

B. Globally-Synchronous Locally-Asynchronous Framework

The globally-synchronous locally-asynchronous framework is illustrated in Figure 7. It contains no clock signal; rather it is "self-timed" in the sense that a new phase of the computation begins when an external sink removes the entire quantity of molecules Y, i.e., the previous output value, and supplies a new quantity of molecules X, i.e., the current input value. Each delay element in this framework is modeled by three molecular types, namely RGB. Figure 8 shows how the computations in asynchronous framework are performed in three phases and how delay elements are implemented using three molecular types R_i, G_i, B_i. In this framework, the moving-average filter is implemented by the reactions in Figure 9. The molecular types corresponding to signals are X, A, C, R, G, B, and Y. To illustrate the design, we use colors to categorize some of these types into three categories: Y and R in red; G in green; and X and B in blue. The group of the first three reactions shown in the $S1$ column of Figure 9 transfers the concentration of X to A and to C, a $fanout$ operation. The concentrations of A and C are both reduced

S1	S2	S3	S4
$g + X \xrightarrow{k_{slow}} A + C$	$2\,R \xrightarrow{k_{fast}} 2R + R'$	$2S_r \xrightarrow{k_{slow}} 2S_r + r$	$R' + X \xrightarrow{k_{fast}} A + C$
$2\,A \xrightarrow{k_{fast}} Y$	$2\,Y \xrightarrow{k_{fast}} 2Y + R'$	$2S_g \xrightarrow{k_{slow}} 2S_g + g$	$G' + R \xrightarrow{k_{fast}} G$
$2\,C \xrightarrow{k_{fast}} R$	$2\,G \xrightarrow{k_{fast}} 2G + G'$	$2S_b \xrightarrow{k_{slow}} 2S_b + b$	$B' + G \xrightarrow{k_{fast}} B$
$b + R \xrightarrow{k_{slow}} G$	$2\,B \xrightarrow{k_{fast}} 2B + B'$	$R' + r \xrightarrow{k_{fast}} R'$	$R' + B \xrightarrow{k_{fast}} Y$
$r + G \xrightarrow{k_{slow}} B$	$2\,X \xrightarrow{k_{fast}} 2X + R'$	$G' + g \xrightarrow{k_{fast}} G'$	
$g + B \xrightarrow{k_{slow}} Y$	$2\,R' \xrightarrow{k_{fast}} \emptyset$	$B' + b \xrightarrow{k_{fast}} B'$	
	$2\,G' \xrightarrow{k_{fast}} \emptyset$		
	$2\,B' \xrightarrow{k_{fast}} \emptyset$		

Fig. 9: Set of molecular reactions for the asynchronous implementation of the moving-average filter [26].

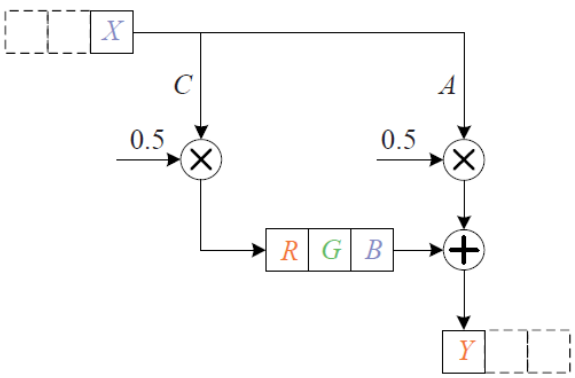


Fig. 7: Block diagram for the asynchronous implementation of the moving-average filter [26].

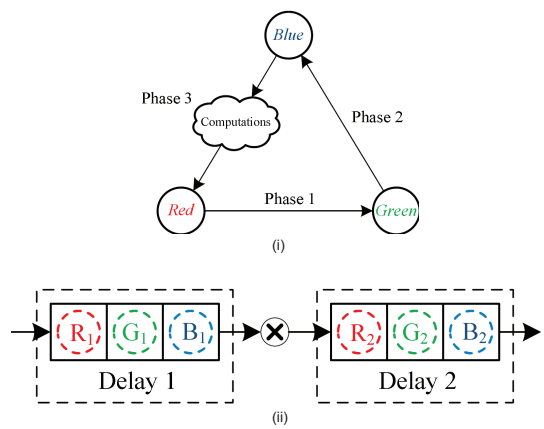


Fig. 8: (i) Implementing delay elements using the 3-phase asynchronous scheme. (ii) Cascaded delay elements implemented using asychronous scheme [26].

to half, scalar multiplication operations. The concentration of A is transferred to the output Y, and the concentration of C is transferred to R. The transfer to R is the first phase of a delay operation. Once the signal has moved through the delay operation, the concentration of B is transferred to the output Y. Since this concentration is combined with the concentration of Y produced from A, this is an addition operation. The final group of three reactions shown in the $S1$ column of Figure 9 implements the delay operation. The concentration of R is transferred to G and then to B. Transfers between two color categories are enabled by the absence of the third category: red goes to green in the absence of blue; green goes to blue in the absence of red; and blue goes to red in the absence of green. The reactions are enabled by molecular types r, g, and b that we call absence indicators. The absence indicators ensure that the delay element takes a new value only when it has finished processing the previous value. In the group of reactions shown in the $S2$ column of Figure 9 molecules of types R', G', and B' are generated from the signal types that we color-code red, green, and blue, respectively. The concentrations of the signal types remain unchanged. This generation/consumption process ensures that equilibria of the concentrations of R', G', and B' reflect the total concentrations of red, green, and blue color-coded types, respectively. Accordingly, we call R', G', and B' color concentration indicators. They serve to speed up signal transfers between color categories, and provide global synchronization.

In the group of reactions shown in the $S3$ column of Figure 9, molecules of the absence indicator types r, g, and b are generated from external sources S_r, S_g, and S_b. At the same time, they are consumed when R', G', and B' are present, respectively. Therefore, the absence indicators only persist in the absence of the corresponding signals: r in the absence of red types; g in the absence of green types; and b in the absence of blue types. They only persist in the absence of these types because otherwise "fast" reactions consume them quickly.

Finally, the reactions shown in the S4 column of Figure 9 provide positive feedback kinetics. These reactions effectively speed up transfers between color categories as molecules in

one category are "pulled" to the next by color concentration indicators. Note that the concentration of the input X is sampled in the green-to-blue phase. The output Y is produced in the blue-to-red phase.

Although the RGB scheme doesn't have an independent global clock signal it provides a global synchronization by categorizing signals into three phases, so called RGB phases. Many local RGB blocks enable locally-asynchronous computation while global color concentrations, R',G',B', provide global synchronization. In fact, they form a nonsymmetric clock dependent on the signal values of local RGB blocks.

Another fully-asynchronous framework has been proposed in [30]. In a fully-asynchronous system signal transfers and computations start from the input of the system and progress to its output in multiple phases. Each delay element in a fully-asynchronous system is modeled by two molecular types [30] and introduces two phases. Reactions for each phase are fired as soon as the preceding phase is completed. In addition to FIR filter, an IIR filter and an 8-point FFT have been implemented using this framework [26], [30].

IV. Molecular Sensing and Digital Computing Systems

Although analog computing systems are important due to their efficiency and their application in *in vivo* systems, digital computing systems are more robust [41], [33], [42]. In fact, regardless of the implementation technology, the fundamental reason for the robustness of the digital computation lies in information theory: information is coded across many 1-bit-precise interacting computational channels in the digital approach but on one channel in the analog approach [33].

Although complex molecular digital systems may be impractical today, these will be practical in near future as synthetic biology is seeing remarkable progress for synthesizing more complex systems *in vitro* especially from DNA. As a practical *in vitro* example, implementation of a scalable digital system, so called *seesaw gates*, with DNA strand-displacement reactions have been used to implement simple logical AND/OR gates, and 2-bit-precise square roots in [42].

Roughly speaking, in a digital molecular system, absence or existence of a molecular type defines whether the related signal is logically '0' or '1', respectively. More precisely, if the concentration of a molecular type is close to 0 nM it represents logical '0', while if it is close to a distinguishable nonzero value, it represents logical '1'. In this paper, for *in vitro* DNA implementations, we consider concentrations near 1 nM as the logical value '1' and near 0 nM as logical value '0'.

Molecular digital systems require molecular analog-to-digital conversion (ADC). This paper, presents a new molecular implementations of ADCs and DACs. Figure 10 illustrates a complete digital system.

