Asynchronous Computation with Molecular Reactions

Hua Jiang, 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, mriedel, parhi}@umn.edu

***Abstract*—Mass-action kinetics of chemical reaction networks (CRNs) is powerful to describe computations through transfers of chemical concentrations. Here we present methods for asynchronously implementing digital signal processing (DSP) operations such as filtering with CRNs. We first review an implementation of DSP operations using molecular reactions based on a three-phase transfer scheme. This is an example of a locally asynchronous, globally synchronous implementation. We then present a fully asynchronous method that transfers signals based on absence of other signals. We illustrate our methodology with the design of finite impulse response (FIR) filters. The computation is exact and independent of specific reaction rates. Although conceptual for the time being, the proposed methodology has potential applications in domains of synthetic biology such as biochemical sensing and drug delivery.**

I. Introduction

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]. Due to its abstract representation, chemical reaction networks (CRNs) can describe not only existing chemical processes, but it can also be used as a programming language to model computational networks.

Mass-action kinetics of CRNs is powerful to describe computations through transfers of chemical concentrations [7], [8]. Until recently, however, realization of such chemical systems has been difficult because it is almost impossible to map every reaction in a CRN to feasible chemical reactions. It is reported that kinetics of an arbitrary CRN can be implemented by DNA strand displacement reactions [5], [6], as long as all reactions in the network are either unimolecular or bimolecular reactions.

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*). In this paper, we apply and extend expertise from digital signal processing (DSP), a sophisticated, mature domain [9], to the domain of synthetic biology.

This work is supported by NSF grant #CCF-1117168.

A. 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. The value k is called the *kinetic constant*. We model the molecular dynamics in terms of *mass-action kinetics* [10], [11]: reaction rates are proportional to the quantities of the participating molecular types and 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). Most prior schemes for molecular computation depend on specific values of the kinetic constants: the formulas that they compute include the k's. This limits the applicability since the kinetic constants are not constant at all; they depend on factors such as cell volume and temperature. So the results of the computation are not robust.

We aim for robust constructs: in our methodology we use only coarse rate categories ("fast" and "slow"). Given such categories, the computation is exact and independent of the specific reaction rates. In particular, it does not matter how fast any "fast" reaction is relative to another, or how slow any "slow" reaction is relative to another – only that "fast" reactions are fast relative to "slow" reactions.

B. Organization

The rest of the paper is organized as follows. First, in Section II, we provide background for implementing DSP systems with molecular reactions. Next, in Section III, we review a locally asynchronous, globally synchronous implementation. Then in Section IV, we propose a new fully asynchronous implementation. Finally, in Section V, we provide concluding remarks.

II. Preliminaries

DSP systems consist of delay elements and computational elements. This section discusses the abstraction of data flow,

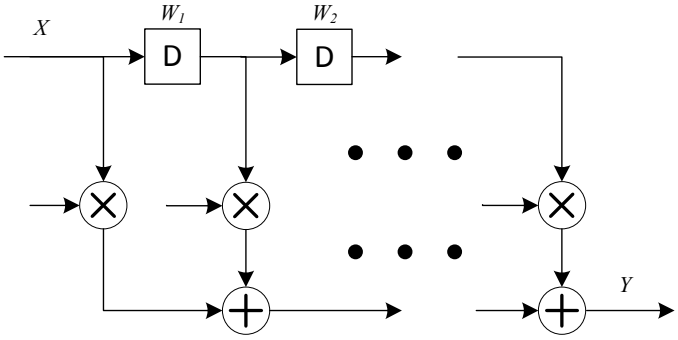


Fig. 1: Block diagram of a general FIR filter.

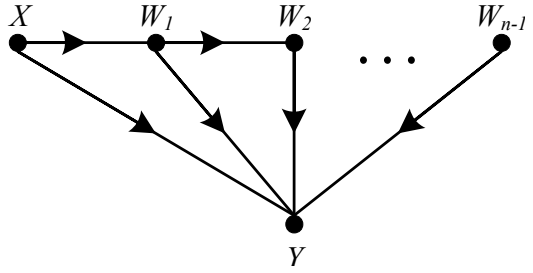


Fig. 2: DFG of an n-tap FIR filter.

which is determined by delay elements. Implementation of computational elements, e.g., scalar multiplier and adder, are addressed later.

