Project Description
Digital Signal Processing with Biomolecular Reactions

1 Context

In the nascent field of synthetic biology, researchers are striving to create biological systems with functionality not seen in nature. Examples include Salmonella that secretes spider silk proteins [58], yeast that degrades biomass into ethanol [49], and *E. coli* that produces antimalarial drugs [48].

The field aims to apply engineering methods to biology in a deliberate way. Beyond engineering ends, such methods also provide a constructive means to validating new science. As the great Caltech physicist Richard Feynman stipulated, "If I can't create it, I don't understand it." Understanding is achieved by constructing and testing simplified systems from the bottom up, teasing out and nailing down fundamental principles in the process [10]. As Drew Endy, an eloquent proponent of synthetic biology, describes it: natural biological systems are fiendishly complicated; instead of endlessly probing them with experiments, in some cases, we are better off rebuilding the functionality from the ground up. This provides engineered "surrogates" that are easier to understand and interact with.

Whether viewed as an engineering discipline that tinkers to achieve new functionality or one that builds systems from the ground up, synthetic biology is still in its early stages. 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. Creating and integrating synthetic components remains an ad hoc process. The field has now reached a stage where it calls for the development of *conceptual* design methodologies.

Richard Newton had a visionary view, articulated in a talk that he gave shortly before his death in 2007, titled "The Future is Bio-Design Automation" – a view reprised by Jan Rabaey in his keynote speech at the Design Automation Conference in 2007 commemorating Newton. The view is that synthetic biology represents but a new technological substrate for design automation. For electronics, we have a design methodology that involves clear abstractions, standardized interfaces, a constrained design space, and availability of intellectual property. The same requirements exist in biology; designers need to build models, compress them for analysis, and synthesize them into substrates such as *E. coli* or yeast. As Rabaey commented, "the potential synergy with electronic design automation (EDA) is huge." Indeed, there have been quite a few deliberate attempts to bring concepts from digital circuit design into synthetic biology [4, 5, 8, 57, 56, 60, 59, 40, 50, 54, 61].

2 Overview

We bring a particular mindset to tackle the problem of synthesizing new biological functions. We tackle synthesis at a *conceptual* level, working with abstract molecular types. Working at this level, we implement *computational* constructs, that is to say, chemical reaction networks that compute specific outputs as a function of inputs. Then we map the conceptual designs onto specific chemical substrates.

The idea of molecular computing dates back to seminal work by Len Adleman, who discussed solutions to combinatorial problems such as the Hamiltonian Path Problem and Boolean Satisfiability [1]. In spite of the initial hype, including claims of massive parallelism – "100 Teraflop performance in a test tube!" screamed the headlines at the time – such applications were never compelling. Chemical systems are inherently slow and messy, taking minutes or even hours to finish, and producing fragmented results. Such systems will never be competitive with conventional silicon computers for tasks such as number crunching.

Rather, the impetus of this research is the design of "embedded controllers" – chemical reactions, engineered into biological systems such as viruses and bacteria, to perform useful molecular computation *in situ* where it is needed. For example, consider a system for chemotherapy drug delivery with engineered bacteria. The goal is to get bacteria to invade tumors and selectively produce a drug to kill the cancerous cells. Embedded control of the bacteria is needed to decide where and how much of the drug they should deliver. The computation could be as simple as: "If chemical type X is present, produce chemical type Y" where X is a protein marker of cancer and Y is the chemo drug. Or it could be more complicated: produce Z if X is present and Y is not present or vice-versa (i.e, an exclusive-or function). Or it could be time-varying computation: produce Z if the rate of change of X is within certain bounds (i.e., band-pass filtering). Exciting recent work along these lines includes [3] and [55].

This work is a continuation of our current research funded by an NSF EAGER grant, CCF-0946601. In prior work, we had described a computational constructs for chemical reaction networks: logical operations such as copying, comparing and incrementing/decrementing [52]; programming constructs such as "for" and "while" loops [53]; and arithmetic operations such as multiplication, exponentiation and logarithms [52, 53]. The EAGER grant has allowed use to make significant advances in synthesizing digital processing (DSP) operations such as filtering [23]. As we describe below, we have provided the first conceptual designs of sequential iterative computation. Indeed, DSP systems are non-terminating in nature, i.e., the same computation is repetitively executed. Input signals are sampled and processed iteratively. As a proof of concept, we have described robust finite-impulse response (FIR) and infinite-impulse (IIR) filters. We have mapped these designs to DNA strand-displacement reactions.

Unlike 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. 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. We achieve iterative computation through a self-timed, three-phase synchronization protocol that we call RGB (for *Red*, *Blue* and *Green*) that transfers quantities between molecular types based on the absence of other types. As we describe below, the scheme can be used to implement *delay elements*. Together with biomolecular constructs for operations such *addition*, *scalar multiplication*, and *fanout*, we present a methodology for implementing arbitrary DSP operations.

The PIs on this project Keshab Parhi and Marc Riedel are, respectively, experts in VLSI signal processing architectures and molecular computing. Together, our collaboration over last two years has led to co-supervision of three students, Hua Jiang, Philip Senum and Sasha Kharam, working on these topics. This grant will allow us to continue our collaboration beyond the duration of the EAGER grant which expires in July 2011.

Specifically, this project will address the following design challenges: transforming time-domain signals into spectral-domain signals by Fast Fourier Transform (FFT) operations; implementing general filtering operations such as high, low, and band-pass filtering; implementing crosstalk cancellation; and implementing decision equalization operations. (In all cases, the inputs and outputs are time-varying quantities of molecular types. For instance, in the case of an FFT, the output is a time-varying quantity that corresponds to the frequency of the changes in the input quantity.) Bipolar encoding techniques will be developed for implementing filters with negative coefficients. The impact of specific DSP constructs such as pipelining, retiming, folding and unfolding on biomolecular designs will be investigated. The methodology will be developed and evaluated both in a conceptual sense and in a practical sense. All DSP designs will be translated to DNA-strand displacement reactions.

3 Computational Model

The theory of reaction kinetics underpins our understanding of biological and chemical systems [22]. It is a simple and elegant formalism: chemical reactions define *rules* according to which reactants form products; each rule fires at a *rate* that is proportional to the quantities of the corresponding reactants that are present. On the computational front, there has been a wealth of research into efficient methods for simulating chemical reactions, ranging from ordinary differential equations (ODEs) [12] to stochastic simulation [15].