We present molecular implementations of a k-bit analog to digital converter and a k-bit digital to analog converter. We also review the molecular implementation of basic digital logic gates. Using these gates, we demonstrate a 3-bit molecular binary adder including two ADCs required to sample and digitize the two input operands and a DAC to output an analog signal. A DNA implementation of the complete system is also demonstrated in Section V. It can be noted that all of the molecular reactions are rate-independent. In other words, no matter what the speed rates of the reactions are and how they may change during the computation, the steady-state concentrations compute the correct desired outputs.

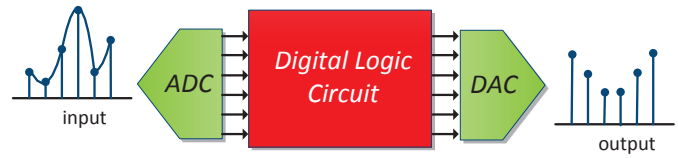


Fig. 10: Block diagram of a general system developed in this paper.

A. Analog to Digital Converter (ADC)

This subsection describes molecular implementation of analog to digital converter. A 3-bit example is considered. Let the input molecular type, i, have an analog concentration between 0 nM and 8 nM. The output is a 3-bit digital number $x = x_2x_1x_0$. Each bit is considered as logical '0' if its concentration is approximately 0 nM and logical '1' if its concentration is approximately 1 nM.

We start with the most significant bit, x_2. This bit should be set to 1 when i is larger than 4 nM and to zero when i is less than 4 nM. Reactions (22) implement a one-bit comparator that determines x_2. The initial concentration of T_2 represents the threshold for the comparator which is set to 4 nM.

$$\begin{array}{rcl} i + T_2 & \longrightarrow & w_2 \\ i + x_{2n} & \longrightarrow & x_2 + i \\ T_2 + x_2 & \longrightarrow & x_{2n} + T_2 \end{array} \tag{22}$$

In the first reaction, i and T_2 molecules combine and the one with larger initial concentration remains and the other one vanishes. The first reaction is independent of the second and third reactions because i and T_2 remain unaltered in the second and third reactions. However, activation of the second or third reactions depends on the outcome of the first reaction. After completion of the first reaction only one of the second or third reactions is active. If i is larger than T_2, the third reaction stops firing while the second reaction transfers all molecules of x_{2n} to x_2. Alternately, if i is less than T_2, second reaction stops and third reaction transfers x_2 to x_{2n} completely. x_2 and x_{2n} are initialized to 0 nM and 1 nM, respectively. Note that in general for a k-bit ADC, each bit, i.e., x_j where $j = 0, 1, ..., k-1$, is modeled by two molecular types, i.e., x_j and x_{jn}, called the bit and its complement molecular types. All of the x_j species are initialized to 0 nM and x_{jn} species are initialized to 1 nM. Furthermore, for each j, the total concentration of x_j and x_{jn}, is constantly 1 nM, i.e., if the concentration of x_j is C, then the concentration of x_{jn} is $(1 - C)$, both in nM.

Table I shows the final concentrations for i, x_2 and w_2 after completion of Reactions (22). i_0 denotes the initial concentration of i. If $i_0 > T_2$ then i can be used to compute the second bit of x, *i.e.*, x_1. If $i_0 < T_2$ then w_2 can be used to determine x_1. Reactions (23) and (24) determine x_1 for the

TABLE I: Stable concentration of molecules i, x_2, and w_2 after completion of Reactions (22).

	i	w_2	x_2
$i_0 < 4$	0	i_0	0
$i_0 > 4$	$i_0 - 4$	4	1

TABLE II: Stable molecular concentrations after completion of Reactions (23) and (24).

	i	w_2	w_1	w_1'	x_2	x_1
$i_0 < 2$	0	0	0	$\underline{i_0}$	0	0
$2 < i_0 < 4$	0	$\underline{i_0 - 2}$	0	2	0	1
$4 < i_0 < 6$	0	4	$\underline{i_0 - 4}$	0	1	0
$6 < i_0$	$\underline{i_0 - 6}$	4	2	0	1	1

above two cases. The initial concentrations for both threshold molecules, T_1 and T_1', are equal to 2 nM. Similar to Reactions (22), the first three reactions of (23) implement a one-bit comparator. However, here, the molecular concentration of i and T_1 are compared to determine x_1 when x_2 is nonzero. This is equivalent to comparing initial i_0 to 6 nM. Similarly the first three reactions of (24) compare w_2 and T_1' to determine x_1 when x_2 is zero. This is equivalent to comparing initial i_0 to 2 nM.

$$\begin{aligned} x_2 + i + T_1 &\longrightarrow w_1 + x_2 \\ x_2 + i + x_{1n} &\longrightarrow x_1 + i + x_2 \\ x_2 + T_1 + x_1 &\longrightarrow x_{1n} + T_1 + x_2 \\ x_{2n} + w_1 &\longrightarrow i + T_1 + x_{2n} \end{aligned} \tag{23}$$

$$\begin{aligned} x_{2n} + w_2 + T_1' &\longrightarrow w_1' + x_{2n} \\ x_{2n} + w_2 + x_{1n} &\longrightarrow x_1 + w_2 + x_{2n} \\ x_{2n} + T_1' + x_1 &\longrightarrow x_{1n} + T_1' + x_{2n} \\ x_2 + w_1' &\longrightarrow w_2 + T_1' + x_2 \end{aligned} \tag{24}$$

Before the concentration of x_2 reaches its stable value, both x_2 and x_{2n} may have nonzero concentrations and both sets of Reactions (23) and (24) can be fired. The fourth reactions of (23) and (24) are added to undo undesired reactions fired during the transient time. For example, when the final concentration of x_2 is zero the fourth reaction of (23) transfers w_1 back to i and T_1 in order to undo the first reaction. The initial concentrations for x_1 and x_{1n} are 0 nM and 1 nM, respectively. After x_1 and x_{1n} are stabilized to their final concentrations, depending on the initial value of i, one of them has the concentration of 1 nM and the other 0 nM.

Except i, none of the molecular types participating in Reactions (22) is altered by Reactions (23) and (24). However, Reactions (23) and (24) need the final concentrations of x_2 and x_{2n} from Reactions (22). Thus, the concentrations of molecules of Reactions (23) and (24) reach stable values after reactions in (22) are completed. For different values of i_0, Table II shows the final concentrations after Reactions (23) and (24) are completed.

Finally, in order to determine the least significant bit (LSB) of x, i.e., x_0, depending on i_0's value, the molecular types underlined in Table II are used. For each range of i_0, the concentration of its related molecular type is compared to 1 nM to determine x_0. For example when $i_0 > 6$, Reactions (25) are used to determine x_0. The initial concentration of threshold molecules T_0 is 1 nM. Because both x_2 and x_1 are nonzero for $i_0 > 6$, the first three reactions compare i with 1 nM. It is equivalent to comparing i_0 with 7 nM. That is to say, for $i_0 > 6$, x_0=1 nM if i_0 >7 nM and x_0=0 nM if i_0 <7 nM. The last two reactions of (25) are used to undo the undesirable combination of i and T_0 during the transient time when any of x_2 or x_1 is zero.

$$\begin{aligned} x_2 + x_1 + i + T_0 &\longrightarrow w_0 + x_2 + x_1 \\ x_2 + x_1 + i + x_{0n} &\longrightarrow x_0 + i + x_2 + x_1 \\ x_2 + x_1 + T_0 + x_0 &\longrightarrow x_{0n} + T_0 + x_2 + x_1 \\ x_{2n} + w_0 &\longrightarrow i + T_0 + x_{2n} \\ x_{1n} + w_0 &\longrightarrow i + T_0 + x_{1n} \end{aligned} \tag{25}$$

Similarly for each range of i_0 five reactions are used to determine x_0. Due to space limit, these three sets of reactions, each containing five reactions, are not listed here.

The number of bits or the resolution of ADC can be increased by adding the required comparisons and their related undo reactions. In general for k-bit ADC $2^{k+1} + 2(k-1)$ molecular types are required while the number of required reactions is $\sum_{j=1}^{k}(j+2)2^{j-1} = (k+1)2^k - 1$. The precision (sensitivity) of ADC depends on its acceptable input range and the number of its output bits.

Figure 11 shows results for the mass-action kinetic model simulation of the proposed ADC for different values of i_0 .