A. Signal Transfer Model

DSP systems are intrinsically represented by data flows. Block diagram of a general n-tap finite impulse response (FIR) filter is shown in Fig. 1. Fig. 2 represents a data flow graph (DFG) derived from the block diagram. Each delay element in Fig. 1 corresponds to a node in the DFG in Fig. 2. The communicating edges in Fig. 2 represent computation paths. The input node is marked as node X and the output node is marked as node Y. The nodes X and Y may be interpreted as input and output flip-flops.

To implement signal transfers with molecular reactions, each node is assigned to a different molecular type. The molecular transfer from source type to destination type corresponding to an edge implements the signal transfer. For example, reaction

$$W_1 \xrightarrow{\text{slow}} W_2 \tag{3}$$

implements the transfer from W_1 to W_2.

When the transfer is complete, the concentration of the destination molecular type has been increased by previous concentration of source type. There are no molecules of the source type; the source node has been emptied and is ready to take new signal values.

B. Computational Elements

Scalar multiplication

$$y = \frac{c_2}{c_1} x$$

is implemented by reaction

$$c_1 X \xrightarrow{\text{fast}} c_2 Y, \tag{4}$$

where c_1 and c_2 are constants.

Every time this reaction fires, c_1 molecules of X get transfered to c_2 molecules of Y. Once the reaction has fired to completion, i.e., fully consumed all molecules of X, the requisite operation of scalar multiplication is complete.

Addition operation

$$y = x_1 + x_2$$

is implemented by choosing several reactions with the same product:

$$\begin{array}{rcl} X_1 & \longrightarrow & Y \\ X_2 & \longrightarrow & Y. \end{array} \tag{5}$$

Once both reactions have fired to completion, the concentration of Y will be the former concentration of X_1 plus the former concentration of X_2.

The fanout operation duplicates concentrations. It is implemented by choosing a reaction producing several different products from a single reactant:

$$X \xrightarrow{\text{fast}} Y_1 + Y_2. \tag{6}$$

Once this reaction has fired to completion, both the concentration of Y_1 and the concentration of Y_2 will be equal to the former concentration of X.

In our design, reactions of computational elements are fast and reactions of signal transfer are slow, so that computational reactions do not affect signal transfer.

C. Absence Indication

Consider that we have a group of molecular types $S_1, S_2, \cdots, S_n$. a_S of this group is a molecular type such that when molecules of any type in this group are present, concentration of a_S is nearly zero; otherwise, when concentrations of S_i $(i = 1, \cdots, n)$ are all zero, concentration of a_S is nonzero. Therefore, a_S is called absence indicator of S_i [12]. This is satisfied by the reactions below:

$$\begin{array}{rcl} \varnothing & \xrightarrow{\text{slow}} & a_S \\ S_1 + a_S & \xrightarrow{\text{fast}} & S_1 \\ S_2 + a_S & \xrightarrow{\text{fast}} & S_2 \\ & \vdots & \\ S_n + a_S & \xrightarrow{\text{fast}} & S_n. \end{array} \tag{7}$$

Here the symbol $\varnothing$ indicates "no reactants" meaning that the products are generated from a large or replenishable source. The first reaction slowly but continually generates molecules of a_S. In the following reactions, $S_1, S_2, \cdots, S_n$ quickly consume a_S, but keep their own concentrations. Therefore, molecules of a_S accumulate only when all types in $S_1, S_2, \cdots, S_n$ are absent. Absence indication is used to control signal transfer, as discussed in next two sections.

