A Synthesis Flow for Digital Signal Processing with Biomolecular Reactions

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

Department of Electrical and Computer Engineering
University of Minnesota
200 Union St. S.E., Minneapolis, MN 55455
{hua, veden002, mriedel, parhi}@umn.edu

***Abstract*—We present a methodology for implementing digital signal processing (DSP) operations such as filtering with biomolecular reactions. From a DSP specification, we demonstrate how to synthesize biomolecular reactions that produce time-varying output quantities of molecules as a function of time-varying input quantities. Unlike all previous schemes for biomolecular computation, ours produces designs that are dependent only on coarse rate categories for the reactions ("fast" and "slow"). Given such categories, the computation is exact and independent of the specific reaction rates. We implement DSP operations through a self-timed "handshaking" protocol that transfers quantities between molecular types based on the absence of other types. We illustrate our methodology with the design of a simple moving-average filter as well as a more complex biquad filter. We validate our designs through transient stochastic simulations of the chemical kinetics. Although conceptual for the time being, the proposed methodology has potential applications in domains of synthetic biology such as biochemical sensing and drug delivery. We are exploring DNA-based computation via strand displacement as a possible experimental chassis.**

I. Introduction

In the nascent field of synthetic biology, researchers are striving to create biological systems with functionality not seen in nature. Recent accomplishments portend of a coming revolution. From *Salmonella* that secretes spider silk proteins [1], to yeast that degrades biomass into ethanol [2], to *E. coli* that produces antimalarial drugs [3], the potential impacts are far-reaching.

The field has been driven by experimental expertise; much of its success has been attributable to the skill of the researchers in specific domains of biology. The electronic circuit design community has expertise to contribute to this endeavor [4], [5], [6], [7], [8]. In particular, digital signal processing (DSP) is a sophisticated, mature domain [9], [10]. In this paper, we apply and extend expertise from DSP to the domain of synthetic biology.

One of the great successes of integrated circuit design has been in abstracting and scaling the design problem. The physical behavior of transistors is understood in terms of differential equations – say, with models found in tools such as SPICE [11]. However, the design of circuits occurs at more abstract levels – in terms of switches, gates, and modules. Many analogous levels of abstraction exist for biological systems. These range from molecular dynamics, to protein networks, to genetic regulatory networks, to signaling pathways, to complete cellular systems, to multicellular organisms.

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

A. Computational Model

Here we will discuss a particular level of abstraction, analogous in some ways to transistor netlists: biomolecular reactions. A biomolecular reaction is a rule that specifies how types of molecules combine to produce other types. Consider the reaction

$$a + b \xrightarrow{\text{fast}} 2c.$$

When this reaction fires, one molecule of a is consumed, one of b is consumed, and two of c are produced. (Accordingly, a and b are called the *reactants* and c the *product*.)

We will examine the abstraction from a design perspective: how can we synthesize biomolecular reactions that produce specific output quantities of molecules as a function of input quantities? The challenge in setting up such computation is that the reactions execute asynchronously at different rates. The rate at which each reaction fires is proportional both to a rate constant and to the quantities of its reactants present. (The rate "constant" is not constant at all; it depends on factors such as temperature and volume.)

We aim to implement computation that is robust in the sense that it does not depend on the rates. We produce designs that only specify rates in qualitative terms, e.g., "fast" vs. "slow." (In our notation, such qualitative rates are listed above the arrows for reactions.) Given such coarse rate categories, the computation is exact and independent of the specific reaction rates. In particular, it doesn't 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.

The evolution of a biomolecular system over time can be characterized through stochastic simulation. First proposed by Gillespie, stochastic simulation has become the workhorse of computational biology [12], [13], [14] – the equivalent, one might say, of SPICE. Such simulation tracks integer quantities of the molecular species, executing reactions at random based on propensity calculations. Repeated trials are performed and

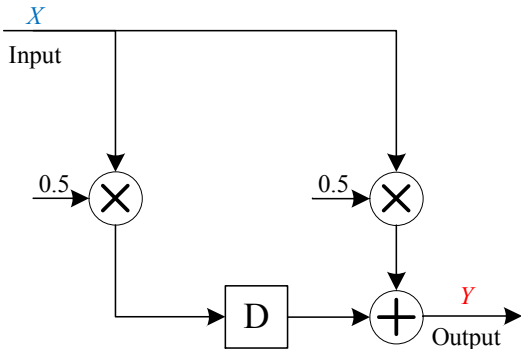


Fig. 1: A two-tap moving average filter.