B. Molecular Digital Logic Circuits

In this section we demonstrate how digital designs can be implemented by molecular reactions. We describe molecular implementations of simple logic AND/OR/XOR gates, a binary adder, and a square-root unit. The method we use here for implementing logical gates is similar to the method presented in [43]. However, in [43] three regulation bit operation reactions are needed for each bit, Whereas these reactions are not required in our complete system implementation due to the self-regulated bits output by the proposed ADC. Here, self-regulated means for each bit only the related molecular type, x_j, or its complement, x_{jn}, but not both, has stable non-zero concentration.

1) Logic Gates: We only consider two-input gates AND, OR, and XOR. Gates with more than two inputs can be easily implemented by cascading two-input gates. Let X and Y denote the inputs of a gate and Z the output.

AND Gate: We start with an AND gate. The output of a logical AND gate is '1' only if both inputs are '1'. It means that if either X='0' or Y='0' then the output Z should be zero. In other words, when concentration of x_n or y_n, i.e., complement molecular types of inputs, is nonzero molecules of z should be transferred to z_n in order to set Z='0'. This can be implemented by Reactions (26).

$$\begin{aligned} x_n + z &\longrightarrow x_n + z_n \\ y_n + z &\longrightarrow y_n + z_n. \end{aligned} \tag{26}$$

When both x and y have stable nonzero concentrations, all molecules of z_n should be transferred to z in order to set Z='1'. This can be implemented by Reactions (27).

$$\begin{aligned} x + y &\longrightarrow x + y + z' \\ 2z' &\longrightarrow \varnothing \\ z' + z_n &\longrightarrow z. \end{aligned} \tag{27}$$

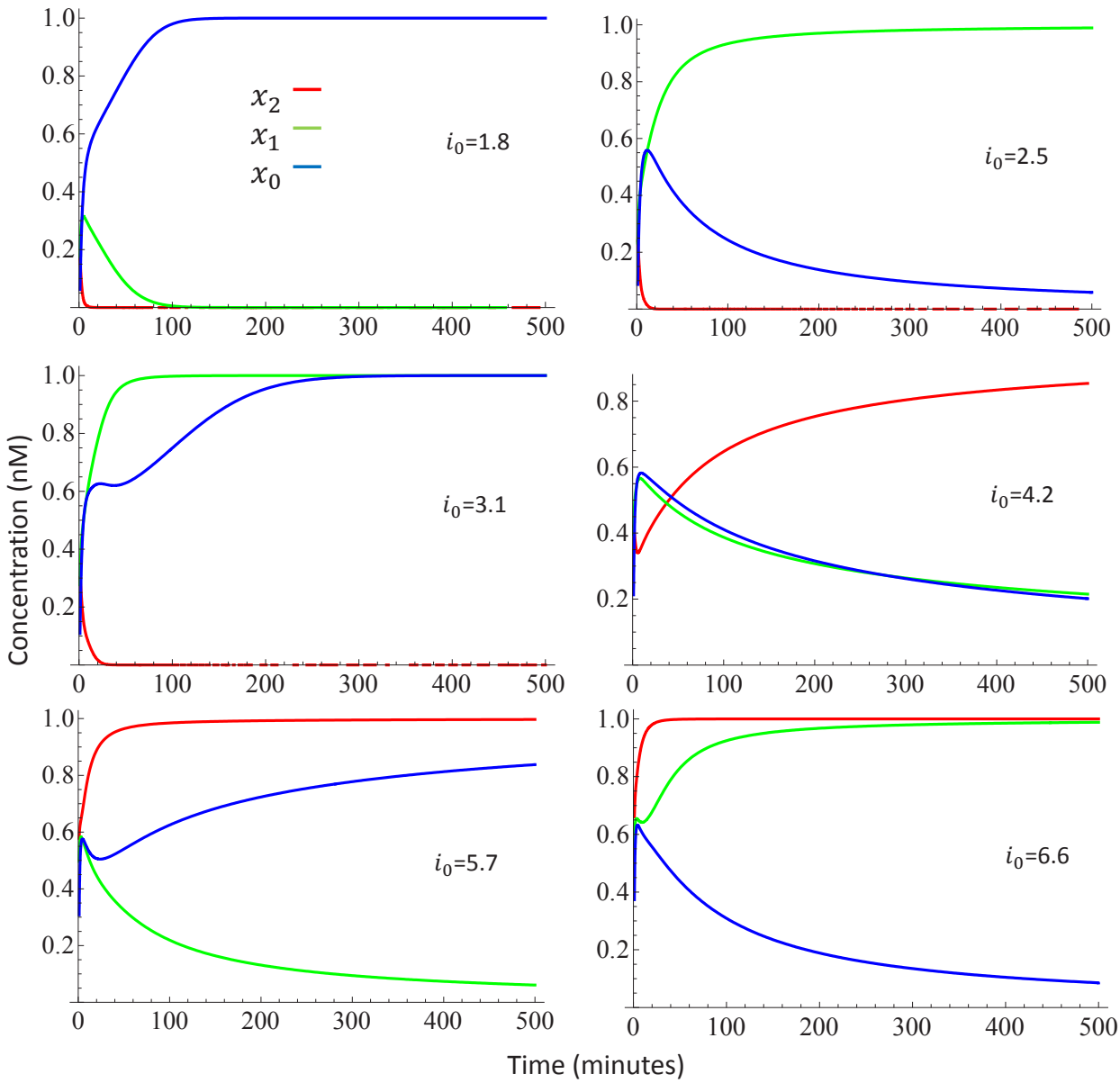


Fig. 11: Simulation results of 3-bit molecular ADC for different input concentrations.

In the first reaction of (27), x combines with y to generate z', an indicator that Z should be set to '1'. z' 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 and y are present, these reactions maintain the concentration of z' at an equilibrium level. When either x or y is not present, z' gets cleared out. In the last reaction, z' transfers z_n to z.

One should note that the input concentrations don't change in logic computations. This enables the outputs of the ADC to be input to other logic gates if needed.

OR Gate: The output of an OR gate is '1' if any of its inputs is '1'. For molecular implementation it means that if either x or y has nonzero concentration then all molecules of z_n should be transferred to z. It is implemented by Reactions (28). In the other case, i.e., when both inputs have zero concentrations, molecules of z should be transferred to z_n as implemented by Reactions (29).

$$\begin{aligned} x + z_n &\longrightarrow x + z \\ y + z_n &\longrightarrow y + z. \end{aligned} \tag{28}$$

$$\begin{aligned} x_n + y_n &\longrightarrow x_n + y_n + z' \\ 2z' &\longrightarrow \varnothing \\ z' + z &\longrightarrow z_n. \end{aligned} \tag{29}$$

XOR Gate: The output of a two-input XOR gate is '1' when inputs are complements of each other. In molecular implementation it means that when either x and y_n or x_n and y have nonzero concentrations, molecules of z_n should be transferred to z as implemented by Reactions (30). For the inputs with the same logical level the output should set to zero and molecules of z should be transferred to z_n. This is

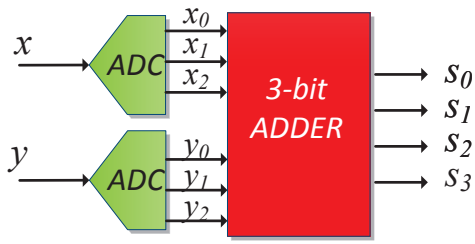


Fig. 13: Block diagram of the system for verifying molecular 3-bit adder.

implemented by Reactions (31).

$$\begin{array}{rcl} x_n + y & \longrightarrow & x_n + y + z' \\ x + y_n & \longrightarrow & x + y_n + z' \\ 2z' & \longrightarrow & \varnothing \\ z' + z_n & \longrightarrow & z. \end{array} \tag{30}$$

$$\begin{array}{rcl} x_n + y_n & \longrightarrow & x_n + y_n + z'_n \\ x + y & \longrightarrow & x + y + z'_n \\ 2z'_n & \longrightarrow & \varnothing \\ z'_n + z & \longrightarrow & z_n. \end{array} \tag{31}$$

NAND, NOR, and XNOR gates can be implemented by exchanging z and its complement in the transfer reactions, z_n in the opposite directions of those of the AND, OR, and XOR gates, respectively.