Consider the reaction

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

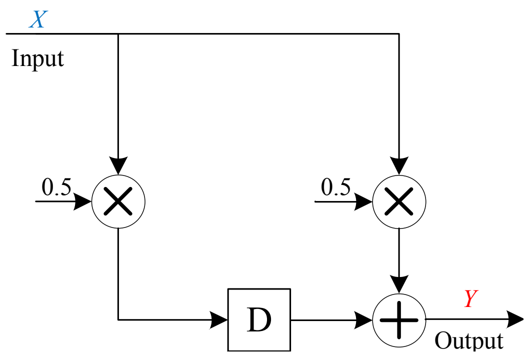
When this reaction fires, one molecule of a is consumed, one of b is consumed, and two of c are produced. (Accordingly, a and b are called the *reactants* and c the *product*.) The rate of the reaction is proportional to the quantities of a and b present, as well as to the reaction constant, in this case "fast." Although we refer to rates in relative and qualitative terms – e.g., “fast” vs. “slow” – these are, in fact, quantitative values that are either deduced from biochemical principles or measured experimentally.

A description of a biomolecular system in terms of a collection of reactions is analogous, in some ways, to a transistor netlist. Stochastic simulation and ODE solvers are, in some ways, analogous to SPICE [35]. As with an electronic system, once a biomolecular system is specified, the task of simulating it is well understood. Indeed, the mindset of *analysis* still prevails in this domain: a set of chemical reaction exists, designed by nature and perhaps modified by human engineers; the objective is to understand and characterize its behavior. Comparatively little work has been done at a conceptual level in tackling the inverse problem of *synthesis*: how can one design a set of chemical reactions that implement specific behavior?

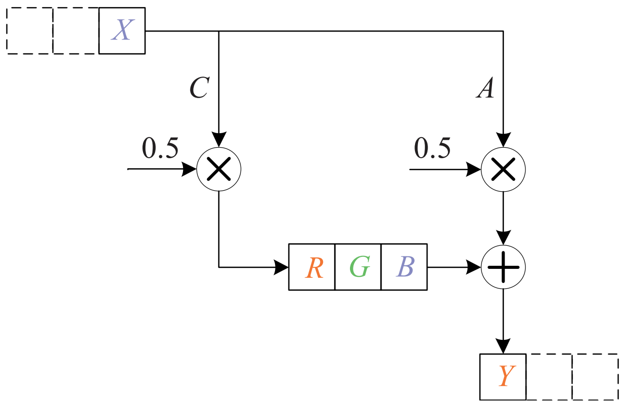
4 Proof of Concept

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.

4.1 Simplified Form



(a) Schematic.



(b) Three-phase configuration.

Figure 1: A two-tap moving average filter.

An FIR filter is shown in Figure 1a. 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 1b. 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 supplies the input. The output Y is produced in the blue-to-red phase. We assume that an external sink consumes these molecules.

4.2 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.

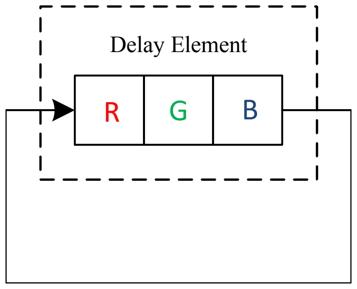


Figure 2: RGB cycle in isolation.

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 2. Suppose that this cycle is implemented with the following reactions:

$$\begin{array}{lcl} b+R & \xrightarrow{\text{slow}} & G \\ r+G & \xrightarrow{\text{slow}} & B \\ g+B & \xrightarrow{\text{slow}} & R \end{array} \quad (6) \qquad \begin{array}{lcl} \varnothing & \xrightarrow{\text{slow}} & r \\ \varnothing & \xrightarrow{\text{slow}} & g \\ \varnothing & \xrightarrow{\text{slow}} & b \end{array} \quad (7) \qquad \begin{array}{lcl} R+r & \xrightarrow{\text{fast}} & R \\ G+g & \xrightarrow{\text{fast}} & G \\ B+b & \xrightarrow{\text{fast}} & B. \end{array} \quad (8)$$

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 4.2. Here we simulated the chemical kinetics for Reaction (6)–(8) [34]. In this figure, we see that the quantities oscillate with an attenuating envelope and converge to one third of the initial quantity of R.

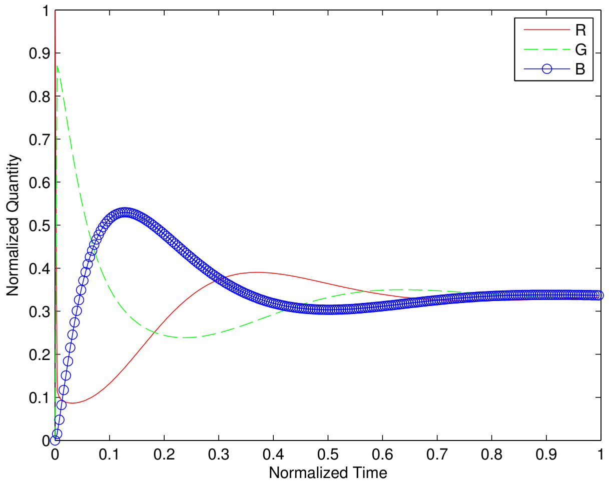


Figure 3: Simulation of the RGB Transfer Reactions (6)–(8).

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:

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

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 (9) 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

$$\begin{aligned} 2B &\xrightarrow{\text{slow}} I_B \\ I_B &\xrightarrow{\text{fast}} 2B \\ I_B + G &\xrightarrow{\text{fast}} 3B, \end{aligned} \tag{10}$$

and from B to R:

$$\begin{aligned} 2R &\xrightarrow{\text{slow}} I_R \\ I_R &\xrightarrow{\text{fast}} 2R \\ I_R + B &\xrightarrow{\text{fast}} 3R. \end{aligned} \tag{11}$$

With the reactions (9), (10), and (11), 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 4.2 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.

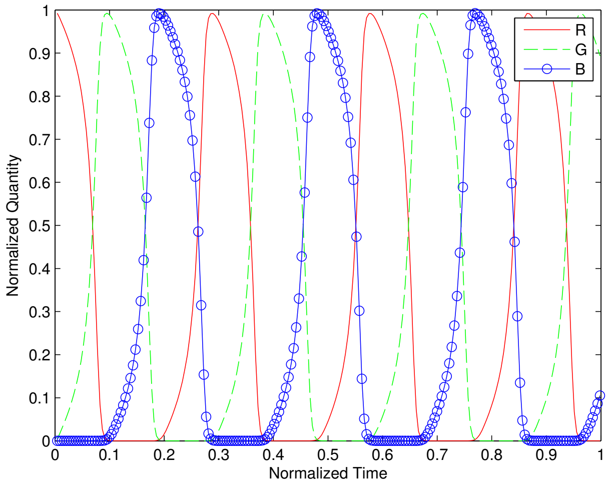


Figure 4: Simulation of the RGB transfers with the Reactions (9), (10), and (11).