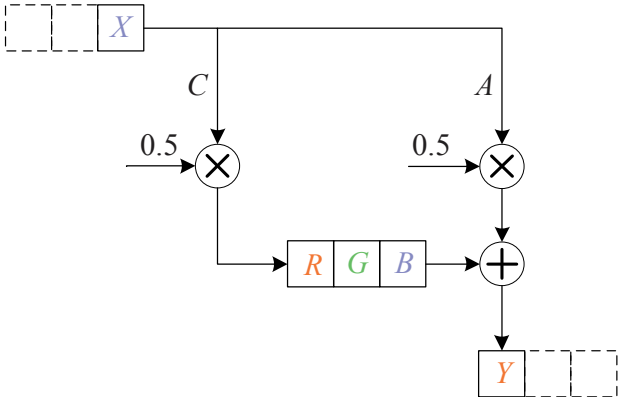


Fig. 2: Two-tap moving average filter in a three-phase configuration.

the probability distribution of different outcomes is estimated by averaging the results.

B. Organization

The paper is organized as follows. First, in Section II, we give a detailed example: we present a biomolecular implementation of a two-tap moving average filter. Then, in Section III, we present the general methodology for synthesizing DSP systems. To illustrate the general method, we provide a second, detailed example: a biomolecular implementation of a biquad filter. In Section IV, we present a general system synthesis flow. In Section V, we provide simulation results. Finally, in Section VI, we conclude with some remarks about potential applications for this work.

II. Example: A Moving-Average Filter

We illustrate our design methodology with a detailed example: a finite impulse response (FIR) filter. To elucidate the concepts, we present the design in simplified form first and then with some refinements.

A. Simplified Form

An FIR filter is shown in Figure 1. This system computes a *moving average*: given a time-varying input signal X, the output Y is a smoother version of it. More precisely, the output is one-half the current input value plus one-half the previous value. We implement a biomolecular moving-average filter with the following reactions.

$$\begin{aligned} g + X &\xrightarrow{\text{slow}} A + C \\ 2C &\xrightarrow{\text{fast}} R \\ 2A &\xrightarrow{\text{fast}} Y \end{aligned} \tag{1}$$

$$\begin{aligned} b + R &\xrightarrow{\text{slow}} G \\ r + G &\xrightarrow{\text{slow}} B \\ g + B &\xrightarrow{\text{slow}} Y \end{aligned} \tag{2}$$

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} r \\ R + r &\xrightarrow{\text{fast}} R \\ Y + r &\xrightarrow{\text{fast}} Y \\ \varnothing &\xrightarrow{\text{slow}} g \\ G + g &\xrightarrow{\text{fast}} G \\ \varnothing &\xrightarrow{\text{slow}} b \\ B + b &\xrightarrow{\text{fast}} B \\ X + b &\xrightarrow{\text{fast}} X \end{aligned} \tag{3}$$

Here the symbol $\varnothing$ indicates "no reactants" meaning the products are generated from a large or replenishable source. The molecular types are labeled in Figure 2. In the proposed scheme, there are three phases of computation. We color code the molecular types in corresponding color categories: Y and R in red; G in green; and X and B in blue.

In the group of reactions (1), the quantity of the input X is transfered to the same quantity of types A and C (a *fanout* operation). Then the quantities of A and C are reduced to half (*scalar multiplication* operations). Then the quantity of A is transfered to the output Y and the quantity of C is transfered to R, the first of three types of a *delay* operation. The next two are G and B. Once the signal has moved through the delay operation, the quantity of B is transfered to the output Y. (Since this quantity is combined with the quantity of Y produced from A, this is an *addition* operation.)

Within each delay operation, quantities are transfered from R to G, and then to B; this is accomplished by the group of reactions (2). 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. This handshaking ensures that the delay element takes a new value only when it has finished processing the previous value. It is implemented by the group of reactions (3). These continually generate the types r, g, and b that we call "*absence indicators*." These types only persist in the absence of the corresponding signals: r in the absence of R and Y; g in the absence of G; and b in the absence of X and B. They only persist in the absence because otherwise "fast" reactions consume them quickly.

Note that the quantity of the input X is sampled in the green-to-blue phase. We assume that an external source sup-

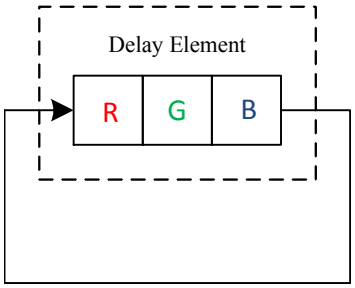


Fig. 3: RGB cycle in isolation.

plies the input. The output Y is produced in the blue-to-red phase. We assume that an external sink consumes these molecules.