2) Binary Adder: By cascading AND, OR, and XOR gates we implement more complex digital systems such as a 3-bit adder. The adder consists of one half adder (HA) for the LSB and two full adders (FA) as shown in Figure 12a. Internal schematics of HA and FA are shown in Figure 12b. A general n-bit adder can be easily implemented by extending 3-bit adder using additional FAs for new bits. Cascaded gates for the adder are implemented by molecular reactions presented in Section IV.B. However, other molecular logic gates such as seesaw gates [42] can also be used. In order to verify the functionality of the 3-bit adder we implement the structure shown in Figure 13. Two analog concentrations, x and y, are converted to two 3-bit digital data using the proposed ADC. These two digital numbers are added using the 3-bit adder. The output, $s = s_3s_2s_1s_0$, is a 4-bit digital number representing the digital sum of x and y. Figure 14 shows the simulation results for different concentrations of inputs, x and y.

3) Square-root Unit: As another example of digital computing, we implement square-root of a 4-bit number. Figure 15 shows the schematic of its circuit. In Figure 15, the three-input NAND gate can be implemented by cascading a two-input AND gate with a two-input NAND gate. However, it is more efficient to implement three-input NAND by reactions (32). In these reaction x_1, x_2, and x_3 are inputs and y is the output.

$$\begin{array}{rcl} x_{1n} + y_n & \longrightarrow & x_{1n} + y \\ x_{2n} + y_n & \longrightarrow & x_{2n} + y \\ x_{3n} + y_n & \longrightarrow & x_{3n} + y \\ x_1 + x_2 & \longrightarrow & x_1 + x_2 + x_{12} \\ x_{12} + x_3 & \longrightarrow & x_3 + y' \\ 2y' & \longrightarrow & \varnothing \\ y' + y & \longrightarrow & y_n \end{array} \tag{32}$$

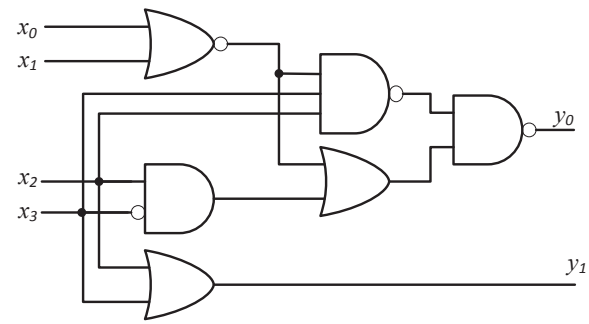


Fig. 15: Schematic for 4-bit Square-root unit.

The strategy used for the direct implementation of three-input NAND in (32) is similar to that of two-input NAND.

Figure 16 shows the simulation results for the square root circuit implemented by molecular reactions.

C. Digital to Analog Converter (DAC)

After performing computations in digital form, in order to convert the computed signal to its analog form, a DAC is required. Using recombinase-based logic and memory, a DAC has been implemented in [44]. For this DAC various digital combinations of the input inducers result in multiple levels of analog gene expression outputs on the basis of the varying strengths of the promoters used and the sum of their respective outputs. This section presents molecular implementations of a k-bit DAC with controlling the impact of each bit on the analog output concentration. Reactions (33) show a 1-bit template for implementing DAC.

$$\begin{array}{rcl} x_j + V_j & \longrightarrow & out + x_j + M_j \\ out + x_{jn} + M_j & \longrightarrow & x_{jn} + V_j \end{array} \tag{33}$$

where x_j and x_{jn}, respectively, represent the input bit and its complement molecular type. out is the analog output of DAC with initial concentration of zero. Molecular type V_j denotes the value of the input bit. In other words, it defines the amount of concentration that is added to the output if input bit, x_j, is nonzero. If x_j is the LSB then V_j is initialized to 1 nM and if it is the bit next to the LSB then V_j is initialized to 2 nM and so on.

Even when the stable value of x_j is zero, during the transient state x_j may have nonzero concentration. The second reaction of (33) prevents undesired output increase due to the nonzero concentration of x_j in transient state. M_j controls the amount of deducted concentration from the output such

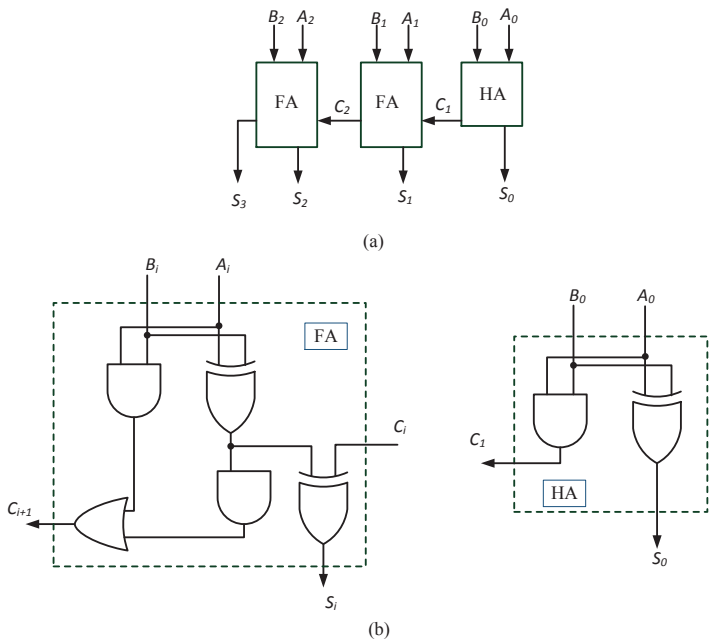


Fig. 12: Schematic of the 3-bit adder; (a) Block diagram, (b) Internal circuits for HA and FA blocks.

that this amount is the same as the amount added to output undesirably during the transient state. In other words, without M_j, the second reaction continues transferring out molecules to V_j during the steady-state. However, this degrades the effects of other bits on the DAC's output, since the molecular type out is common for all bits. The initial concentration for M_j is zero.

The 1-bit template presented here can be easily extended to a $k-bit$ DAC; for each additional bit, one instance of Reactions (33) is added. Therefore, to construct a k-bit DAC, a chemical reaction network including k copies of the 1-bit template are used with proper initial values of V_j. As an example, Reactions (34) illustrate a 4-bit DAC using the proposed template. The initial concentrations of V_0, V_1, V_2, and V_3 are 1, 2, 4, and 8 nM, respectively.

$$\begin{aligned}
x_0 + V_0 &\longrightarrow out + x_0 + M_0 \\
out + x_{0n} + M_0 &\longrightarrow x_{0n} + V_0 \\
x_1 + V_1 &\longrightarrow out + x_1 + M_1 \\
out + x_{1n} + M_1 &\longrightarrow x_{1n} + V_1 \\
x_2 + V_2 &\longrightarrow out + x_2 + M_2 \\
out + x_{2n} + M_2 &\longrightarrow x_{2n} + V_2 \\
x_3 + V_3 &\longrightarrow out + x_3 + M_3 \\
out + x_{3n} + M_3 &\longrightarrow x_{3n} + V_3
\end{aligned} \tag{34}$$

D. A complete molecular digital System

We now illustrate molecular implementation of a digital adder where concentrations of two analog molecules x and y are converted to 3-bit digital, then added using a binary adder, and the 4-bit output is converted to an analog value s. Two molecular ADCs, a molecular digital adder, and a molecular DAC are used to construct a complete system as shown in Figure 17. The functionality of the complete molecular system is verified. Figure 18 shows the simulation results for the complete system illustrated in Figure 17 for different input concentrations.

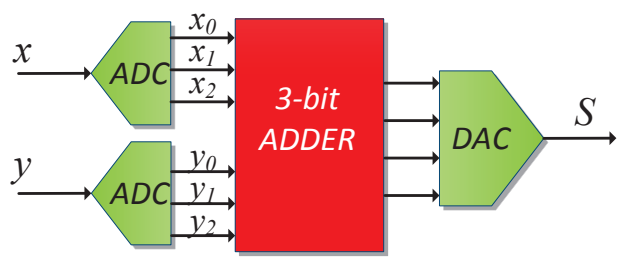


Fig. 17: Block diagram of a simple prototype developed and verified in this paper.

V. DNA Implementation

This section describes mapping of the molecular reactions to DNA. We illustrate mapping the complete digital adder of Section IV.D including ADC, adder and DAC to DNA strand displacement reactions.

Considering each strand (single or double) of DNA as a molecule, it is possible to implement CRNs with DNA strand-displacement mechanism. For example Figure 19 shows DNA strand-displacement primitive for implementing $A+B \underset{r}{\overset{f}{\rightleftharpoons}} C+D$.

