SHF: Small: Advanced Digital Signal Processing with DNA

1. Introduction

It is cliché to begin a CISE proposal with a statement of Moore's Law. And yet, the remarkable and sustained exponential pace of progress in computer performance spurs all research in the field: our mandate is to produce novel ideas to improve the technology, not in its present form, but in the form where it will be in 2, 5 or 10 years, as the exponential pace of progress continues.

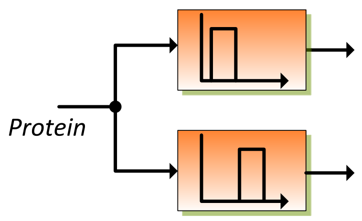


Figure 1. Activation of two pathways by two band-pass filters.

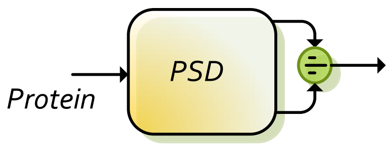


Figure 2. Protein monitoring for drug delivery.

Due to its digital nature, DNA-based computing is following a trajectory similar to silicon-based computing. Indeed, it is following a curve called the "Carlson Curve" [1], a biotechnological equivalent of Moore's Law named after Rob Carlson. The Carlson Curve predicts exponential (even *hyper-exponential*) decreases in cost and increases in performance of DNA-based technologies, such as sequencing, synthesis, and computation [2].

In a sense, the goals of this proposal are far ahead of the curve. This project will pursue the design of advanced and complex computation with DNA that is beyond the state-of-the-art and indeed beyond what current applications call for. Existing applications of molecular computing typically call for very simple computation, such as "if *condition* is true, then activate *pathway* A; else activate *pathway* B." Existing technology allows for simple circuits that implement such Boolean logic, consisting of a dozen or so biological components [3]-[13]. However, given the trajectory of the field, the technology will allow for much more complicated circuits soon. Applications, some of which we can guess, others that will emerge unexpectedly, will follow. It is both timely and opportune to consider how to design complex circuits with the new technology at the present time.

This proposal discusses techniques for implementing computation in general, and advanced digital signal processing operations in particular, using molecular reactions in general, and DNA-based reactions in particular. We consider synthesis of molecular reactions to implement signal processing functions such as finite-impulse response (FIR) and infinite impulse response (IIR) digital filters, fast Fourier transforms (FFT), and power spectral density (PSD) computations.

Whereas electronic systems perform computation in terms of voltage, i.e., energy per unit charge, molecular systems perform computation in terms of molecular concentrations, i.e., *molecules per unit volume* [14]-[23]. A particularly promising strategy for such computation is based on the mechanism of DNA strand displacement [24],[25].

The goal is no computation *per se*; molecular systems are inherently slow and inaccurate compared to electronic systems. Rather, the goal is construct the biological equivalent of *embedded controllers*: molecular systems engineered to perform useful computation *in situ*, where it is needed, for instance for drug delivery and for monitoring the effectiveness of drug therapy. Consider some potential application scenarios. Figure 1 illustrates the filtering of a time-varying concentration of a protein through two different frequency bands. The result is either to activate or to inhibit different pathways. The protein may be a biomarker for cancer. The result may be the activation of a pathway to produce a chemo drug. In a more complicated scenario, a *band-pass* filtered signal may activate or inhibit a pathway. In a still more complex scenario, the *ratio of spectral power in two different bands* might be the requisite trigger.

Such scenarios, although hypothetical, are motivated by exciting recent research [26], [27]. In our own recent work [28]-[30], we showed that a ratio of spectral power computed from an electroencephalogram (EEG) electrode can predict seizures up to an hour before it happens. We have also identified ratios of spectral power from magneto-encephalograms (MEGs) that can be used as biomarkers for schizophrenia [31],[32]. These ratios could be computed using a ratio of the two filtered outputs shown in Figure 1; however, computing such a ratio using power spectral density as shown in Figure 2 reduces the computation complexity significantly. We recently proposed a low-power architecture for computing the power spectral density (PSD) in electronic systems [33]. The Fast Fourier Transform (FFT) is the basic building block in the PSD architecture. One of the goals of this project is to demonstrate how to compute a power spectral density and a spectral ratio using molecular reactions.