B. Refinement

The essential aspect of the FIR design is that, within the RGB sequence for a delay operation, the full quantity of the preceding type is transfered to current type before the transfer to the succeeding type begins. For example, the reaction

$$r + G \xrightarrow{\text{slow}} B \tag{4}$$

should not fire until the reaction

$$b + R \xrightarrow{\text{slow}} G \tag{5}$$

has fired to completion.

However, the rate of a reaction is proportional to the quantities of its reactants. As molecules of R are transfered to G, the quantity of R decreases and so Reaction (5) slows down. With smaller quantities of R present, there will be larger quantities of the corresponding absence indicators r present. Meanwhile, the quantity of G increases so the rate of Reaction (4) increases. As a result, the transfer from G to B starts well before the transfer from R to G is complete. Similarly, the transfers from B to R and from R to G start earlier than they should. As a result, the iterative computation for the operation of our FIR filter fails.

To examine this issue, let us consider the RGB cycle in isolation, as illustrated in Figure 3. Suppose that this cycle is implemented with the following reactions:

$$\begin{aligned} b + R &\xrightarrow{\text{slow}} G \\ r + G &\xrightarrow{\text{slow}} B \\ g + B &\xrightarrow{\text{slow}} R \\ \varnothing &\xrightarrow{\text{slow}} r \\ \varnothing &\xrightarrow{\text{slow}} g \\ \varnothing &\xrightarrow{\text{slow}} b \\ R + r &\xrightarrow{\text{fast}} R \\ G + g &\xrightarrow{\text{fast}} G \\ B + b &\xrightarrow{\text{fast}} B. \end{aligned} \tag{6}$$

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, and B, but this oscillation is damped. This is confirmed by the experimental results in Figure 4. Here we simulated the chemical kinetics for the Reactions (6) [14]. In this figure, we see that the quantities oscillate with an attenuating envelope and converge to one third of the initial quantity of R.

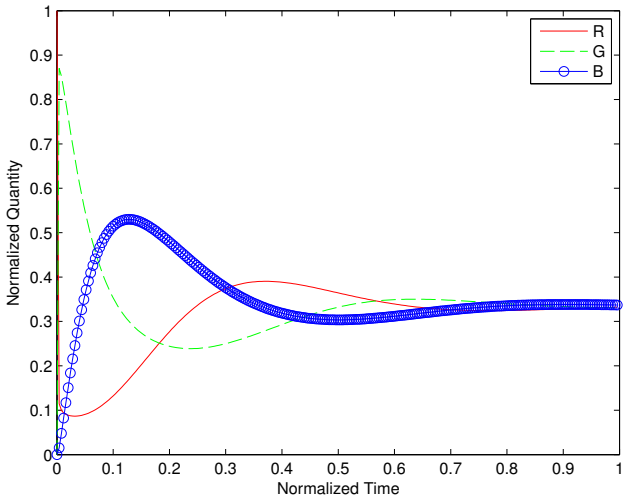


Fig. 4: Simulation of the Chemical Kinetics for the RGB Transfer Reactions (6).

A refinement to the RGB scheme solves this problem. We include reactions that accelerate and isolate the transfers in each phase. For the R to G phase, we add the reactions:

$$\begin{aligned} 2G &\xrightarrow{\text{slow}} I_G \\ I_G &\xrightarrow{\text{fast}} 2G \\ I_G + R &\xrightarrow{\text{fast}} 3G. \end{aligned} \tag{7}$$

In these 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 $2G \xrightarrow{\text{slow}} I_G$, the transfer will occur at a rate proportional to the *square* of the quantity of G.[1] Unlike Reaction (5), the rate of transferring R to G with Reactions (7) does not depend on the quantity of R; rather it has a quadratic dependence on the quantity of G, so the more G we have, the faster G is produced.

Symmetrically, we include the following reactions to transfer quantities from G to B

$$\begin{aligned} 2B &\xrightarrow{\text{slow}} I_B \\ I_B &\xrightarrow{\text{fast}} 2B \\ I_B + G &\xrightarrow{\text{fast}} 3B, \end{aligned} \tag{8}$$

[1] A rigorous discussion of chemical kinetics is beyond the scope of this paper. Interested readers can consult the references [12], [13], [15] and [16].