Toehold 1 of strand A starts binding to its complement toehold 1^* of B. Then branch migration happens and domain 2 of A displaces domain 2 of strand $2-3$. Finally, toehold 3 and 3^* are separated and two new strands (molecules), C and D, are produced.

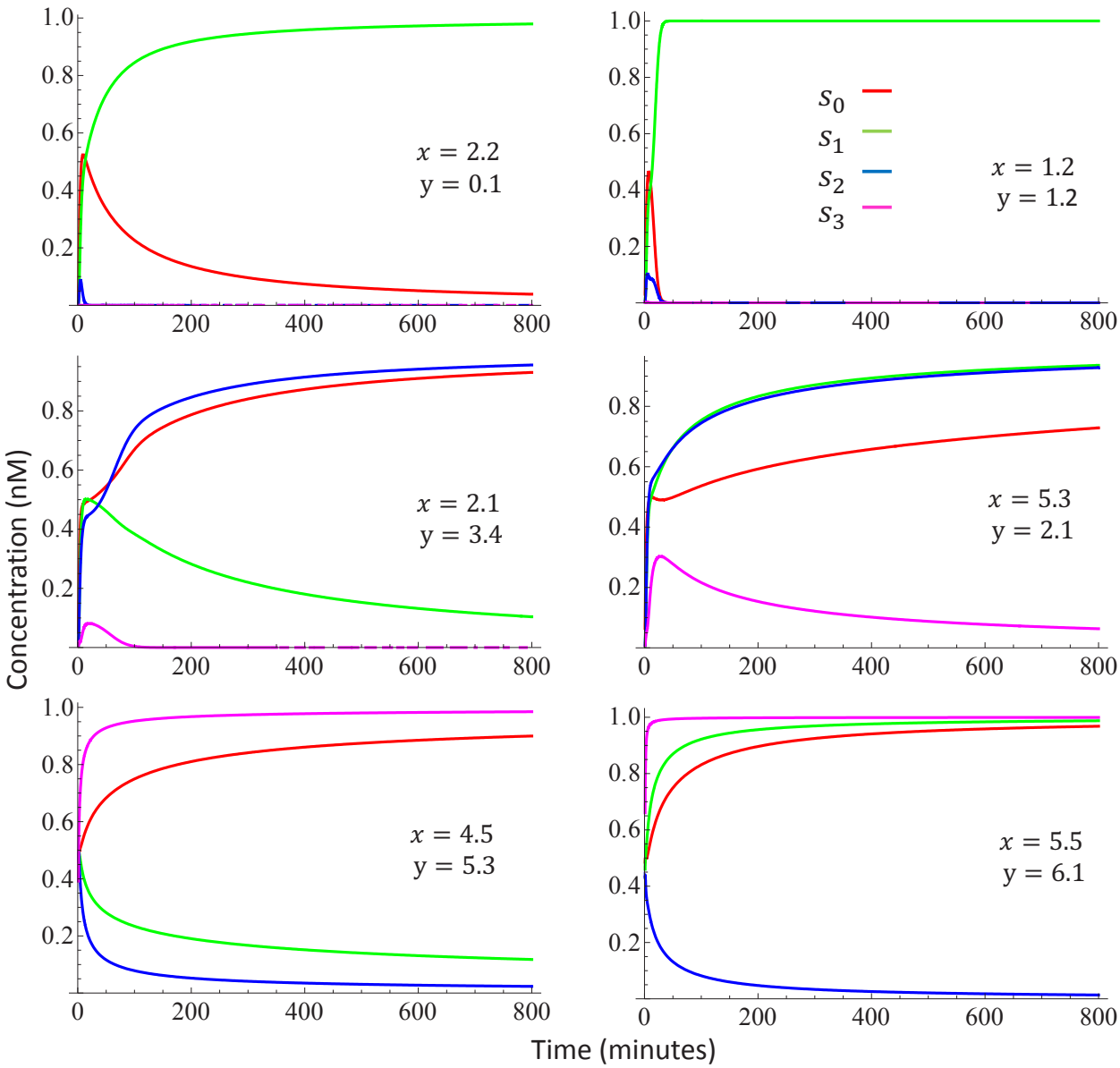


Fig. 14: Simulation results of the molecular implementation of the system shown in Figure 13.

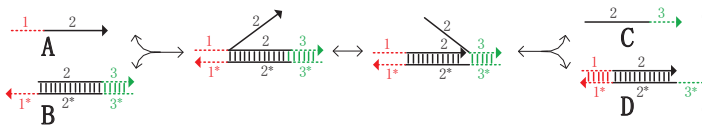


Fig. 19: Implementation of $A + B \underset{r}{\overset{f}{\rightleftharpoons}} C + D$ using DNA strand-displacement mechanism.

A general method of mapping CRNs to DNA strand-displacement reactions has been presented in [23] by Soloveichik, et. al. In their method based on the number of reactants a chemical reaction is converted to a series of DNA strand-displacement reactions similar to Figure 19. Similarly, for our design we generate the corresponding DNA reactions and simulate the system using the kinetic differential equations to characterize the behavior of the system.

The initial concentrations of auxiliary complexes is set to $C_{max} = 10^{-5}$M, and the maximum strand displacement rate constant is $q_{max} = 10^6$ M^{-1} s^{-1}. For all of the reactions the rate constant is considered as $10^5 M^{-1} S^{-1}$. Figure 20 shows the ODE simulation results for the DNA implementation of the complete system illustrated in Figure 17 for different inputs.

VI. Comparison between Molecular and Electronic Circuits

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*). One of the great successes of electronic 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

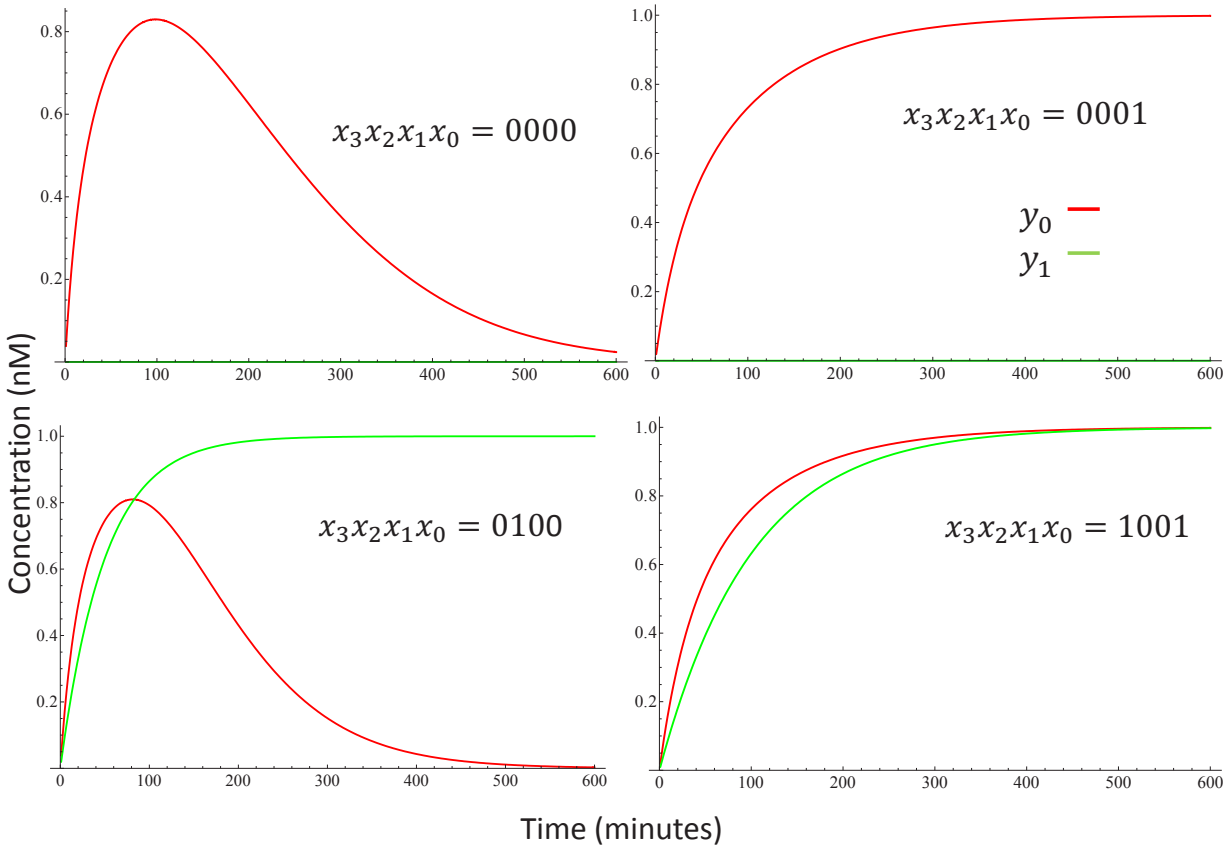


Fig. 16: Kinetics simulations that compute the Square-root of 0, 1, 4, and 9 using the molecular implementation of unit shown in Figure 15.

as SPICE [45]. However, the design of circuits occurs at more abstract levels – in terms of switches, gates, and modules. Research in molecular computation could benefit from this hierarchical approach.