[1] A rigorous discussion of chemical kinetics is beyond the scope of this proposal. Interested readers can consult the references [15, 14, 16] and [17].

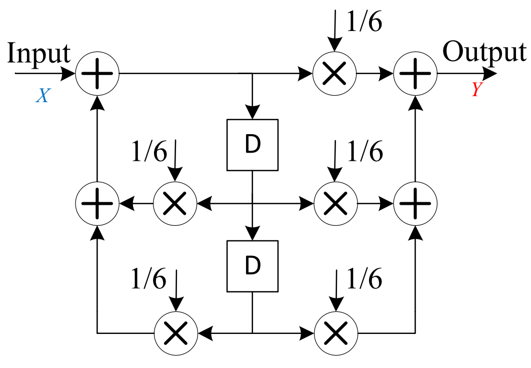
With these refinements, the full set of reactions for the moving average filter is:

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

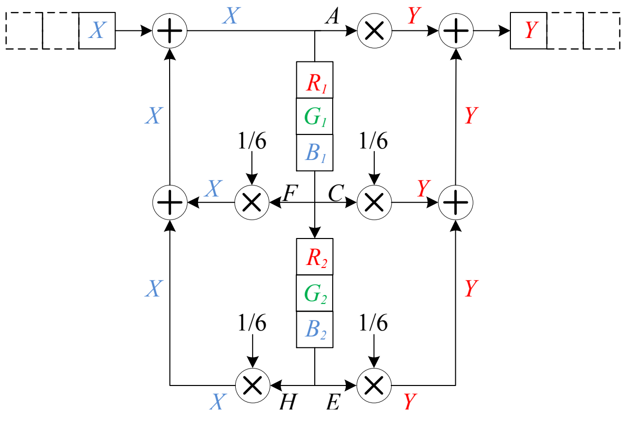
$$\begin{aligned}
g + X &\xrightarrow{\text{slow}} A + C \\
2C &\xrightarrow{\text{fast}} R \\
2A &\xrightarrow{\text{fast}} Y \\
b + R &\xrightarrow{\text{slow}} G \\
r + G &\xrightarrow{\text{slow}} B \\
g + B &\xrightarrow{\text{slow}} Y,
\end{aligned} \tag{13}$$

$$\begin{aligned}
\varnothing &\xrightarrow{\text{slow}} r \\
R + r &\xrightarrow{\text{fast}} R \\
Y + r &\xrightarrow{\text{fast}} Y \\
\varnothing &\xrightarrow{\text{slow}} g \\
G + g &\xrightarrow{\text{fast}} G \\
\varnothing &\xrightarrow{\text{slow}} b \\
B + b &\xrightarrow{\text{fast}} B \\
X + b &\xrightarrow{\text{fast}} X.
\end{aligned} \tag{14}$$

4.3 Simulation Results



(a) Schematic.

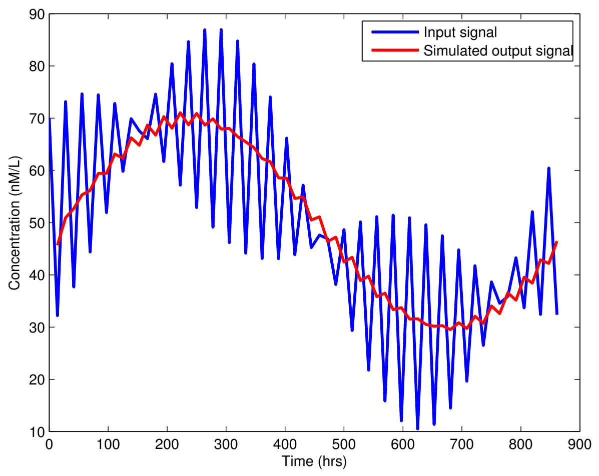


(b) Three-phase configuration.

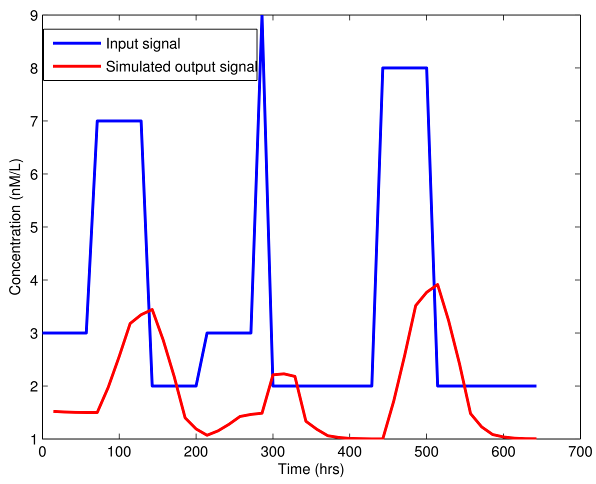
Figure 5: A biquad filter.

We present simulation results for our biomolecular implementations of the two-tap moving-average filter, shown in Figure 1a, as well as a biquad filter, shown in Figure 5. We used Gillespie's stochastic simulation algorithm (SSA) [15, 14]. It performs a Monte Carlo simulation of the chemical kinetics. 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 6a and 6b.

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.) For the moving average filter, the input signal is composed of a low-frequency component and a high-frequency component. In the output signal, the high-frequency component is almost filtered out. For the biquad filter, the input signal contains abrupt changes. The output signal is a smoothed version of the input signal. In both cases, the simulation results show nearly perfect agreement with the expected behavior.



(a) Simulation results of moving average filter.



(b) Simulation results of biquad filter.

Figure 6: Simulation results of two filters.

Table 1: 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}

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 1.

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.

5 Proposed Work

Building upon this preliminary work, this project will develop a complete methodology for synthesizing DSP operations with biomolecular reactions. DSP systems are generally specified in terms of four basic modules: *fanout*, *scalar multiplication*, *addition*, and *delay elements*. Biomolecular constructs for these four modules were illustrated with the moving-average filter in the previous section.

The first deliverable of the project will be a methodology for synchronize transfers through delay elements with a *clock signals*. Then, building upon such a framework for sequential computation, the project will deliver:

- A methodology for performing time-to-frequency domain transformations with Fast Fourier Transforms (FFTs).
- A methodology for performing decision-feedback equalization (DFE).
- A methodology for echo and near-end crosstalk (NEXT) cancellation.