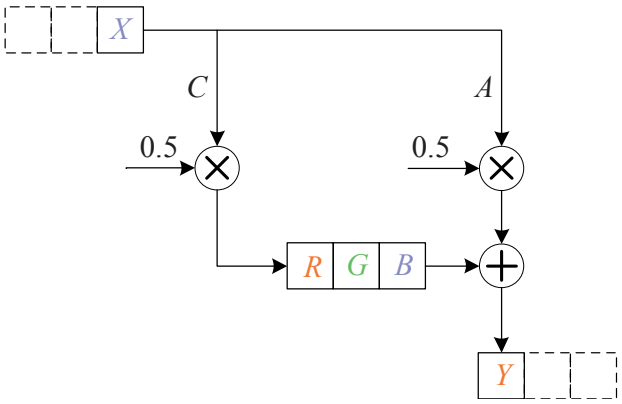


Fig. 3: Two-tap moving average filter in a three-phase configuration.

III. Globally Synchronous, Locally Asynchronous Implementation

We have presented a method to implement DSP systems with molecular reactions using a three-phase signal transfer scheme (referred to as the RGB scheme) [12]. In each computation cycle, or a single iteration of the DFG, signal transfers among delay elements are synchronized globally by the RGB scheme.

We implement delay elements by transferring concentrations between molecular types based on the absence of other types. Each delay element DE_i is assigned three molecular types $R(ed)_i$, $G(reen)_i$ and $B(lue)_i$. This is a 3-slow implementation, in which each delay is replaced by 3 delays (see [9], p. 120). Signal going through DE_i will be first transferred to R_i, and then G_i and B_i. We consider system input X as blue and system output Y as red. The molecular types of a two-tap moving-average filter are labeled in Fig. 3.

A computation cycle, in which an input value is processed and an output value is computed, completes in three phases. In each phase the signals are transfered from molecular types in one color category to the next. Transfers of molecules from one color to another are enabled by absence of the third color. For example, $R_i \longrightarrow G_i$ fires only when a_B, absence indicator of all blue molecules, present. Therefore, transfers among color categories of different delay elements are globally synchronized.

The transfers are implemented by the following reactions.

Phase 1 reactions:

$$\begin{aligned} a_B + R_i &\xrightarrow{\text{slow}} G_i \\ 2G_i &\xrightarrow{\text{slow}} I_{G,i} \\ I_{G,i} &\xrightarrow{\text{fast}} 2G_i \\ I_{G,i} + R_i &\xrightarrow{\text{fast}} 3G_i \\ Y &\xrightarrow{\text{collect}} \varnothing \end{aligned} \tag{8}$$

In the first reaction in Reactions (8), transfer of R_i to G_i is enabled by a_B. The next three reactions provide positive feedback kinetics. These reactions effectively speed up transfers between color categories as molecules in one category are "pulled" to the next. Since Y is considered red, it is collected in Phase 1.

Similarly, reactions of Phase 2 and 3 are

Phase 2 reactions:

$$\begin{aligned} a_R + G_i &\xrightarrow{\text{slow}} B_i \\ 2B_i &\xrightarrow{\text{slow}} I_{B,i} \\ I_{B,i} &\xrightarrow{\text{fast}} 2B_i \\ I_{B,i} + G_i &\xrightarrow{\text{fast}} 3B_i \\ \varnothing &\xrightarrow{\text{inject}} X \end{aligned} \tag{9}$$

Phase 3 reactions:

$$\begin{aligned} a_G + B_i &\xrightarrow{\text{slow}} Computations \\ 2R_j &\xrightarrow{\text{slow}} I_{R,j} \\ I_{R,j} &\xrightarrow{\text{fast}} 2R_j \\ I_{R,j} + B_i &\xrightarrow{\text{fast}} Computations + 2R_j \\ Computations &\xrightarrow{\text{fast}} R_j. \end{aligned} \tag{10}$$

a_R is the absence indicator of R_i's and Y; a_G is the absence indicator of G_i's; a_B is the absence indicator of B_i's and X. These are implemented by reactions similar to Reactions (7). Computations are carried out in Phase 3, during the transfer from blue to red. The computation reactions fire much faster than the transfer reactions. Therefore, R_j molecules are immediately produced from B_i molecules. Note that R_j molecules produced in Phase 3 will be a red type of any succeeding delay element DE_j along the signal path from DE_i.