We point out several fundamental differences in characteristics of molecular and electronic circuits. These are summarized in Table III. Fanout operations in electronic circuits are free while these are expensive in molecular implementations. Addition operations are free in molecular systems, but are expensive in electronic circuits. The critical path of an electronic circuit is typically bounded by computation time; the delay elements enable reduction of critical path and faster computation. However, molecular implementations of delay elements require inherently slow transfer reactions; the speed of molecular systems is bounded by the communication bound as opposed to the computation bound. The computations in molecular systems are inherently highly parallel unlike in electronic systems where parallelism requires significant increase in hardware resources. Finally the electronic circuits are highly integrated while the molecular systems are not suitable for highly integrated implementations. DNA and electronic systems also differ fundamentally with respect to storage properties. DNA systems can hold their concentrations indefinitely while the charge or stored value in an electronic system can leak and needs to be refreshed periodically.

VII. Discussion and Concluding Remarks

This paper presented methodologies for implementing continuous-time, discrete-time and digital processing with molecular reactions. Several examples are presented to illustrate the approaches presented in the paper.

Although pertaining to biology, the contributions of this paper are neither experimental nor empirical; rather they are constructive and conceptual. We design robust digital logic with molecular reactions. For the molecular digital systems, 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.

Intense efforts by the synthetic biology community have been devoted to the implementation of computation and logical functions with genetic regulatory elements [46]-[50]. For example design of robust logical circuits using chemically wired cells have been presented in [41] for single logic gates. Also genetic circuits consisting of multi-layer logical gates have been implemented in single cell in [51]. Yet, progress seems to have stalled at the complexity level of circuits with perhaps 7-15 components. In fact, *in vivo* engineering of such circuits is full of experimental difficulties. In contrast, *in vitro* molecular computation with DNA strand displacement is following a Moore's Law-like trajectory in the scaling of its complexity. Thus, due to their complexity, systems presented in this paper are more likely to be physically realizable *in vitro* than *in vivo*.

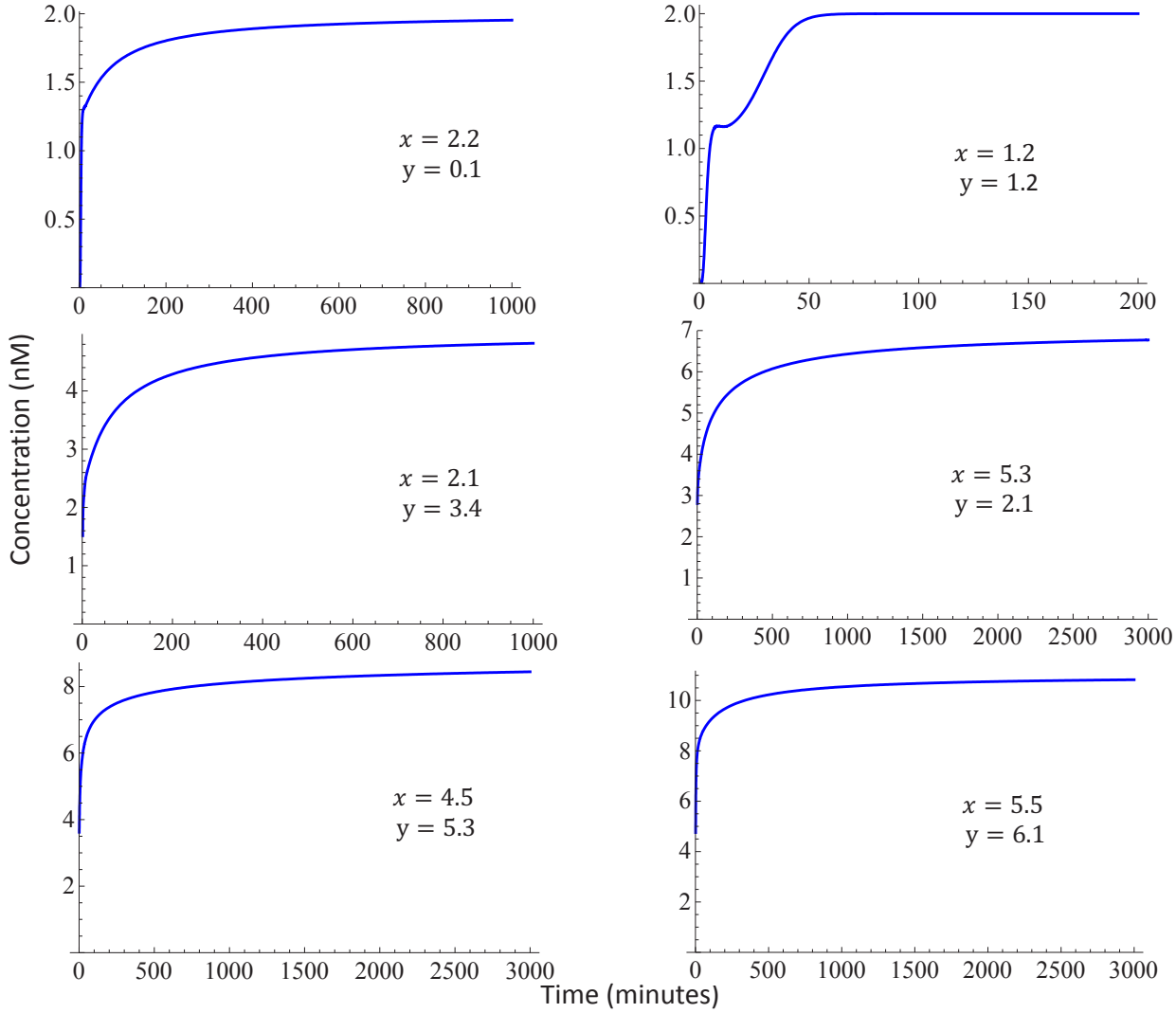


Fig. 18: Simulation results for the system shown in Figure 17.

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 [52], [53], [54], [55].

One should notice that there is quantization error in the ADC component. This is similar to the quantization error for other types of ADC usually used in digital signal processing systems [56]. The error can decrease the accuracy of system. The quantization error can be reduced by increasing the ADC resolution and, consequently, increasing the number of bits of ADC and DAC components.

In future work, we will perform more detailed studies of the characteristics of biomolecular continuous-time, discrete-time and digital processing systems including noise analysis. For instance, we will study how the resolution correlates with changing molecular concentrations and how robust the designs are to parametric variations. Also, we will develop faster implementations. The main bottleneck in current implementations has been speed. Unlike in electronic systems, where the speed is limited by changes in electric charge, the speed in molecular systems is limited by changes in molecular concentrations, which are inherently slow.

We will investigate new scheduling approaches where multiple computations are mapped into different phases of transfer. Reducing currently achievable sample periods from hundreds of hours to a few hours, or even a few minutes, will enable experimental demonstration of some example signal processing functions using DNA.

References

[1] R. H. Carlson, *Biology Is Technology: The Promise, Peril, and New Business of Engineering Life*. Cambridge, MA: Harvard UP, 2010.

[2] R. Carlson, "The pace and proliferation of biological technologies," *Biosecurity and Bioterrorism: Biodefense Strategy, Practice, and Science*, vol. 1, pp. 203–214, Sep. 2003.

[3] M. Conrad, "Molecular computing," *Advances in Computers*, vol. 31, pp. 235–324, 1990.