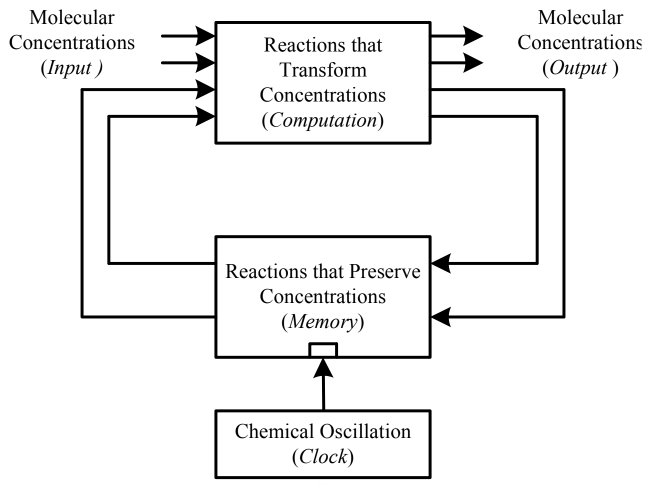

Molecular Concentrations (*Input*)
Reactions that Transform Concentrations (*Computation*)
Molecular Concentrations (*Output*)
Reactions that Preserve Concentrations (*Memory*)
Chemical Oscillation (*Clock*)

Figure 7: Block diagram of a synchronous sequential system.

- A methodology for complex filter designs, including those with negative coefficients.

Finally, the impact of specific DSP constructs such as pipelining, retiming, folding and unfolding on biomolecular designs will be investigated.

5.1 Synchronous Sequential Computation

The general structure of our design is illustrated in Figure 7. As in an electronic system, our molecular system consists of separate reactions that implement *computation* and *memory.* A clock signal synchronizes transfers between computation and memory. For the computational reactions, we refer the reader to prior work [23, 52, 53]. Operations such as addition and scalar multiplication are straightforward. Operations such as multiplication, exponentiation, and logarithms are trickier. These can be implemented with reactions that implement iterative constructs analogous to "for" and "while" loops. (They do so robustly and exactly, without any specific dependence on the rates.)

One of the main contributions of project will be a novel method for clock signal generation and for implementing memory.

5.1.1 Clock Generation

In electronic circuits, a clock signal is generated by an oscillatory circuit that produce periodic voltage pulses. For a molecular clock, we choose reactions that produce sustained oscillations in the chemical concentrations. With such oscillations, a low concentration corresponds to logical value of zero; a higher concentration corresponds to a logical value of one. Techniques for generating chemical oscillations are very well-known in the literature. Classic examples include the Lotka-Volterra, Brusselator and Arsenite-Iodate-Chlorite systems [11, 25]. However, none of these schemes are quite suitable for synchronous sequential computation. We require that the clock signal be perfectly symmetrical, with abrupt transitions between the phases.

We will develop a design for a four-phase chemical oscillator. The clock phases will be represented by molecular types $R(ed)$, $G(reen)$, $B(lue)$, and $Y(ellow)$. First consider the reactions in Figure 8. Reactions 15 generates molecular types r, g, b, and y slowly and constantly. Here the symbol $\varnothing$ indicates "no reactants" meaning the products are generated from a large or replenishable source. In Reactions 16, the types R, G, B, and Y quickly consume the types r, g, b, and y, respectively. Call R, G, B, and Y the *phase signals* and

$$\begin{aligned} \varnothing &\xrightarrow{\text{slow}} r \\ \varnothing &\xrightarrow{\text{slow}} g \\ \varnothing &\xrightarrow{\text{slow}} b \\ \varnothing &\xrightarrow{\text{slow}} y \end{aligned} \qquad (15) \qquad\qquad \begin{aligned} R + r &\xrightarrow{\text{fast}} R \\ G + g &\xrightarrow{\text{fast}} G \\ B + b &\xrightarrow{\text{fast}} B \\ Y + y &\xrightarrow{\text{fast}} Y. \end{aligned} \qquad (16)$$

Figure 8: Reactions for a 4-Phase Oscillator.

r, g, b, and y the *absence indicators*. The latter are only present in the absence of the former. The reactions

$$\begin{aligned} R + y &\xrightarrow{\text{slow}} G \\ G + r &\xrightarrow{\text{slow}} B \\ B + g &\xrightarrow{\text{slow}} Y \\ Y + b &\xrightarrow{\text{slow}} R \end{aligned} \qquad (17)$$

transfer one phase signal to another, in absence of its previous one. The essential aspect is that, within the RGBY sequence, the full quantity of the preceding type is transfered to the current type before the transfer to the succeeding type begins.

To achieve sustained oscillation, we introduce positive feedback. This is provided by reactions

$$\begin{aligned} 2G &\underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} I_G \\ R + I_G &\xrightarrow{\text{fast}} 3G \\ 2B &\underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} I_B \\ G + I_B &\xrightarrow{\text{fast}} 3B \\ 2Y &\underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} I_Y \\ B + I_Y &\xrightarrow{\text{fast}} 3Y \\ 2R &\underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} I_R \\ Y + I_R &\xrightarrow{\text{fast}} 3R. \end{aligned} \qquad (18)$$

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 $2G \xrightarrow{\text{slow}} I_G$, 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.[2]

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 Y. We choose two nonadjacent phases, R and B, as the clock phases.

Our scheme for chemical oscillation works remarkably well. Figure 9 shows the concentrations of R and B as a function of time, obtained through ordinary differential equation (ODE) simulations of the reactions 15, 16, 17 and 18. We note that the R (red) and B (blue) phases are non-overlapping.

[2]A rigorous discussion of chemical kinetics is beyond the scope of this paper. Interested readers can consult [11].

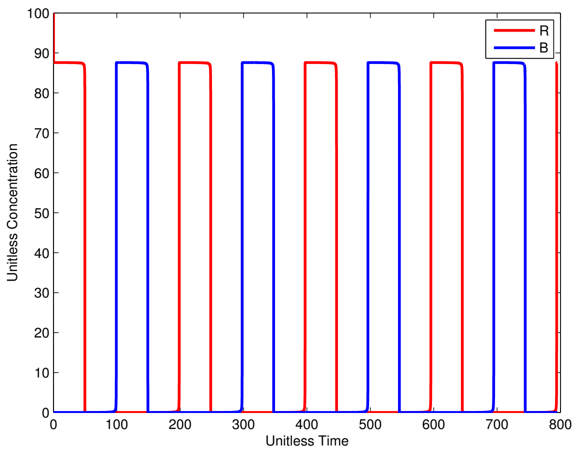


Figure 9: ODE simulation of the chemical kinetics of the proposed clock.

5.1.2 Memory

To implement sequential computation, we must store and transfer signals across clock cycles. In electronic systems, storage is typically implemented with flip-flips. In our molecular system, we implement storage and transfer using a two-phase protocol, synchronized on phases of our clock. Every memory unit S_i is assigned two molecular types D'_i and D_i. Here D'_i is the first stage and D_i the second.