With the synthesis method discussed above, the full set of reactions for the moving average filter is listed below:

$$\begin{aligned} a_G + X &\xrightarrow{\text{slow}} A + C \\ 2C &\xrightarrow{\text{fast}} R \\ 2A &\xrightarrow{\text{fast}} Y \\ a_B + R &\xrightarrow{\text{slow}} G \\ a_R + G &\xrightarrow{\text{slow}} B \\ a_G + B &\xrightarrow{\text{slow}} Y, \end{aligned} \tag{11}$$

$$\begin{aligned} 2R &\xrightarrow{\text{slow}} I_R \\ I_R &\xrightarrow{\text{fast}} 2R \\ I_R + X &\xrightarrow{\text{fast}} A + C + 2R \\ 2G &\xrightarrow{\text{slow}} I_G \\ I_G &\xrightarrow{\text{fast}} 2G \\ I_G + R &\xrightarrow{\text{fast}} 3G \\ 2B &\xrightarrow{\text{slow}} I_B \\ I_B &\xrightarrow{\text{fast}} 2B \\ I_B + G &\xrightarrow{\text{fast}} 3B \\ 2Y &\xrightarrow{\text{slow}} I_Y \\ I_Y &\xrightarrow{\text{fast}} 2Y \\ I_Y + B &\xrightarrow{\text{fast}} 3Y, \end{aligned} \tag{12}$$

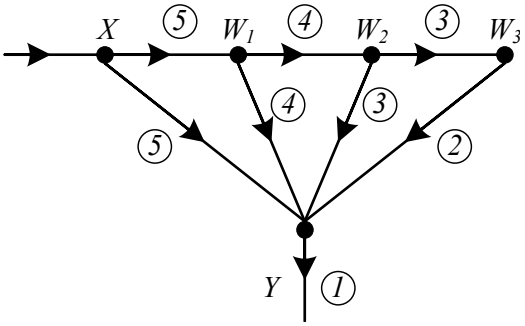


Fig. 4: DFG of a 4-tap FIR filter with signal transfer sequence. Step numbers are in circles.

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} a_R \\ R + a_R &\xrightarrow{\text{fast}} R \\ Y + a_R &\xrightarrow{\text{fast}} Y \\ \varnothing &\xrightarrow{\text{slow}} a_G \\ G + a_G &\xrightarrow{\text{fast}} G \\ \varnothing &\xrightarrow{\text{slow}} a_B \\ B + a_B &\xrightarrow{\text{fast}} B \\ X + a_B &\xrightarrow{\text{fast}} X. \end{aligned} \tag{13}$$

IV. Fully Asynchronous System Implementation

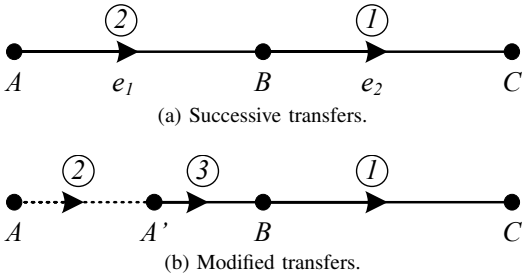
DSP systems can also be implemented fully asynchronously. In this section we present a new method for such implementation.

A. Signal Transfer Sequence

For a DSP system to be implemented fully asynchronously with molecular reactions, each directed edge of the DFG is assigned to a *step number*. The step number is similar to a scheduling step in software scheduling. The step number represents the step at which the signal transfer of an edge takes place. A key challenge is to assign these step numbers to the edges such that the assignment is *conflict-free* and requires the fewest possible steps. Scheduling such an assignment is a main contribution of this paper.

Transfer assigned to step i can fire only after all transfers assigned to step $i-1$ have been completed. Consider a 4-tap filter with all multiplier coefficients assigned to be 1. The DFG with assigned steps of the filter is shown in Fig. 4. Given a DFG with step assignment, every node is represented by a specific molecular type. Each directed edge is mapped to a reaction transferring its source type to its destination type. Proceeding from right to left, the signal transfers of the directed edges in Fig. 4 are assigned step numbers in increasing order. The input edge of a node is assigned to a step number that is 1 higher than its output.