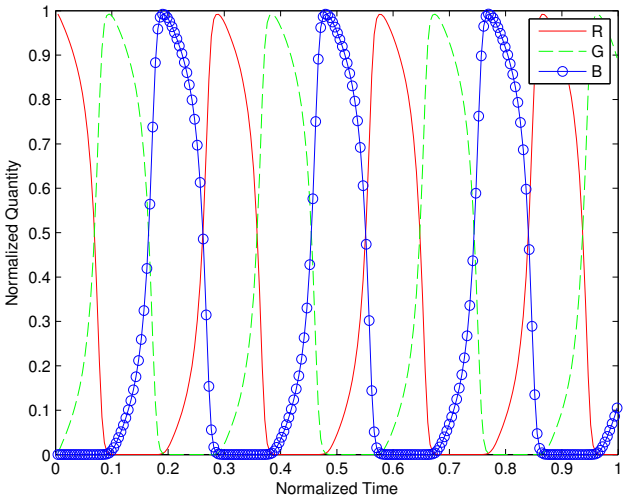


Fig. 5: Simulation of the Chemical Kinetics for the RGB transfers with the Reactions (7), (8), and (9).

and from B to R:

$$\begin{aligned} 2R &\xrightarrow{\text{slow}} I_R \\ I_R &\xrightarrow{\text{fast}} 2R \\ I_R + B &\xrightarrow{\text{fast}} 3R. \end{aligned} \tag{9}$$

With the reactions (7), (8), and (9), we get the oscillatory behavior that we need: these reactions effectively speed up transfers between color categories as molecules in each category are "pulled" to the next one. Figure 5 shows a simulation of the chemical kinetics. We see that the quantities of R, G, and B oscillate with constant amplitudes; there is no attenuation at all.

With these refinements, the full set of reactions for the moving average filter is:

$$\begin{aligned} 2R &\xrightarrow{\text{slow}} I_R \\ I_R &\xrightarrow{\text{fast}} 2R \\ I_R + X &\xrightarrow{\text{fast}} A + C + 2R \\ 2G &\xrightarrow{\text{slow}} I_G \\ I_G &\xrightarrow{\text{fast}} 2G \\ I_G + R &\xrightarrow{\text{fast}} 3G \\ 2B &\xrightarrow{\text{slow}} I_B \\ I_B &\xrightarrow{\text{fast}} 2B \\ I_B + G &\xrightarrow{\text{fast}} 3B \\ 2Y &\xrightarrow{\text{slow}} I_Y \\ I_Y &\xrightarrow{\text{fast}} 2Y \\ I_Y + B &\xrightarrow{\text{fast}} 3Y, \end{aligned} \tag{10}$$

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

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} r \\ R + r &\xrightarrow{\text{fast}} R \\ Y + r &\xrightarrow{\text{fast}} Y \\ \varnothing &\xrightarrow{\text{slow}} g \\ G + g &\xrightarrow{\text{fast}} G \\ \varnothing &\xrightarrow{\text{slow}} b \\ B + b &\xrightarrow{\text{fast}} B \\ X + b &\xrightarrow{\text{fast}} X. \end{aligned} \tag{12}$$

III. General DSP System Synthesis

DSP systems are generally specified in terms of four basic modules: *fanout*, *scalar multiplication*, *addition*, and *delay elements*. We describe biomolecular constructs for these four modules. (These constructs are all generalizations of the concepts illustrated with the moving-average filter in the previous section.) We illustrate the general design method with a second detailed example, a biquad filter.

A. Scalar Multiplier

Scalar multiplication performs the operation

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

where c_1 and c_2 are constants. This operation is implemented by choosing reactions with the appropriate coefficients:

$$c_1 X \longrightarrow c_2 Y. \tag{13}$$

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.

B. Adder

Addition performs the operation

$$y = x_1 + x_2.$$

This operation is implemented by choosing several reactions with the same product:

$$\begin{aligned} X_1 &\longrightarrow Y \\ X_2 &\longrightarrow Y. \end{aligned} \tag{14}$$

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

C. Fanout

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

$$X \longrightarrow Y_1 + Y_2. \tag{15}$$

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

A *transfer module* is a special case of fanout module. It simply transfers a molecular quantity from one type to another:

$$X \longrightarrow Y. \tag{16}$$

Transfer modules are used to resolve type assignment conflicts.

D. Delay Element

A delay element in an electronic DSP system stores a signal across successive clock cycles. It is implemented by clocked memory. Our scheme for biomolecular computation has no clock, of course. The system is completely asynchronous with variable rates. Yet we still seek to implement delay operations. We do so using "self-timed" operations based on a three-phase handshaking protocol that transfers quantities between molecular types based on the absence of other types. This is the three-phase scheme discussed for the moving-average filter in Section II.