We distinguish between discrete-time signal processing and digital signal processing. While signals are sampled periodically in both systems, the signal is represented as an analog value in the former while the signal is quantized to a digital value in the latter. Each has its advantages. Discrete-time signal processing systems are similar to sampled data systems and require lower molecular concentrations; however, the resolution cannot be precisely controlled. Digital systems are more precise, but require higher molecular concentrations, just as digital numbers in electronic systems require multiple bits. A major component of this project is to study how to implement analog-to-digital (A/D) and digital-to-analog (D/A) conversion with molecular reactions. Such A/D and D/A conversion have not been explored before by the molecular computing community. We propose to investigate and implement both discrete-time and digital signal processing systems using DNA strand displacement as the target experimental chassis [24], [25].

As principal investigators (Parhi and Riedel), we began working on digital signal processing using molecular reactions five years ago. Our efforts have been funded by two NSF grants: an EAGER grant (CCF 0946601) during 2009-2011 and another regular grant CCF-1117168 that started in 2011. These two grants have allowed us to prove, for the first time, that digital signal processing and sequential computations can be implemented using molecular reactions [34]-[38]. These results are significant because, unlike electronic systems, molecular computing is inherently asynchronous and parallel: when reactants are present, reactions fire at variable rates. Our work demonstrated that, through molecular transfer reactions, delay elements can be implemented in a robust manner independent of the number of delay elements. We were the first to present synchronized sequential computation with delay elements. Our work also demonstrated that, using new *bistable* molecular reactions based on approaches similar to dual-rail logic in electronic circuits, we can implement robust logic gates such as NAND and NOR [29]. We also demonstrated the implementation of D flip-flops, and computations such as binary addition, linear feedback shift registers and square-root using molecular reactions [39]. This proposal will build on the success of our prior and current work and will explore implementation of complex signal processing functions for both discrete-time and digital signal processing applications.

The proposed effort will extend our prior work in two innovative and new directions. First, a complete digital signal processing systems implementation will be demonstrated. Such a system will contain A/D and D/A converters. Detailed studies of the properties of such systems will be performed, e.g., how the resolution correlates with changing molecular concentrations and how robust the designs are to parametric variations. These tradeoffs have not been explored before. Second, the project will develop faster implementations of discrete-time as well as digital signal processing systems. The main bottleneck in prior discrete-time signal processing implementations has been speed. Unlike in electronic systems, where the speed is limited by changes in electric charge, the speed in molecular systems is limited by changes in molecular concentrations, which are inherently slow. We propose new scheduling approaches where *multiple computations* are mapped into different phases of transfer. This is based on generalization of our prior schemes such as the Red-Green-Blue (RGB) scheme and our synchronous scheme, based on sustained chemical oscillations. The proposed new scheduling approaches allow computation of *parallel outputs* without increasing the number of delay transfer reactions. We expect to demonstrate that this approach can increase the overall sample speed compared to our current and prior work. Reducing currently achievable sample periods from 40-80 hours to 4-8 hours will enable experimental demonstration of some example signal processing functions using DNA. Furthermore, we will investigate tradeoffs in discrete-time and digital implementations of signal processing functions with respect to speed, accuracy, and robustness.

2. Prior work

In this section, we describe our prior work on digital logic implementations and discrete-time signal processing using molecular reactions and simulation using DNA. In Section 2.1, we briefly describe our novel *bistable* mechanism to implement digital logic gates. These reactions were used to implement other complex functions such as D flip-flops, binary adders, square-root units and linear feedback shift registers. The details of these implementations have been described in [39]. The main advantage of the proposed reactions is robustness. In Section 2.2, we describe implementation of delay elements and signal transfer through delay elements using the RGB scheme and the synchronous scheme.