The blue phase reactions are:

$$\begin{aligned} B + D_i &\xrightarrow{\text{fast}} Computations + B \\ Computations &\xrightarrow{\text{fast}} D'_j. \end{aligned} \tag{19}$$

Every unit S_i releases the signal it stores in its second stage D_i. The released signal is operated on by reactions in *computational modules*. These generate results and push the them into the first stages of succeeding memory units. Note that D'_j molecules will be the first stage of any succeeding memory unit S_j along the signal path from S_i.

The red phase reactions are

$$R + D'_i \xrightarrow{\text{fast}} D_j + R. \tag{20}$$

Every unit S_i transfers the signal it stores in D'_i to D_j, preparing for the next cycle. For the equivalent of delay (D) flip-flops in digital logic, $i = j$. For other types of memory units, i and j can be different. For example, for a toggle (T) flip-flop, S_j is the complementary bit of S_i: $D'_i \longrightarrow D_j$ and $D'_j \longrightarrow D_i$ toggle the pair of bits in each clock cycle. The transfer diagram for our memory design is shown in Figure 10.

5.2 More Sophisticated FIR/IIR Filters

The project will investigate design techniques for implementing sophisticated DSP filtering operations. In all of our existing designs, the signals are positive numbers. This is not surprising since the signals correspond to quantities of molecular types; obviously, we cannot have negative quantities of molecules. And yet, negative numbers are critical for many DSP operations.

We will investigate techniques for implementing negative values through bipolar encodings: a signal X will be represented by two molecular types X_p and X_n, where X_p represents the positive component, and X_n the negative component. Accordingly, X equals $[X_p] - [X_n]$. Such a representation is similar to a signed binary number representation (SBNR) representation in electronic designs. Concepts established for operating on redundant number system will be explored. Consider the reaction

$$X_p + X_n \xrightarrow{\text{fast}} \varnothing \tag{21}$$

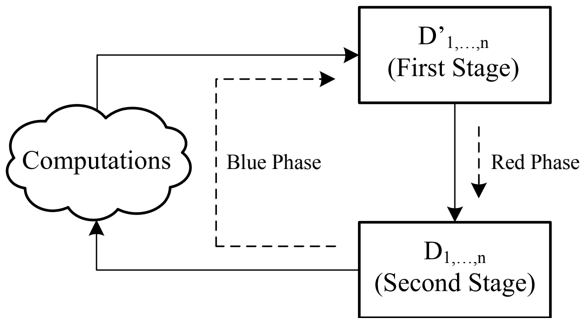


Figure 10: The two-phase memory transfer scheme.

It cancels out equal-valued quantities $[X_p]$ and $[X_n]$ whenever possible. Given a representation of negative numbers, subtraction can be implemented by addition of negative numbers.

With a scheme for representing negative numbers, it will possible to implement negative-coefficient scalar multipliers:

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

is implemented by

$$\begin{array}{rcl} c_1 X_p & \longrightarrow & c_2 Y_n \\ c_1 X_n & \longrightarrow & c_2 Y_p. \end{array} \tag{22}$$

With negative-coefficient multipliers, it will be possible to implement a broad range of filters, including band-pass and high-pass filters. The proposed effort will be directed towards integrating these concepts with various digital filters such as FIRs and IIRs. The kinetics and robustness of these reactions will be studied.

5.3 FFT

Given a robust biomolecular clocking scheme, we will implement Fast Fourier Transforms (FFTs). The FFT operation is used in spectral or frequency-domain analysis of systems. A variety of FFT architectures have been proposed [6, 13, 19, 20, 51]. The signal flow graph and and a possible 2-parallel architecture of a 16-point FFT system are shown in Figures 11a and 11b, respectively. A simple example of a 4-point two-parallel FFT system is shown in Figure 11c.

An FFT operation is predicated on a representation of *complex* numbers. To this end, we will use two molecular types X_p^* and X_n^*, assigned to the complex part of a signal X, in addition to X_p and X_n. We include a reaction of the form:

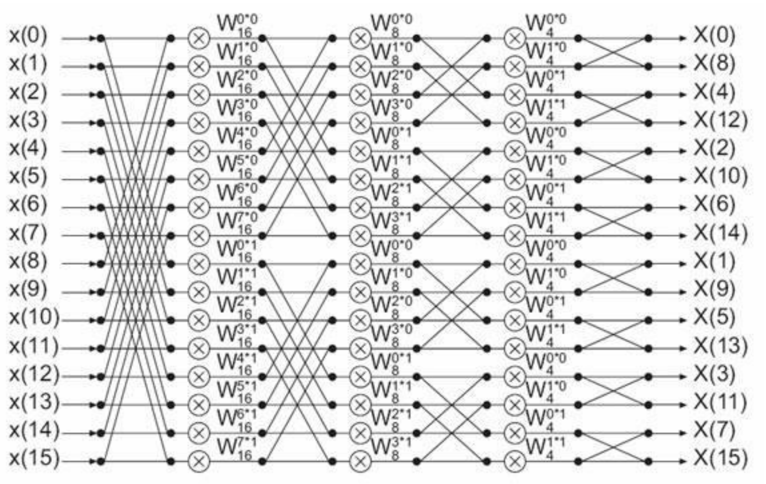
$$X_p^* + X_n^* \xrightarrow{\text{fast}} \varnothing. \tag{23}$$

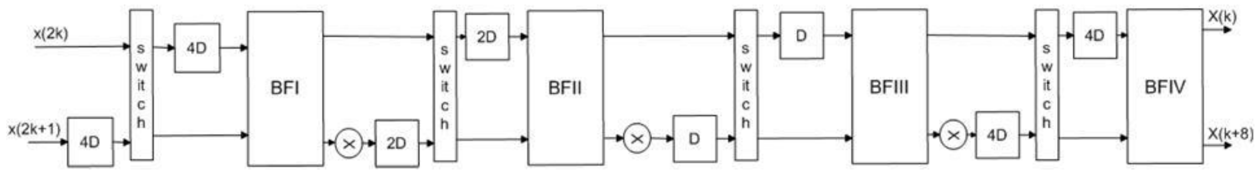
There are several switches in FFT systems. Each selects one of the two incoming signals alternatively in different cycles. To achieve this switching functionality in our molecular design, we use two alternating selection signals. With the techniques proposed in Section 5.1, we generate these with a pair of D-flip-flops, as shown in Figure 12. We implement this computation with following reactions:

$$\begin{array}{rcl} R + S_0' & \xrightarrow{\text{fast}} & S_0 + R \\ R + S_1' & \xrightarrow{\text{fast}} & S_1 + R \\ B + S_1 & \xrightarrow{\text{fast}} & S_0' + B \\ B + S_0 & \xrightarrow{\text{fast}} & S_1' + B \end{array} \tag{24}$$