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

[5] J. Chen and D. H. Wood, "Computation with biomolecules," *Proceedings of the National Academy of Sciences*, vol. 97, no. 4, pp. 1328–1330, 2000.

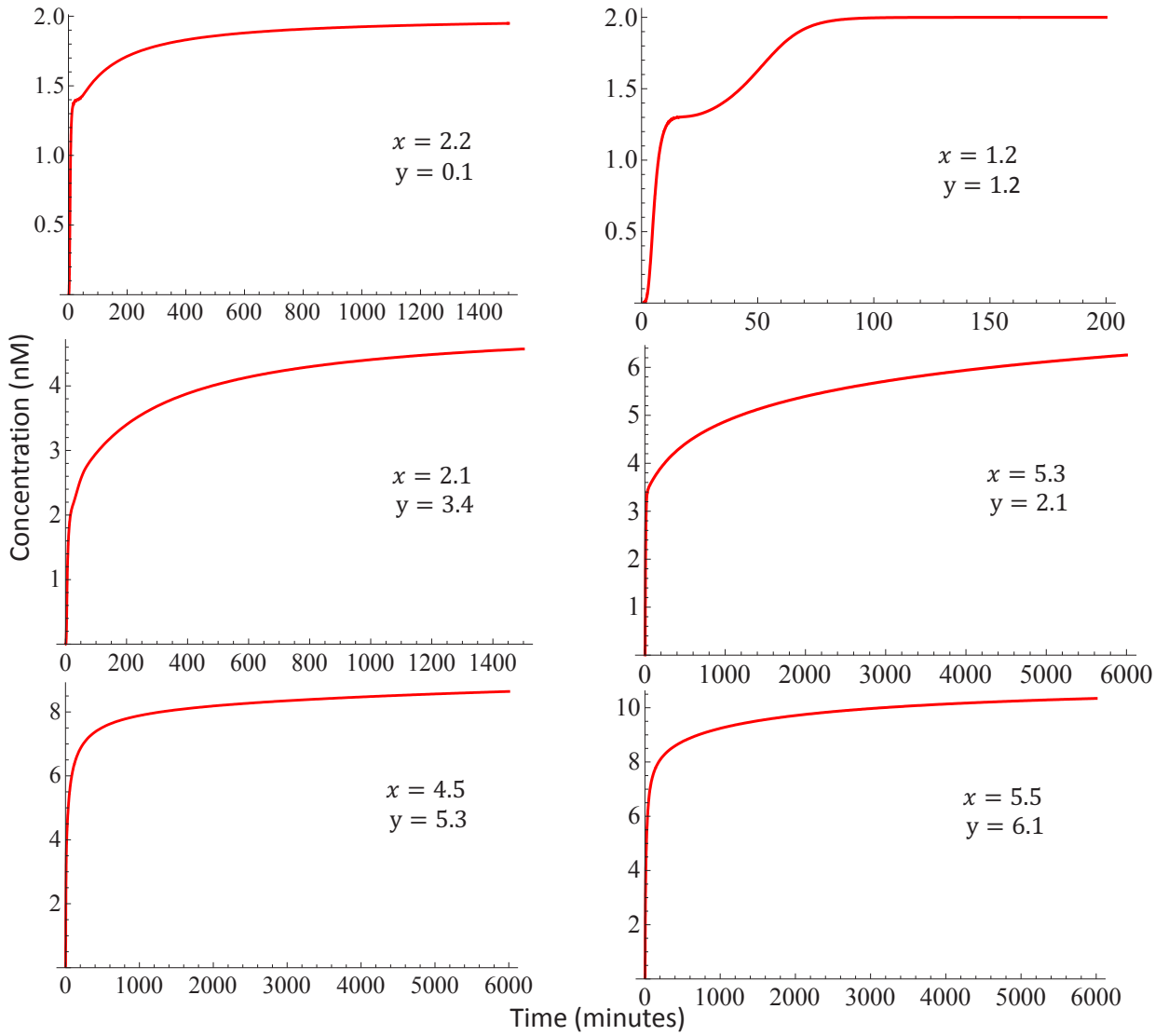


Fig. 20: Simulation results for the DNA implementation of the system shown in Figure 17.

[6] Y. Benenson, "Biocomputers: from test tubes to live cells," *Mol. Biosyst.*, vol. 5, pp. 675–685, 2009.

[7] N. Nandagopal and M. B. Elowitz, "Synthetic biology: integrated gene circuits," *Science*, vol. 333, pp. 1244–1248, 2011.

[8] A. J. Hockenberry and M. C. Jewett, "Synthetic in vitro circuits," *Science*, vol. 16, pp. 253–259, 2012.

[9] A. J. Genot, T. Fujii, and Y. Rondelez, "Scaling down dna circuits with comptetitive neural networks," *J. R. Soc. Interface*, vol. 10.

[10] S. A. Salehi, M. D. Riedel, and K. K. Parhi, "Markov chain computations using moleular reactions," *IEEE International Conference on Digital Signal Processing (DSP)*, 2015.

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

[12] M. N. Stojanovic and D. Stefanovic, "A deoxyribozyme-based molecular automaton," *Nat. Biotechnol*, vol. 21, pp. 1069–1074, 2003.

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

[14] T. Ran, S. Kaplan, and E. Shapiro, "Molecular implementation of simple logic programs," *Nat. Nanotechnol*, vol. 4, pp. 642–648, 2009.

[15] Y. Benenson, "Biomolecular computing systems: principles, progress and potential," *Nature Reviews Genetics*, vol. 13, pp. 455–468, 2012.

[16] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*. 5th ed. Garland Science, USA, 2007.

[17] P. Erdi and J. Toth, *Mathematical Models of Chemical Reactions: Theory and Applications of Deterministic and Stochastic Models*. Manchester University Press, 1989.

[18] F. Horn and R. Jackson, "General mass action kinetics," in *Archive for Rational Mechanics and Analysis*, no. 47, 1972, pp. 81–116.

[19] W. W. Chen, M. Niepel, and P. K. Sorger, "Classic and contemporary approaches to modeling biochemical reactions," *PMC Genes Dev.*, vol. 24, no. 17, pp. 1861–1875, 2010.

[20] K. Rinaudo, L. Bleris, R. Maddamsetti, S. Subramanian, R. Weiss, and Y. Benenson, "A universal rnai-based logic evaluator that operates in mammalian cells," *Nature biotechnology*, vol. 25, no. 7, pp. 795–801, 2007.

[21] Z. Xie, L. Wroblewska, L. Prochazka, R. Weiss, and Y. Benenson, "Multi-input rnai-based logic circuit for identification of specific cancer cells," *Science*, vol. 333, no. 6047, pp. 1307–1311, 2011.

[22] A. Reynolds, D. Leake, Q. Boese, S. Scaringe, W. S. Marshall, and A. Khvorova, "Rational sirna design for rna interference," *Nature biotechnology*, vol. 22, no. 3, pp. 326–330, 2004.

[23] 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.

[24] 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.

[25] K. Oishi and E. Klavins, "Biomolecular implementation of linear i/o systems," *IET Syst. Biol.*, vol. 5, pp. 252–260, 2011.

[26] H. Jiang, S. Salehi, M. Riedel, and K. Parhi, "Discrete-time signal

TABLE III: Comparison between molecular (DNA) and electronics (silicon) computing systems.

Aspect	**DNA-based**			**Silicon-based**
	analog	Discrete-time	digital	
Addition	free	free	expensive	expensive
Multiplication	less expensive	less expensive	expensive	expensive
Fanout	expensive	expensive	free	free
Delay/Signal Transfer	very expensive			almost free
Bound on Performance	communication bounded			computation bounded
Speed	ultra-slow			very fast
Area	$\sim$ nm			$\sim$ nm
Parallelism	highly-parallel			less-parallel
Level of Integration	no integration			highly-integrated
Application Area	*in vivo/in vitro*			industrial/consumer

processing with DNA," *ACS Synthetic Biology*, vol. 2, no. 5, pp. 245–254, 2013.

[27] H. Jiang, M. Riedel, and K. Parhi, "Synchronous sequential computation with molecular reactions," *Proc. of 2011 ACM/IEEE Design Automation Conference*, pp. 836–841, 2011.

[28] ——, "Digital signal processing with molecular reactions," *IEEE Design and Test Magazine, (Special Issue on Bio-Design Automation in Synthetic Biology)*, vol. 29, no. 3, pp. 21–31, 2012.