2.1. Digital Logic using Molecular Reactions

The most straightforward interpretation of binary values in the context of molecular computation is to assign a threshold to the concentration of a designated molecular type [40]. When the concentration exceeds a threshold level, the bit is considered a logical 1; otherwise it is considered a logical 0. Although such a representation is conceptually simple, it requires external mechanisms for comparing the concentration of the designated molecular type with the threshold. Furthermore, it suffers from signal degradation over time: unwanted residue accumulates every time a signal value changes, unless there is some mechanism to clear the signal. To mitigate these issues, we use a complementary representation (reminiscent of a “dual-rail” encoding). For a single bit X, we use two molecular types, X_0 and X_1. The presence of X_0 indicates that X is set to 0; the presence of X_1 indicates that X is set to 1. Clearly, X_0 and X_1 should not be present at the same time or else the value of X would be ambiguous. We use following set of reactions to ensure that this does not happen:

$$X_0 + X_1 \rightarrow S_X, \qquad S_X + X_0 \rightarrow 3X_0, \qquad S_X + X_1 \rightarrow 3X_1 \tag{1}$$

In Reactions (1), a molecule of X_0 combines with a molecule of X_1 to produce a molecule of S_X. This molecule of S_X then combines with a molecule of X_0 or X_1, depending on which it meets first. The choice is competitive: both X_0 and X_1 are trying to increase their concentration via the intermediary type S_X; whichever has a higher concentration wins. The concentration of the loser effectively drops to zero. So this mechanism clears out the leakage of molecular types that would otherwise occur when bits are set. We can map Reactions (1) to DNA strand displacements. This bistability forms the basis of our representation of a bit. The rate kinetics of reactions (1) have been studied in our publication [39].

Given this robust representation of binary bits, we demonstrate how to implement logic gates with molecular reactions. We only consider two-input gates; gates with more than two inputs can be easily implemented by cascading two-input gates. Suppose the inputs of a gate are X and Y, and the output is Z. These signals are represented by the concentrations of X_0/X_1, Y_0/Y_1, and Z_0/Z_1, respectively. Each one of X, Y , and Z is regulated by its own version of the bit operation reactions:

$$\begin{gathered} X_0 + X_1 \rightarrow S_X, \qquad S_X + X_0 \rightarrow 3X_0, \qquad S_X + X_1 \rightarrow 3X_1 \\ Y_0 + Y_1 \rightarrow S_Y, \qquad S_Y + Y_0 \rightarrow 3Y_0, \qquad S_Y + Y_1 \rightarrow 3Y_1 \\ Z_0 + Z_1 \rightarrow S_Z, \qquad S_Z + Z_0 \rightarrow 3Z_0, \qquad S_Z + Z_1 \rightarrow 3Z_1 \end{gathered} \tag{2}$$

For each of the four entries in the truth table for the gate, if the value of Z is 1, then molecules of Z_0, if any, should be transferred to Z_1. Similarly, if the value of Z is 0, then molecules of Z_1, if any, should be transferred to Z_0. Let us first consider an AND gate. By definition, either X = 0 or Y = 0 sets Z to 0, which means that when either X_0 or Y_0 is present, Z_0 should be generated and Z_1 should be cleared out. This is implemented by the reactions

$$X_0 + Z_1 \rightarrow X_0 + Z_0, \qquad Y_0 + Z_1 \rightarrow Y_0 + Z_0 \tag{3}$$

Here, X_0 and Y_0 transfer Z_1 to Z_0 but keep their own concentrations unchanged. Z is set to 0 if it has not already been. Z should be set to 1 only when both X = 1 and Y = 1. This is implemented by the reactions

$$X_1 + Y_1 \rightarrow X_1 + Y_1 + Z'_1 \quad , \quad 2\,Z'_1 \rightarrow src \quad , \quad Z_0 + Z'_1 \rightarrow Z_1 \tag{4}$$

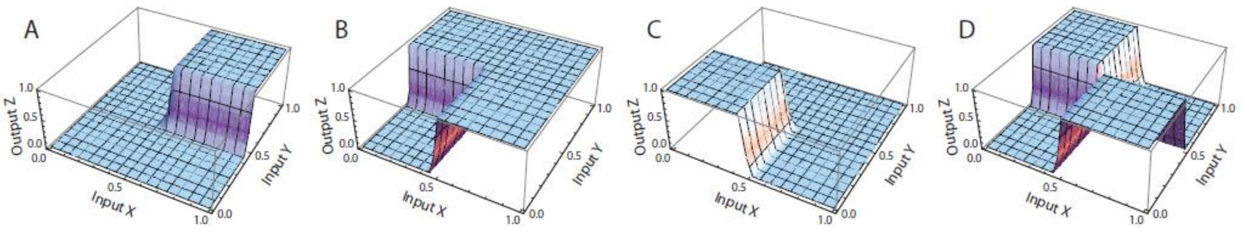


Figure 3. DNA-level simulation results for the logic gates. Concentration values are normalized.