Every delay element DE_i in a DSP specification is assigned three molecular types R_i, G_i and B_i. The following reactions implement the delay operation:

Phase 1 reactions:

$$\begin{aligned} b + R_i &\xrightarrow{\text{slow}} G_i \\ 2G_i &\xrightarrow{\text{slow}} I_{G,i} \\ I_{G,i} &\xrightarrow{\text{fast}} 2G_i \\ I_{G,i} + R_i &\xrightarrow{\text{fast}} 3G_i \\ Y &\xrightarrow{\text{collect}} \varnothing \end{aligned} \tag{17}$$

Phase 2 reactions:

$$\begin{aligned} r + G_i &\xrightarrow{\text{slow}} B_i \\ 2B_i &\xrightarrow{\text{slow}} I_{B,i} \\ I_{B,i} &\xrightarrow{\text{fast}} 2B_i \\ I_{B,i} + G_i &\xrightarrow{\text{fast}} 3B_i \\ \varnothing &\xrightarrow{\text{inject}} X \end{aligned} \tag{18}$$

Phase 3 reactions:

$$\begin{aligned} g + B_i &\xrightarrow{\text{slow}} Computations \\ 2R_j &\xrightarrow{\text{slow}} I_{R,j} \\ I_{R,j} &\xrightarrow{\text{fast}} 2R_j \\ I_{R,j} + B_i &\xrightarrow{\text{fast}} Computations + 2R_j \\ Computations &\xrightarrow{\text{fast}} R_j. \end{aligned} \tag{19}$$

A computation cycle, in which an input value is accepted 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. Computations are carried out in Phase 3, during the transfer from blue to red. The transfer diagram is shown in Figure 6. 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.

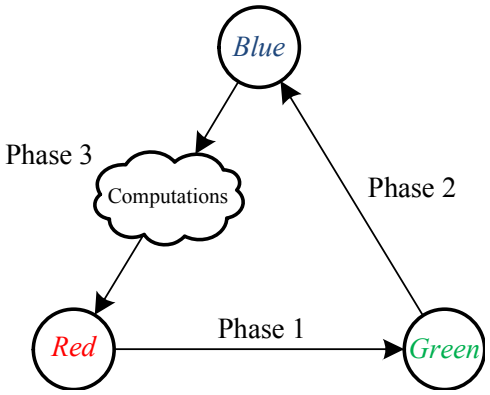


Fig. 6: The three-phase transfer scheme.

The following reactions generate the absence indicators types.

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} r \\ R_i + r &\xrightarrow{\text{fast}} R_i \\ Y + r &\xrightarrow{\text{fast}} Y \\ \varnothing &\xrightarrow{\text{slow}} g \\ G_i + g &\xrightarrow{\text{fast}} G_i \\ \varnothing &\xrightarrow{\text{slow}} b \\ B_i + b &\xrightarrow{\text{fast}} B_i \\ X + b &\xrightarrow{\text{fast}} X \end{aligned} \tag{20}$$

The absence indicators r, g and b are continuously and slowly generated. However, they only persist in the absence of the corresponding color-coded signal molecules, since they are quickly consumed by the signal molecules if these are present. This feature assures that so long as any reaction in a given phase has not fired to completion, the succeeding phase cannot begin.

There are only these three absence indicators r, g and b, regardless of the number of delay elements. Through these common absence indicators, the corresponding phases of *all* delay elements are synchronized: all the delay elements must wait for each to complete its current phase before they can all move to the next phase.

E. Example of an Biquad filter

We illustrate our synthesis method with a second example. Biquad filters are basic building blocks of modern DSP systems [10]. A biquad filter is shown in Figure 7 and the corresponding molecular types are labeled in Figure 8. This infinite impulse response (IIR) filter is realized by the

following reactions:

$$
\begin{aligned}
b + R_1 &\xrightarrow{\text{slow}} G_1 \\
2G_1 &\xrightarrow{\text{slow}} I_{G1} \\
I_{G1} &\xrightarrow{\text{fast}} 2G_1 \\
I_{G1} + R_1 &\xrightarrow{\text{fast}} 3G_1 \\
b + R_2 &\xrightarrow{\text{slow}} G_2 \\
2G_2 &\xrightarrow{\text{slow}} I_{G2} \\
I_{G2} &\xrightarrow{\text{fast}} 2G_2 \\
I_{G2} + R_2 &\xrightarrow{\text{fast}} 3G_2,
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
r + G_1 &\xrightarrow{\text{slow}} B_1 \\
2B_1 &\xrightarrow{\text{slow}} I_{B1} \\
I_{B1} &\xrightarrow{\text{fast}} 2B_1 \\
I_{B1} + G_1 &\xrightarrow{\text{fast}} 3B_1 \\
r + G_2 &\xrightarrow{\text{slow}} B_2 \\
2B_2 &\xrightarrow{\text{slow}} I_{B2} \\
I_{B2} &\xrightarrow{\text{fast}} 2B_2 \\
I_{B2} + G_2 &\xrightarrow{\text{fast}} 3B_2,
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
g + B_1 &\xrightarrow{\text{slow}} R_2 + F + C \\
2R_2 &\xrightarrow{\text{slow}} I_{R2} \\
I_{R2} &\xrightarrow{\text{fast}} 2R_2 \\
I_{R2} + B_1 &\xrightarrow{\text{fast}} 3R_2 + F + C \\
g + B_2 &\xrightarrow{\text{slow}} H + E \\
2Y &\xrightarrow{\text{slow}} I_Y \\
I_Y &\xrightarrow{\text{fast}} 2Y \\
I_Y + B_2 &\xrightarrow{\text{fast}} 2Y + H + E \\
g + X &\xrightarrow{\text{slow}} R_1 + A \\
I_Y + X &\xrightarrow{\text{fast}} 2Y + R_1 + A \\
2R_1 &\xrightarrow{\text{slow}} I_{R1} \\
I_{R1} &\xrightarrow{\text{fast}} 2R_1 \\
I_{R1} + X &\xrightarrow{\text{fast}} 3R_1 + A,
\end{aligned}
\tag{23}
$$