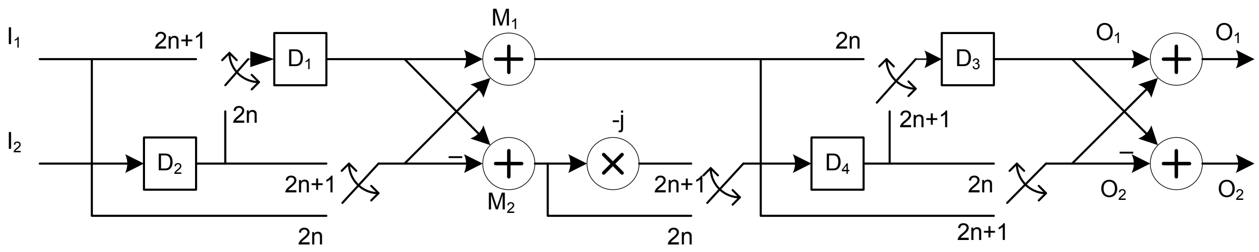
The transfer reactions enabled by S_1' or S_0' implement the switches.

With the implementation of clock, delay elements, switch, and computational modules, and representation of negative and complex numbers, a complete design of an FFT system will be possible.


(a)



(b)



(c)

Figure 11: Fast Fourier Transform

5.4 Decision-Feedback Equalization

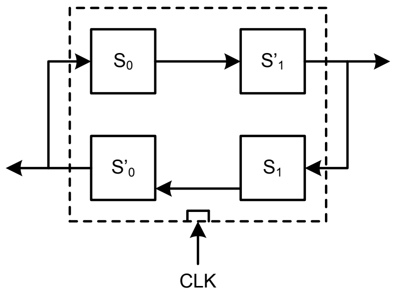


Figure 12: Generating the selection signals.

Decision-feedback equalization exploits prior decisions to eliminate the time-dispersion of signals [21, 18, 39, 47, 38, 36, 9]. A simple decision-feedback equalizer (DFE) is shown in Figure 13. In this system, the previous output signals are filtered and added to or subtracted from new input signal. DFE operations are used to cancel crosstalk and other types of interference from signal paths.

Making use of our comparator module, described in [52], will implement the requisite thresholding operation. Based upon the synchronization technique discussed above, we will implement the equalizer.

5.5 Echo and NEXT Cancellation

Will will apply our methodology to the design of systems for echo and near-end crosstalk (NEXT) cancellation. A diagram for echo cancellation is shown in Figure 14.

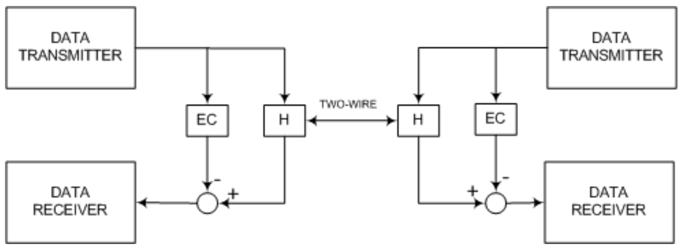


Figure 14: A diagram for echo cancellation.

5.6 Impact of High-Level Transformation

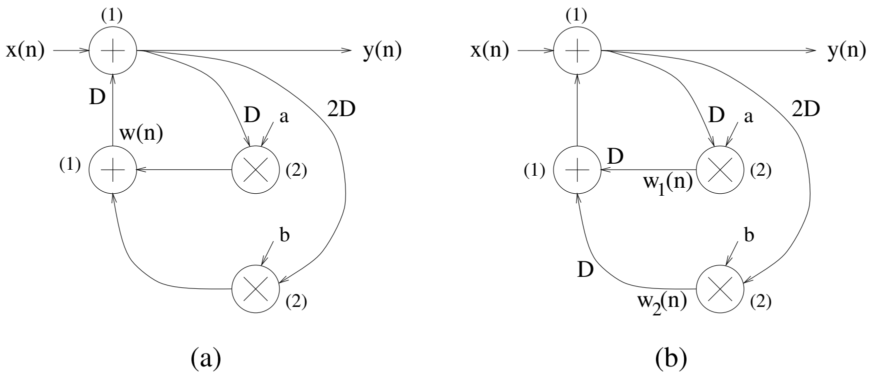


Figure 15: (a) Original system. (b) System after retiming. (Taken from [37].)

In DSP systems, high-level transforms techniques, such as retiming, unfolding, and folding, are used to manipulate a circuit-level design while preserving the functionality [37]. The goal is to modified the circuit to improve area, computation speed, power and other metrics.

Retiming is a transformation that changes the locations of delay elements in circuits yet perserves the functionality, as shown in Figure 15. By rearranging the delay elements, we may minimize the total the number delay operations. Unfolding is a transformation that creates more than one iteration of a DSP program, as shown in Figure 16. It is used for high-speed processing and low-cost computations. Folding transformations are used for time-multiplexing functional units, such as computational modules, as shown in Figure 17. They provides a

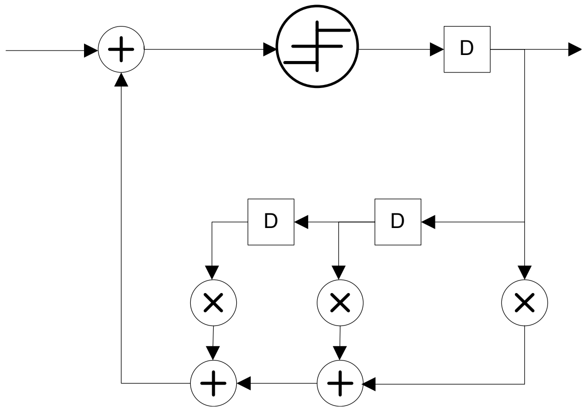


Figure 13: A decision-feedback equalizer.