To simulate the AND gate, we map Reactions (2), (3) and (4) to DNA strand-displacement reactions and generate their corresponding ODEs. The results are shown in Figure 3A.The idea can be used for implementing OR, NOR, and XOR gates [39]. Reactions in (2) are required for all gates. The special molecular reactions for implementing these gates are tabulated in Figure 4. Figure 3 B, C, and D, respectively, show the simulation results for OR, NOR, and XOR gates. Any random combinational logic can be implemented either by using these primitive gates or by direct synthesis.

OR	NOR	XOR
$X_1 + Z_0 \rightarrow X_1 + Z_1$	$X_1 + Z_1 \rightarrow X_1 + Z_0$	$X_0 + Y_1 \rightarrow X_0 + Y_1 + Z'_1$
$Y_1 + Z_0 \rightarrow Y_1 + Z_1$	$Y_1 + Z_1 \rightarrow Y_1 + Z_0$	$X_1 + Y_0 \rightarrow X_1 + Y_0 + Z'_1$
$X_0 + Y_0 \rightarrow X_0 + Y_0 + Z'_0$	$X_0 + Y_0 \rightarrow X_0 + Y_0 + Z'_1$	$2Z'_1 \rightarrow src$
$2Z'_0 \rightarrow src$	$2Z'_1 \rightarrow src$	$Z'_1 + Z_0 \rightarrow Z_1$
$Z'_0 + Z_1 \rightarrow Z_0$	$Z'_1 + Z_0 \rightarrow Z_1$	$X_0 + Y_0 \rightarrow X_0 + Y_0 + Z'_0$
		$X_1 + Y_1 \rightarrow X_1 + Y_1 + Z'_0$
		$2Z'_0 \rightarrow src$
		$Z'_0 + Z_1 \rightarrow Z_0$

Figure 4. Molecular reactions for OR, NOR, and XOR gates.

2.2. Discrete-time Signal Processing

A DSP system is composed of two parts: *computation* and *memory*. The computation part executes arithmetic operations such as addition and multiplication, whereas the memory part consists of delay elements that store signals and transfer these signals to the output of the delay element in subsequent cycles. Our prior work on molecular DSP has demonstrated methods for implementing DSP algorithms using *synchronous* and *RGB* schemes [38]. In both schemes the computation part is implemented by the same set of reactions. The difference between these schemes lies in the implementation of the delay elements and transfer reactions.

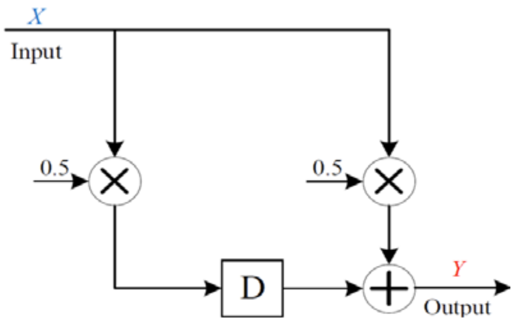


Figure 5. Block diagram for the moving average filter.

One time execution of each computation of a signal processing algorithm is referred as an iteration or a computation cycle. In prior work, we have developed an RGB scheme and a synchronous scheme. These two schemes differ in how an iteration of computation is completed. In the RGB scheme, three types of proteins (R, G, and B) are used. Their absence indicators r, g, and b are used to represent absence of a certain protein. Transfer of R to G enabled by the absence indicator b (absence of B), transfer of G to B enabled by absence indicator r, and transfer of B to R enabled by absence indicator g complete an iteration of computation. In contrast, in a synchronous system, a two-phase clock is used to complete all computations and transfer reactions associated with each iteration.

We have demonstrated implementation of discrete-time finite-impulse response (FIR), and infinite impulse response (IIR) digital filters and fast Fourier transforms (FFTs) using RGB and synchronous schemes. We have shown that synchronous schemes take longer time to complete an iteration but lead to better accuracy while RGB schemes are faster but compute outputs with less accurate results. Although details of all prior implementations cannot be included here due to lack of space, we illustrate both schemes using a simple example: the moving-average filter. The block diagram for the filter is shown in Figure 5. It produces an output value that is sum of one-half the current input value and one-half of the previous value. Given a time-varying input signal, X, the output signal, Y, is a moving average, i.e., a smoother version of the input signal.