$$
\begin{aligned}
6A &\xrightarrow{\text{fast}} Y \\
6C &\xrightarrow{\text{fast}} Y \\
6E &\xrightarrow{\text{fast}} Y \\
6F &\xrightarrow{\text{fast}} X \\
6H &\xrightarrow{\text{fast}} X.
\end{aligned}
\tag{24}
$$

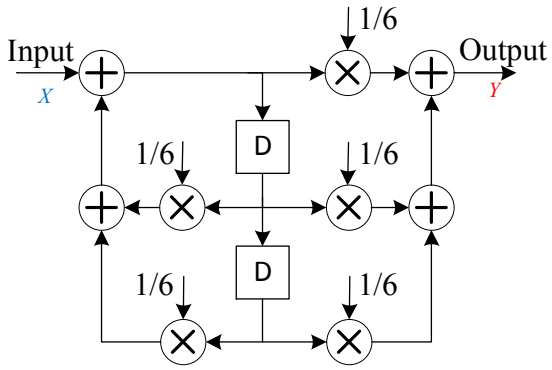


Fig. 7: A biquad filter.

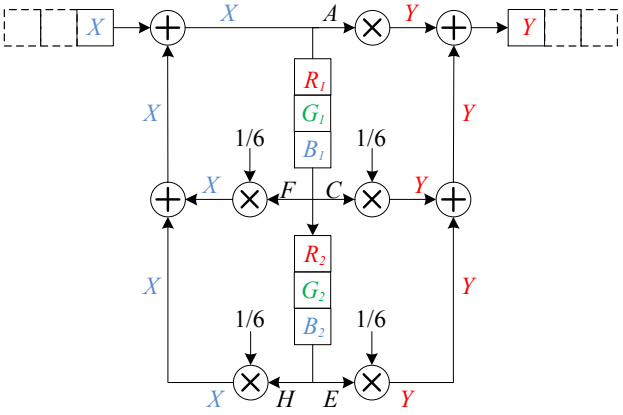


Fig. 8: Biquad filter in a three-phase configuration.

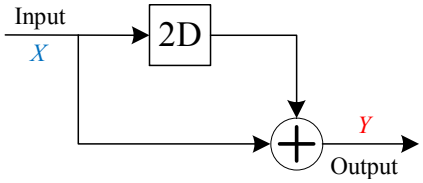
The absence indicator reactions are:

$$
\begin{aligned}
\varnothing &\xrightarrow{\text{slow}} r \\
R_1 + r &\xrightarrow{\text{fast}} R_1 \\
R_2 + r &\xrightarrow{\text{fast}} R_2 \\
Y + r &\xrightarrow{\text{fast}} Y \\
\varnothing &\xrightarrow{\text{slow}} g \\
G_1 + g &\xrightarrow{\text{fast}} G_1 \\
G_2 + g &\xrightarrow{\text{fast}} G_2 \\
\varnothing &\xrightarrow{\text{slow}} b \\
B_1 + b &\xrightarrow{\text{fast}} B_1 \\
B_2 + b &\xrightarrow{\text{fast}} B_2 \\
X + b &\xrightarrow{\text{fast}} X.
\end{aligned}
\tag{25}
$$

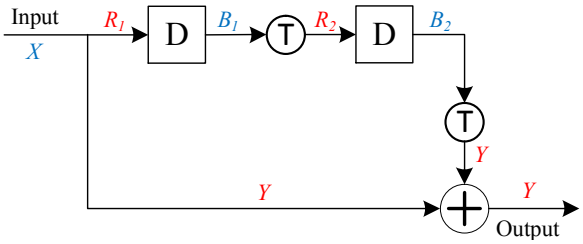
IV. Synthesis Flow

We present guidelines for an automated synthesis flow. The DSP system is represented by a block diagram $G(V, E)$, where the vertex set V represents basic modules – scalar multiplication, addition, fanout and delay element – and the edge set E represents connections. Each edge e_i is assigned a molecular type. The quantity of this type represents the signal flowing through e_i. The types are assigned as follows:

1) Each delay element $DE_i \in V$ is assigned three color-coded molecular types R_i, G_i and B_i. Here R_i corresponds to the input edge, G_i is the internal storage molecule type, and B_i corresponds to the output edge.