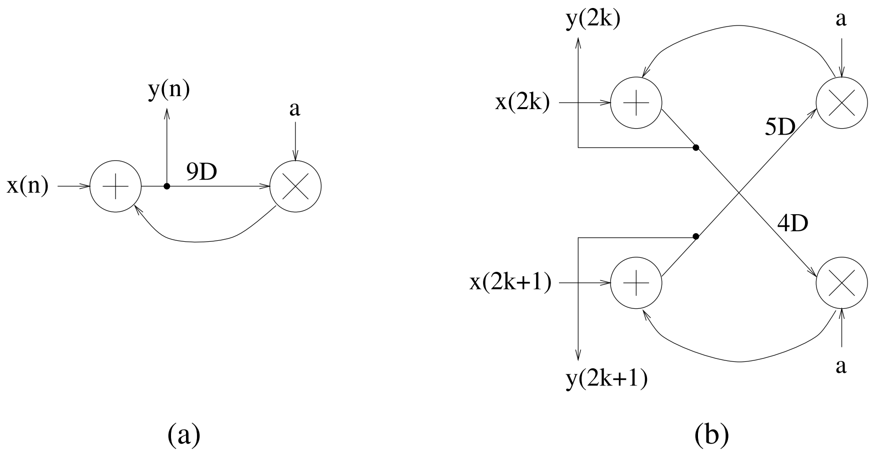


Figure 16: (a) Original system. (b) System after unfolding. (Taken from [37].)

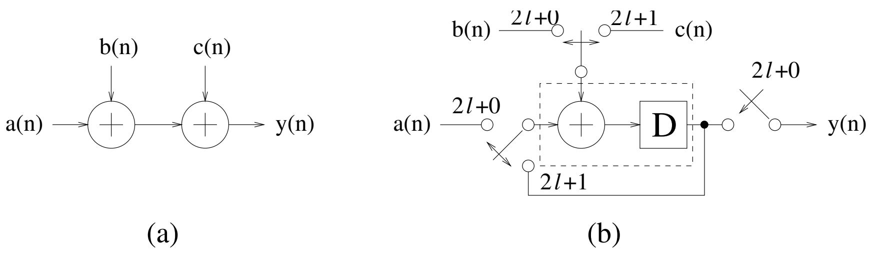


Figure 17: (a) Original system. (b) System after folding. (Taken from [37].)

means for trading system complexity for processing
speed.

In biomolecular systems, the total number of reactions and the total number of molecular types are typically the relevant metrics of system complexity. We will investigate the impact of high-level transformation on these metrics.

6 Educational and Outreach Plan

6.1 Minority Involvement Plan

The PIs will work with the University of Minnesotas College of Science and Engineering Diversity and Outreach program to involve underrepresented students in research. This program manages the NSF-funded North Star STEM Alliance–Minnesotas Louis Stokes Alliance for Minority Participation (LSAMP). One of the core principles of the Diversity and Outreach program is that Mentoring and introduction of research opportunities early in the undergraduate career is the best practice for retention. Through participation in the North Star programs, the students will present their research to North Star fellows to demonstrate their research. They can choose from a selection of outreach events that are provided by the North Star program including a Kickoff Day at the beginning of each year and a spring symposium in the spring semester to showcase research opportunities at the university. Each student will participate in one of these events during their fellowship. The undergraduate students attending these presentations are encouraged by North Star program to seek research positions in labs. North Star also supplies funding for underrepresented students to attend conferences when mentored by a graduate student to increase the exposure of the students to the research community beyond the Universitys laboratories.

6.2 Undergraduate Involvement in Research

The University of Minnesota offers many research opportunities for undergraduate research. Undergraduate research is supported by the university through the Undergraduate Research Opportunity Program. This is a competitive program that requires the students to write a proposal which gets reviewed and scored. The UROP program funds approximately 80% of the applications providing the students with $1400 stipend and $300 for lab supplies. These students generally are mentored by a graduate student in the lab. This provides graduate students the opportunity to learn mentoring skills and to develop interest in their field. The undergraduates can present their research at the end of the year in an undergraduate research symposium.

6.3 K-12 Outreach Plan

The College of Science Engineering (CSE) offers a summer high school student outreach program, Exploring Careers in Engineering and Physical Science (ECEPS). This program offers students a handson introduction to engineering, science and math opportunities on the University of Minnesota Twin Cities campus by providing the students tours, along with short projects, in different labs around the campus. This program is designed to appeal to and reach both girls and underrepresented minorities with an interest in the STEM disciplines. In particular, two of the four possible oneweek sessions are devoted to girls only.

6.4 Integration of Research into Course Curriculum

PI Riedel will integrate results from this research into a new course EE-5393 "Circuits, Computations and Biology" offered jointly through the ECE Department and the new Biomedical Informatics and Computational Biology Program at the University of Minnesota. PI Parhi teaches the courses EE-5329 "VLSI Signal Processing" and EE-5542 "Adaptive Filters." Students in these classes acquire advanced expertise in digital signal processing. (Some of the channel estimation effort through adaptive algorithms in a biochemical environment may be pursued in the EE-5542 if data become available.) These three courses provide an excellent foundation for the research topic of this proposal.

7 Results of Prior NSF Support

PI Parhi has two active NSF grants: 1) Award CCF-0811456: "Collaborative Research: CPA-DA: Noise-Aware VLSI Signal Processing: A New Paradigm for Signal Processing Integrated Circuit Design in Nanoscale Era," started on 9/1/2008; and 2) EAGER grant CCF-0946601: "Synthesizing Signal Processing Functions with Biochemical Reactions" (with M. Riedel) started on 9/1/09. The CCF-0811456 grant has enabled us to create a tool for estimation of power consumption by estimating switching activity in arithmetic circuits, to reduce power consumption in frequency-selective FIR filters by correction circuitry, and to improve reliability of demodulation in orthogonal frequency division multiplexing (OFDM) systems. These results have been published in [27, 28, 29, 30, 31, 32, 33].

PI Riedel has two active NSF grants: 1) CAREER Award 0845650: "Computing with Things Small, Wet, and Random – Design Automation for Digital Computation with Nanoscale Technologies and Biological Processes" started on 9/15/2009; and 2) EAGER grant CCF-0946601 "Synthesizing Signal Processing Functions with Biochemical Reactions" (with K. Parhi) started on 9/1/09. The CAREER award has established novel and transformative approaches to design automation guided by physical views of computation. A broad theme is the application of expertise from an established field, digital circuit design, to new fields, such as nanotechnology and synthetic biology. The results have been published in [2, 7, 23, 24, 26, 41, 42, 43, 44, 45, 46, 52, 53].

References Cited

[1] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(11):1021–1024, 1994.

[2] M. Altun and M. D. Riedel. Lattice-based computation of Boolean functions. In *Design Automation Conference*, pages 609–612, 2010.

[3] 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*, 355(4):619–627, 2006.

[4] J. C. Anderson, C. A. Voigt, and A. P. Arkin. A genetic AND gate based on translation control. *Molecular Systems Biology*, 3(133), 2007.

[5] A. Arkin and J. Ross. Computational functions in biochemical reaction networks. *Biophysical Journal*, 67(2):560 – 578, 1994.

[6] M. Ayinala and K. K. Parhi. Parallel-pipelined radix-2^2 FFT architecture for real valued signals. In *Proceedings of Asilomar Conference on Signal Systems and Computers*, 2010.