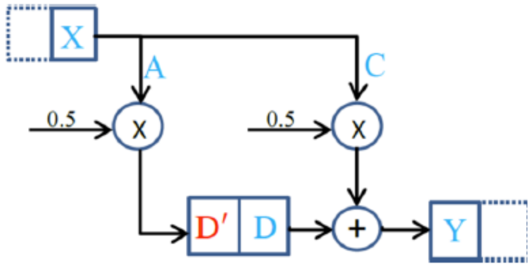


Figure 6. The synchronous implementation for the moving average filter.

2.2.1. Synchronous scheme

Similar to electronic systems, the synchronous scheme has a two-phase clock. The clock signal synchronizes transfers signals between *computation* and *memory* parts. Each delay element in this scheme consists of two molecular types, D and D'. The realization of the moving-average filter in the synchronous framework is shown in Figure 6. We assume that a two-phase clock generates and consumes the molecular types *R* and *B* in alternating fashion. In other words we color-code first and second phase of the clock as *R*(ed) and *B*(lue). Figure 7 shows the concentrations of *R* and *B* as a function of time, for our proposed two-phase clock. Details regarding how to generate an n-phase clock are given later in this section.

Set of molecular reactions for the synchronous implementation of the moving-average filter are given in (5). In the presence of *B*, the input signal X is transferred to molecular types A and C; these are both reduced to half and transferred to D′ and Y, respectively. In the presence of *R*, D′ is transferred to D.

$$\begin{aligned}
&\textit{Phase 1:} \quad && X + B \xrightarrow{k_s} A + C + B \\
& && 2A \xrightarrow{k_f} D' \\
& && 2C \xrightarrow{k_f} Y \qquad\qquad (5)\\
& && D + B \xrightarrow{k_s} Y + B \\
&\textit{Phase 2:} \quad && D' + R \xrightarrow{k_s} D + R
\end{aligned}$$

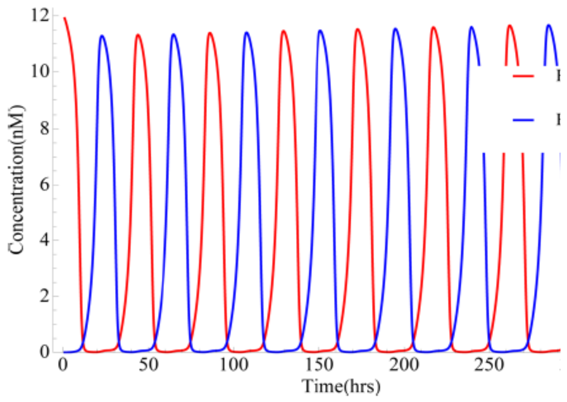


Figure 7. Differential equation simulation of the chemical kinetics of the 2-phase clock. In principle, the amplitude and frequency of oscillations can be controlled.

For a general DSP system, in the first phase of the clock, the computation results are stored in D'_i and in the second phase the concentration of D'_i transfers to D_i. To validate our designs for the moving-average filter, the chemical reactions were mapped to DNA strand displacement reactions, using the method in [25], and their kinetic differential equations were simulated.

The simulation results for the moving-average filter are shown in Figure 8. The input is a time-varying signal concentration X with both high-frequency and low-frequency components. The output is a time varying signal concentration Y. Molecules of X are injected and molecules of Y are collected from the system every 80 hours. The figure shows the theoretical output, *i.e.*, an exact calculation of filtering, as well as the simulated output. We see that our design performs very well, filtering out the high-frequency component as expected. The simulated output concentration does not quite track the theoretical output concentration. The explanation for this is that, due to the small overlap of the clock phases, the next phase begins before the reactions in the current phase complete. To decrease the overlap between two phases of the clock, we can add extra phases and choose two phases in a way to have a symmetric two-phase clock. For example we can use a 6-phase design and choose phases 3 and 6 of a six-phase oscillating pulse as clock signals. Although such an approach is likely to enhance output accuracy, the