(a) A filter with two delay elements directly connected.



(b) Molecular type assignment.

Fig. 9: An example of molecular type assignment. Transfer modules are denoted by a circle with letter "T".

2) The system input and output are assigned types X and Y, respectively. (For simplicity, we only consider systems with a single input and a single output. However, the method easily generalizes to systems with multiple inputs and outputs.)
3) The incoming edges of each adder are assigned the same molecular type. With all the inputs assigned the same type, the system implicitly performs an addition operation: each reaction produces a quantity that is added to the sum.
4) If there are assignment conflicts, transfer modules are included. For instance, if an adder has been assigned two conflicting types T_1 and T_2, say because its inputs are from different delay operations, then a transfer reaction is included:

$$T_1 \xrightarrow{\text{fast}} T_2.$$

This reaction transfers the quantity of T_1 to T_2.
5) Next, if there any unassigned edges, these are assigned arbitrary molecular types (without creating conflicts).
6) With all edges assigned non-conflicting molecular types, reactions are generated for each vertex according to the template of reactions (13) to (19).
7) Finally, the absence indicator reaction set (20) is included. (Note that there is a single type r, a single type g and single type b for the entire system, regardless of the number of delay elements.)

Figure 9 shows an example of transfer modules. Figure 9a shows a simple filter for time-interleaved input data. It contains two delay elements. Since these two delay elements are directly connected, a transfer module is included for converting B_1 to R_2. Similarly, a second transfer module is included for transferring B_2 to Y, the molecular type for the adder. These molecular type assignments are shown in Figure 9b.

V. Simulation Results

We present simulation results for our biomolecular implementations of the two-tap moving-average and biquad filters, discussed in Sections II and III. We used Gillespie's stochastic simulation algorithm (SSA) [12], [13]. It performs a Monte Carlo simulation of the chemical kinetics.

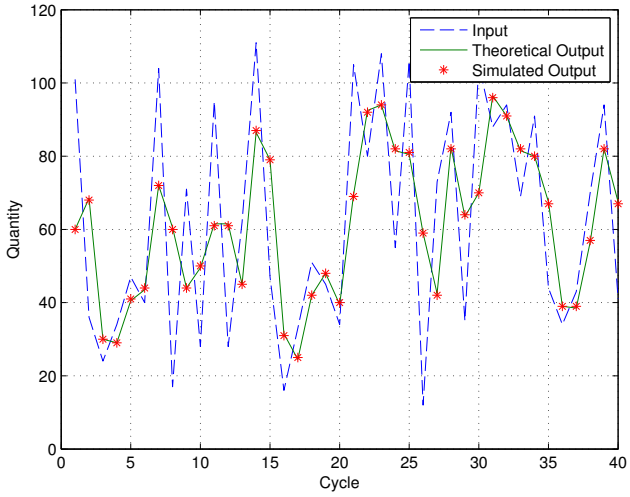


Fig. 10: Simulation results of moving average filter.

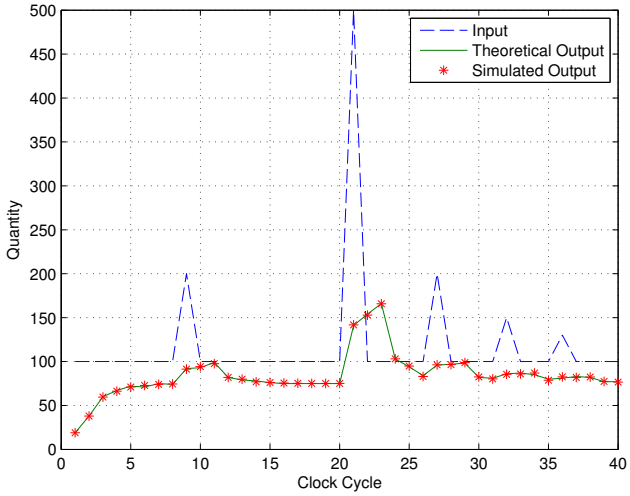


Fig. 11: Simulation results of biquad filter.