[7] J. Backes and M. D. Riedel. Reduction of interpolants for logic synthesis. In *International Conference on Computer-Aided Design*, 2010.

[8] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423–429, 2004.

[9] J. Chen and K. K. Parhi. Fast computation of mimo equalizers and cancellers in 10gbase-t channels. In *Proc. of IEEE ICASSP*, pages III–201–III–204, 2007.

[10] D. Endy. Foundations for engineering biology. *Nature*, 438:449–453, 2005.

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

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

[13] M. Garrido, K. K. Parhi, and J. Grajal. A pipelined FFT architecture for real-valued signals. *IEEE Trans. Cir. Sys. I, Reg. Papers*, 56(12):2634–2643, Dec 2009.

[14] M. Gibson. *Computational Methods for Stochastic Biological Systems.* PhD thesis, California Institute of Technology, 2000.

[15] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[16] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[17] D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2006.

[18] S. S. Haykin. *Adaptive filter theory.* Prentice Hall, 2002.

[19] S. He and M. Torkelson. A new approach to pipeline FFT processor. In *Proc. of IPPS*, pages 766–770, 1996.

[20] S. He and M. Torkelson. Design and implementation of 1024-point pipeline FFT processor. In *Proc. Custom Integr. Circuits Conf.*, pages 131–134, 1998.

[21] M. L. Honig and D. G. Messerschmitt. *Adaptive filters: structures, algorithms, and applications.* Kluwer, 1984.

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

[23] 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*, pages 417–424, 2010.

[24] H. Jiang, M. D. Riedel, and K. K. Parhi. Digital signal processing with biomolecular reactions. In *IEEE Workshop on Signal Processing Systems*, pages 237–242, 2010.

[25] P. De Kepper, I. R. Epstein, and K. Kustin. A systematically designed homogeneous oscillating reaction: the arsenite-iodate-chlorite system. *Journal of the American Chemical Society*, 103(8):2133–2134, 2008.

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

[27] R. Liu and K. K. Parhi. Minimal complexity low-latency architectures for viterbi decoders. In *Proceedings of IEEE Workshop on Signal Processing Systems (SiPS)*, pages 140–145, 2008.

[28] R. Liu and K. K. Parhi. Low-power frequency selective filtering. In *Proceedings of IEEE International Symposium on Circuits and Systems ISCAS*, pages 245–248, 2009.

[29] R. Liu and K. K. Parhi. Noise reduction for low-power broadband filtering. In *Proceedings of IEEE International Symposium on Circuits and Systems ISCAS*, pages 1012–1015, 2009.

[30] R. Liu and K. K. Parhi. Power reduction in frequency-selective fir filters under voltage overscaling. *IEEE Trans. JETCAS*, submitted.

[31] R. Liu and K. K. Parhi. Impulse noise correction in OFDM systems. *IEEE Trans. Signal Processing*, to be submitted.

[32] Y. Liu and K. K. Parhi. Computation error analysis in digital signal processing systems with overscaled supply voltage. *IEEE Trans. VLSI Sys.*, 18(4):517–526, Apr 2010.

[33] Y. Liu, T. Zhang, and K. K. Parhi. Analysis of voltage overscaled computer arithmetics in low power signal processing systems. In *Proceedings of Asilomar Conference on Signal Systems and Computers*, pages 26–29, 2008.

[34] S. Mauch. Cain: Stochastic simulations for chemical kinetics.

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

[36] D. Oh and K. K. Parhi. Low complexity design of high-speed parallel decision feedback equalizers. In *Proc. of IEEE Conference on Applications-Specific Architectures, Systems, and Processors*, pages 118–122, 2006.

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

[38] K. K. Parhi. Pipelining of parallel multiplexer loops and decision feedback equalizers. In *Proc. of IEEE ICASSP*, pages 21–24, 2004.

[39] K. K. Parhi. Design of multi-gigabit multiplexer loop based decision feedback equalizers. *IEEE Trans. VLSI Sys.*, 13(4):489–493, Apr 2005.

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

[41] W. Qian, J. Backes, and M. D. Riedel. The synthesis of stochastic circuits for nanoscale computation. *International Journal of Nanotechnology and Molecular Computation*, 1(4):39–57, 2010.

[42] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. An architecture for fault-tolerant computation with stochastic logic. *IEEE Transactions on Computers (to appear)*, 2010.

[43] W. Qian and M. D. Riedel. Synthesizing cubes to satisfy a given intersection pattern. In *International Workshop on Logic and Synthesis*, pages 217–224, 2010.

[44] W. Qian and M. D. Riedel. Synthesizing logical computation on stochastic bit streams. *submitted to Communications of the ACM*, 2010.

[45] W. Qian and M. D. Riedel. Two-level logic synthesis for probabilistic computation. In *International Workshop on Logic and Synthesis*, pages 95–102, 2010.

[46] W. Qian, M. D. Riedel, and I. Rosenberg. Uniform approximation and Bernstein polynomials with coefficients in the unit interval. Technical report, University of Minnesota, 2010. Submitted to *European Journal of Combinatorics*.

[47] K. J. Raghunath and K. K. Parhi. Parallel adaptive decision feedback equalizers. *IEEE Trans. Signal Processing*, 41(5):1956–1961, May 1993.

[48] 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*, 440:940–943, 2006.

[49] M. Sedlak and N. Ho. Production of ethanol from cellulosic biomass hydrolysate using generically engineered yeast. *Applied Biochemistry and Biotechnology*, 114(1-3):403–416, 2004.

[50] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. In *Science*, volume 314, pages 1585–1588, 2006.

[51] B. R. Sekhar and K. M. M. Prabhu. Radix-2 decimation in frequency algorithm for the computation of the real-valued FFT. *IEEE Trans. Signal Processing*, 47(4):1181–1184, Apr 1999.

[52] P. Senum and M. D. Riedel. Rate-independent biochemical computational modules. In *Proceedings of the Pacific Symposium on Biocomputing*, 2011.

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

[54] D. Soloveichik, G. Seelig, and E. Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010.

[55] S. Venkataramana, R. M. Dirks, C. T. Ueda, and N. A. Pierce. Selective cell death mediated by small conditional RNAs. *Proceedings of the National Academy of Sciences*, 2010 (in press).

[56] R. Weiss. *Cellular Computation and Communications using Engineering Genetic Regulatory Networks.* PhD thesis, MIT, 2003.

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

[58] 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*, 5(309):1–9, 2009.

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

[60] M. N. Win and C. D. Smolke. A modular and extensible RNA-based gene-regulatory platform for engineering cellular function. *Proceedings of the National Academy of Sciences*, 104(36):14283, 2007.

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