We use absence indicator type a_i to represent the completion of step i. a_i is maintained by source nodes of all directed edges assigned to step i. For example, if transfers $src_1 \longrightarrow dest_1$ and $src_2 \longrightarrow dest_2$ are both assigned to step 1, then a_1 is



(a) Successive transfers.

(b) Modified transfers.

Fig. 5: An example of adding intermediate node to solve transfer conflict.

controlled by reactions

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} a_1 \\ src_1 + a_1 &\xrightarrow{\text{fast}} src_1 \\ src_2 + a_1 &\xrightarrow{\text{fast}} src_2. \end{aligned} \tag{14}$$

Molecules of a_1 accumulate only when both src_1 and src_2 are absent, which shows that step 1 has completed.

Given the absence indicators, signal transfers of each step are enabled by the absence indicator of its previous step, except for step 1. For any directed edge e_k assigned to step i, the signal transfer is enabled by a_{i-1}, implemented by:

$$a_{i-1} + src(e_k) \xrightarrow{\text{slow}} dest(e_k), \quad i > 1. \tag{15}$$

B. Conflict Elimination

Reactions (15) implement the signal transfers of all edges in the DFG. However, conflicts exist in this implementation. Suppose we have a signal path as shown in Fig. 5(a). There are two edges in this graph, e_1, assigned to step 2, and e_2, assigned to step 1. These are implemented by reactions:

$$\begin{aligned} B &\xrightarrow{\text{slow}} C \\ A + a_1 &\xrightarrow{\text{slow}} B \\ \varnothing &\xrightarrow{\text{slow}} a_1 \\ B + a_1 &\xrightarrow{\text{fast}} B. \end{aligned} \tag{16}$$

Here, node B is both the source of a transfer in step 1 and the destination of a transfer in step 2. When step 1 is complete, all molecules of B have been transferred to C and absence indicator a_1 starts to accumulate. With a_1, A starts to be transferred to B, which removes a_1 and further inhibits the transfer of edge e_1. Reaction $A + a_1 \overset{slow}{\rightarrow} B$ stops before A is fully transferred to B. This example shows that when a signal node is both the source of a transfer in step i and the destination of a transfer in step $i+1$, the system halts.

To resolve this problem, it is required that

$$step(e_1) \neq step(e_2) + 1 \tag{17}$$

when

$$dest(e_1) = src(e_2).$$

To achieve this, we first draw an equivalent 2-slow version of the DFG, where each delay is replaced by 2 delays (see [9], p. 120). Here we apply 2-slow configuration to A, which is replaced by A and A'. The edge from A to A' is represented

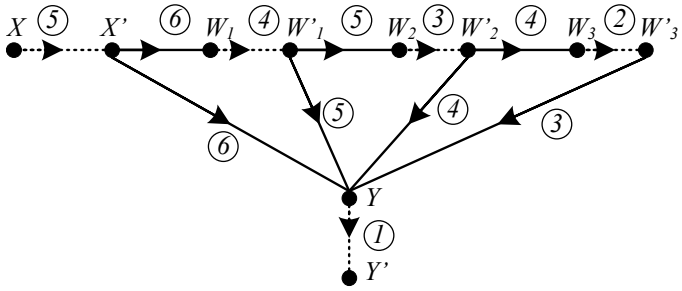


Fig. 6: Modified DFG without conflict.

by a dashed edge. Signal is first transferred from A to A' and then to B, before its further transfer to succeeding nodes. At the beginning of each computational cycle, there are no molecules of A'. The transfer of e_1 in Fig. 5(a) is finished in two steps. The first step, transfer from A to A', is still assigned to step 2; the second step, transfer from A' to B, is assigned to step 3, as shown in Fig. 5(b). Now there are no conflicting edges and absence indicators.

Formally, when a signal node is both the source of a transfer in step i and the destination of a transfer in step $i+1$, i.e., condition (17) does not hold, use the 2-slow implementation for the source node of the second transfer and break the second transfer into two steps, in step $i+1$ and $i+2$. Reactions managing absence indicators are re-generated in accordance with this change.