First, we performed simulations with a rate of 1 for "slow" and a rate of 100 for "fast." We generated 1000 trajectories and computed the mean values. The results for the two filters are plotted in Figures 10 and 11. The input for the moving-average filter is a random sequence and for the biquad filter it is a step followed by several pulse stimuli. The figures show the quantity of the input and output – types X and Y respectively – as a function of computational cycles. A cycle begins when we supply input molecules. It completes once we remove all the output molecules. (We always allow sufficient time for each cycle to complete before initiating the next.) The figures show the theoretical output, i.e., an exact calculation of filtering of the input, with a solid line. They

TABLE I: Relative error in simulations.

λ	Moving Average	Biquad
10	3.2479×10^{-3}	1.8531×10^{-3}
100	8.3496×10^{-4}	1.6890×10^{-3}
1000	5.2885×10^{-4}	1.5048×10^{-3}

show the simulated output with stars. Note that the simulated output has nearly perfect agreement with the theoretical output.

Next, we performed a sequence of simulations with different values for the fast-to-slow ratio, λ. We used the same input sequence each time, and varied λ from 10 to 1000, each time generating 1000 trajectories. The average relative errors for both filters for different values of λ are shown in Table I.

We see that the relative error decreases as λ increases. This is because a higher λ lowers the probability that a slow reaction misfires: i.e., it fires before all fast reactions are complete. We see that generally the biquad filter has a higher error than the moving-average filter. This is expected, since the IIR filters involve feedback that leads to error accumulation. In addition, there are more reactions for this filter; having more reactions increases the chance of obtaining an incorrect firing order.

VI. Remarks

We are exploring the mechanism of DNA-strand displacement advocated by Erik Winfree's group at Caltech as an experimental chassis [17]. They have shown that the kinetics of arbitrary chemical reactions can be emulated with DNA. They provide an assembler that accepts a set of chemical reactions with nearly any rate structure and delivers the corresponding DNA sequences for the displacement reactions. Reaction rates are controlled by designing sequences with different binding strengths. The binding strengths are controlled by the length and sequence composition of "toehold" sequences. With the right choice of toehold sequences, reaction rates differing by as much as 10^6 can be achieved. Our contribution can be positioned as the "front-end" of the design flow; the DNA assembler and experimental chassis described by these authors constitute the "back-end."

On one hand, the design flow discussed here is narrow in scope. It pertains to compiling high-level specifications of computation into abstract reactions. On the other hand, the flow is a *de novo* approach to rational design of functionality, and so potentially much more general in its applicability than methods based on appropriating and reusing existing biological modules.

The purpose of the flow is not to produce computation *per se*; molecular computing will never be competitive with conventional computing for tasks such as number crunching. Rather the main goal of the field is to develop embedded controllers of molecular and chemical processes. Imagine a situation where a *decision feedback equalizer* is implemented entirely through biomolecular reactions: the inputs and outputs are quantities of proteins; the result is a decision to deliver a drug or not, performed adaptively and autonomously.

Beyond the immediate challenges of automating the design of specific functions, a grander challenge is the design of a fully "technology independent" biomolecular CPU. At present, we know how to implement arithmetic and logical functions with biomolecular constructs [8]. We also know how to implement sequential, iterative operations transferring quantities between molecular types (contributions of [18] and this paper). The next step – more challenging than it seems – is to put together these elements to create a full-fledged processor, with a simple instruction set. The entire specification of the CPU would consist of technology-independent biomolecular reactions: abstract molecular types, such as a, b, c, etc., and reactions with qualitative rates, such as "slow" and "fast." This CPU design could be mapped onto DNA strand-displacement reactions or implemented through custom-gene synthesis in *E. coli* or yeast.

References

[1] 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.

[2] M. Sedlak and N. Ho, "Production of ethanol from cellulosic biomass hydrolysate using generically engineered yeast," *Applied Biochemistry and Biotechnology*, vol. 114, no. 1-3, pp. 403–416, 2004.

[3] 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.

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

[5] L. Qian and E. Winfree, "A simple DNA gate motif for synthesizing large-scale circuits," in *DNA Computing*, 2009, pp. 70–89.

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

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

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

[9] A. Oppenheim, R. Schafer, and J. Buck, *Discrete-Time Signal Processing*. Prentice-Hall, 1999.

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

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

[12] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[13] M. Gibson, "Computational methods for stochastic biological systems," Ph.D. dissertation, California Institute of Technology, 2000.

[14] S. Mauch, "Cain: Stochastic simulations for chemical kinetics." [Online]. Available: http://cain.sourceforge.net

[15] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976.

[16] ——, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, 2006.

[17] 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.

[18] H. Jiang, M. D. Riedel, and K. K. Parhi, "Digital signal processing with biomolecular reactions," in *IEEE Workshop on Signal Processing Systems*, 2010.