[29] ——, "Asynchronous computations with molecular reactions," *Asilomar Conf. on Signals, Systems and Computers*, pp. 493–497, 2011.

[30] S. A. Salehi, M. D. Riedel, and K. K. Parhi, "Asynchronous discrete-time signal processing with molecular reactions," *Proc. of Asilomar Conference on Signals, Systems, and Computers*, pp. 1767–1772, 2014.

[31] H. M. Sauro and K. Kim, "Synthetic biology: It's an analog world," *Nature*, vol. 497, pp. 572–573, 2013.

[32] R. Sarpeshkar, "Analog synthetic biology," *Philos. Transact. A. Math. Phys. Eng. Sci.*, vol. 372, no. 20130110, 2014.

[33] ——, "Analog versus digital: extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, pp. 1601–1638, 1998.

[34] R. Daniel, J. R. Rubens, R. Sarpeshkar, and T. K. Lu, "Synthetic analog computation in living cells," *Nature*, vol. 497, pp. 619–623, 2013.

[35] S. Hayat and T. Hinze, "Toward integration of in vivo molecular computing devices: successes and challenges," *HFSP Journal*, vol. 5, no. 2, pp. 239–243, 2008.

[36] T. Kailath, *Linear Systems*. Prentice Hall, Englewood Cliffs, N.J., 1980.

[37] V. V. Kulkarni, H. Jiang, T. Chanyaswad, and M. Riedel, "A biomolecular implementation of systems described by linear and nonlinear ODE's," in *International Workshop on Biodesign Automation*, 2013.

[38] K. K. Parhi, *VLSI Digital Signal Processing Systems*. John Wiley & Sons, 1999.

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

[40] P. D. Kepper, I. R. Epstein, and K. Kustin, "A systematically designed homogeneous oscillating reaction: the arsenite-iodate-chlorite system," *Journal of the American Chemical Society*, vol. 103, no. 8, pp. 2133–2134, 2008.

[41] A. Tamsir, J. J. Tabor, and C. A. Voigt, "Robust multicellular computing using genetically encoded nor gates and chemical 'wires'," *Nature*, vol. 469, pp. 212–215, 2011.

[42] L. Qian and E. Winfree, "Scaling up digital circuit computation with dna strand displacement cascades," *Science*, vol. 332, pp. 1196–1201, 2011.

[43] H. Jiang, M. D. Riedel, and K. K. Parhi, "Digital logic with molecular reactions," in *IEEE International Conference on Computer Aided Design (ICCAD)*. IEEE, 2013, pp. 721–727.

[44] P. Siuti, J. Yazbek, and T. Lu, "Synthetic circuits integrating logic and memory in living cells," *Nature Biotechnology*, vol. 31, pp. 448–?452, 2013.

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

[46] R. Weiss, S. Basu, S. Hooshangi, A. Kalmbach, D. Karig, R. Mehreja, and I. Netravali, "Genetic circuit building blocks for cellular computation, communications, and signal processing," *Natural Computing*, vol. 2, pp. 47–84, 2003.

[47] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in escherichia coli," *Nature*, vol. 403, no. 2000, pp. 339–342, 2000.

[48] 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.

[49] D. Endy, "Foundations for engineering biology," *Nature*, vol. 438, pp. 449–453, 2005.

[50] 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.

[51] T. Moon, C. Lou, A. Tamsi, B. Stanton, and C. A. Voigt, "Genetic programs constructed from layered logic gates in single cells," *Nature*, vol. 491, pp. 249–253, 2012.

[52] 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.

[53] 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.

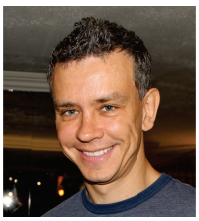
[54] S. Venkataramana, R. M. Dirks, C. T. Ueda, and N. A. Pierce, "Selective cell death mediated by small conditional RNAs," *Proceedings of the National Academy of Sciences*, vol. 107, no. 39, pp. 16 777–16 782, 2010.

[55] 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.

[56] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. 3rd ed. Prentice Hall Press Upper Saddle River, NJ, USA, 2009.

Sayed Ahmad Salehi is Ph.D. candidate in Electrical and Computer Engineering at the University of Minnesota. He received his M.Sc. in Electrical Engineering from the Isfahan University of Technology focusing on VLSI architectures for digital signal processing (DSP). He then was at industry for 8 years working on embedded hardwares for DSP. From 2007 to 2011 he was also a lecturer in DSP at some Iranian universities. He received another M.Sc. from the Department of Electrical and Computer Engineering at the University of Minnesota working on a new topic: signal processing by molecular reactions and biological computations. His paper "Markov Chain Computations using Molecular Reactions" was nominated for the best paper award at IEEE International Conference on DSP 2015. His current research area includes biomolecular computations (particularly signal processing by DNA), systems and synthetic biology.

Hua Jiang is a Software Engineer with the Data Infrastructure Group at LinkedIn Corporation, California. He received his B.Eng. degree at the University of Electronic Science and Technology of China in 2004, M.Eng. degree at Shanghai Jiao Tong University, China, in 2007, and Ph.D. degree at the University of Minnesota, Twin Cities, in 2012, all in Electrical Engineering. He was with the Design Group of Synopsys Inc. His research interests include novel computational paradigms, logic synthesis, and distributed computing.



Marc Riedel is Associate Professor of Electrical and Computer Engineering at the University of Minnesota. From 2006 to 2011 he was Assistant Professor. He is also a member of the Graduate Faculty in Biomedical Informatics and Computational Biology. From 2004 to 2005, he was a lecturer in Computation and Neural Systems at Caltech. He has held positions at Marconi Canada, CAE Electronics, Toshiba, and Fujitsu Research Labs. He received his Ph.D. and his M.Sc. in Electrical Engineering at Caltech and his B.Eng. in Electrical Engineering with a Minor in Mathematics at McGill University. His Ph.D. dissertation titled "Cyclic Combinational Circuits" received the Charles H. Wilts Prize for the best doctoral research in Electrical Engineering at Caltech. His paper "The Synthesis of Cyclic Combinational Circuits" received the Best Paper Award at the Design Automation Conference. He is a recipient of the NSF CAREER Award.

Keshab K. Parhi (S'85-M'88-SM'91-F'96) received the B.Tech. degree from IIT Kharagpur, Kharagpur, India, in 1982, the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, PA, USA, in 1984, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 1988.

He has been with the University of Minnesota, Minneapolis, MN, USA, since 1988, where he is currently a Distinguished Mc Knight University Professor and an Edgar F. Johnson Professor with the Department of Electrical and Computer Engineering. He has published over 575 papers, is the inventor or co-inventor of 29 patents, has authored the textbook VLSI Digital Signal Processing Systems (New York, NY, USA: Wiley, 1999), and co-edited the reference book Digital Signal Processing for Multimedia Systems (Boca Raton, FL, USA: CRC Press, 1999). His current research interests include the VLSI architecture design and implementation of signal processing, communications and biomedical systems, error control coders and cryptography architectures, high-speed transceivers, stochastic computing, secure computing, and molecular computing. He is also currently working on intelligent classification of biomedical signals and images, for applications such as seizure prediction and detection, schizophrenia classification, biomarkers for mental disorder, brain connectivity, and diabetic retinopathy screening.

Dr. Parhi has served on the Editorial Boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I AND PART II, VLSI Systems, Signal Processing, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE SIGNAL PROCESSING MAGAZINE, and served as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I from 2004 to 2005. He currently serves on the Editorial Board of the Springer Journal of Signal Processing Systems. He has served as the Technical Program Co-Chair of the 1995 IEEE VLSI Signal Processing Workshop and the 1996 Application Specific Systems, Architectures, and Processors Conference, and as the General Chair of the 2002 IEEE Workshop on Signal Processing Systems. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society from 1996 to 1998. He served as a Board of Governors Elected Member of the IEEE Circuits and Systems Society from 2005 to 2007. He is a recipient of numerous awards, including the 2013 Distinguished Alumnus Award from IIT Kharagpur, the 2013 Graduate/Professional Teaching Award from the University of Minnesota, the 2012 Charles A. Desoer Technical Achievement Award from the IEEE Circuits and Systems Society, the 2004 F. E. Terman Award from the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, the 2001 IEEE W. R. G. Baker Prize Paper Award, and the Golden Jubilee Medal from the IEEE Circuits and Systems Society in 2000.