With conflict elimination, the DFG of the 4-tap filter is redrawn in Fig. 6. Fig. 6 represents a 2-slow version of Fig. 4; each delay is replaced by 2 delays: one marked with original variable, and another mark with a prime. The original delay and the primed delay nodes are connected by dashed edges. In the assignment in Fig. 6, sampling output Y, i.e., transfer of Y to Y', is assigned to step 1. In the proposed assignment, the signal transfers associated with dashed edges are numbered starting with 2 and in increasing order from right to left. Each outgoing edge of the destination nodes of the dashed edges is assigned a step that is 1 higher than the incoming edge. The filter is implemented by following reactions. Note that a_6 is used to enable step 1.

Signal transfer:

$$\begin{aligned} Y + a_6 &\xrightarrow{\text{slow}} Y' & \qquad W_1 + a_3 &\xrightarrow{\text{slow}} W_1' \\ W_3 + a_1 &\xrightarrow{\text{slow}} W_3' & \qquad W_2' + a_3 &\xrightarrow{\text{slow}} W_3 + Y \\ W_2 + a_2 &\xrightarrow{\text{slow}} W_2' & \qquad X + a_4 &\xrightarrow{\text{slow}} X' \\ W_3' + a_2 &\xrightarrow{\text{slow}} Y & \qquad W_1' + a_4 &\xrightarrow{\text{slow}} W_2 + Y \\ & & \qquad X' + a_5 &\xrightarrow{\text{slow}} W_1 + Y. \end{aligned}$$

Positive feedback kinetics:

$$\begin{aligned} W_3' + W_3 &\xrightarrow{\text{slow}} 2W_3' & \qquad W_3 + W_2' &\xrightarrow{\text{slow}} 2W_3 + Y \\ W_2' + W_2 &\xrightarrow{\text{slow}} 2W_2' & \qquad W_2 + W_1' &\xrightarrow{\text{slow}} 2W_2 + Y \\ W_1' + W_1 &\xrightarrow{\text{slow}} 2W_1' & \qquad W_1 + X &\xrightarrow{\text{slow}} 2W_1 + Y. \end{aligned}$$

Absence indicator:

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} a_1 & \qquad \varnothing &\xrightarrow{\text{slow}} a_4 \\ Y + a_1 &\xrightarrow{\text{fast}} Y & \qquad W_1 + a_4 &\xrightarrow{\text{fast}} W_1 \\ \varnothing &\xrightarrow{\text{slow}} a_2 & \qquad W_2' + a_4 &\xrightarrow{\text{fast}} W_2' \\ W_3 + a_2 &\xrightarrow{\text{fast}} W_3 & \qquad \varnothing &\xrightarrow{\text{slow}} a_5 \\ \varnothing &\xrightarrow{\text{slow}} a_3 & \qquad W_1' + a_5 &\xrightarrow{\text{fast}} W_1' \\ W_2 + a_3 &\xrightarrow{\text{fast}} W_2 & \qquad \varnothing &\xrightarrow{\text{slow}} a_6 \\ W_3' + a_3 &\xrightarrow{\text{fast}} W_3' & \qquad X' + a_6 &\xrightarrow{\text{fast}} X'. \end{aligned}$$

This example illustrates that the 4-tap FIR filter can be completed in 6 steps. In general, any n-tap FIR filter with all "1" coefficients can be completed by this assignment in $n+2$ steps.

V. Conclusions

This paper has introduced a new method for implementing data flow graph fully asynchronously by molecular reactions. Current work is being directed towards generalizing this method to construct a conflict-free assignment for any arbitrary DFG. Future work will be directed towards validating the proposed method by simulating the chemical reactions. We are exploring the mechanism of DNA-strand displacement advocated by Erik Winfree's group at Caltech as an experimental chassis [5]. They have shown that the kinetics of arbitrary chemical reactions can be emulated with DNA. Future work will also address mapping the proposed reactions to DNA strands displacement.

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

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

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

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

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

[12] 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.