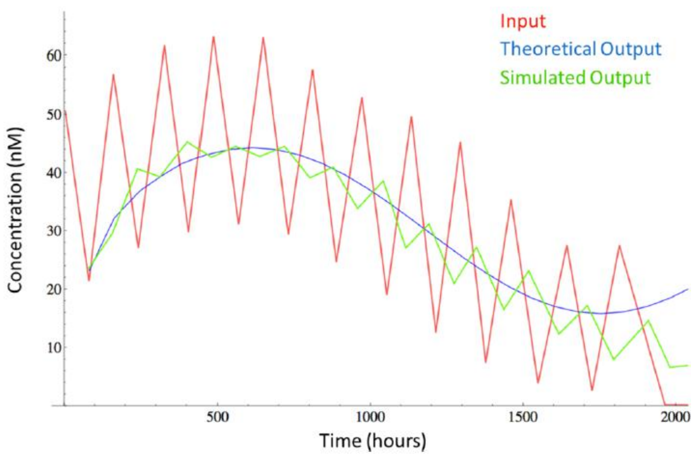


Figure 8. DNA-level simulation of the moving average filter implemented using synchronous scheme.

sample period would be substantially increased as more transfer reactions need to be completed within the iteration. In electronic circuits, a clock signal is generated by an oscillatory circuit that produces periodic voltage pulses. For a molecular clock, we choose reactions that produce sustained oscillations in terms of chemical concentrations. With such oscillations, a low concentration corresponds to a logical value of zero; a high concentration corresponds to a logical value of '1'.

Techniques for generating chemical oscillations are well established in the literature. Classic examples include the Lotka-Volterra, the "Brusselator", and the Arsenite-Iodate-Chlorite systems [41], [42]. Unfortunately, none of these schemes are quite suitable for synchronous sequential computation: we require that the clock signal be symmetrical, with abrupt transitions between the phases. Here, we present a new design for an n-phase chemical oscillator (n ≥ 3). The clock phases are represented by molecular types $P_1, P_2, \ldots, P_n$. First consider the Reactions in (6) and (7).

$$2S_1 \overset{k_s}{\rightarrow} a_1 + 2S_1\ ,\quad 2S_2 \overset{k_s}{\rightarrow} a_2 + 2S_2\ ,\ \ldots\ ,\quad 2S_n \overset{k_s}{\rightarrow} a_n + 2S_n \tag{6}$$

$$P_1 + a_1 \overset{k_f}{\rightarrow} P_1\ ,\quad P_2 + a_2 \overset{k_f}{\rightarrow} P_2\ ,\ \ldots\ ,\ P_n + a_n \overset{k_f}{\rightarrow} P_n \tag{7}$$

In reactions (6), the molecular types $a_1, a_2, \ldots, a_n$are generated slowly and constantly, from source types $S_1, S_2, \ldots, S_n$ whose concentrations do not change with the reactions. Here, all reactions are expressly designed to have two reactants; this permits us to map the reaction to DNA strand displacement reactions effectively. In reactions (7), the types $P_1, P_2, \ldots, P_n$ quickly consume the types $a_1, a_2, \ldots, a_n$, respectively. Call $P_1, P_2, \ldots, P_n$ the phase signals and $a_1, a_2, \ldots, a_n$ the absence indicators. The latter are only present in the absence of the former. The reactions

$$P_1 + a_n \overset{k_s}{\rightarrow} P_2\ ,\quad P_2 + a_1 \overset{k_s}{\rightarrow} P_3\ ,\ \ldots\ ,\ P_n + a_{n-1} \overset{k_f}{\rightarrow} P_2 \tag{8}$$

transfer one phase signal to another, in the absence of the previous one. The essential aspect is that, within the $P_1, P_2, \ldots, P_n$ sequence, the full quantity of the preceding type is transferred to the current type before the transfer to the succeeding type begins. To achieve sustained oscillation, we introduce positive feedback. This is provided by the reactions:

$$\begin{aligned} &2P_1 \overset{k_s}{\rightarrow} I_1\ , \quad 2I_1 \overset{k_f}{\rightarrow} 4P_1\ , \quad P_n + I_1 \overset{k_f}{\rightarrow} 3P_1 \\ &2P_2 \overset{k_s}{\rightarrow} I_2\ , \quad 2I_2 \overset{k_f}{\rightarrow} 4P_2\ , \quad P_1 + I_2 \overset{k_f}{\rightarrow} 3P_2 \\ &\qquad\qquad\vdots \\ &2P_n \overset{k_s}{\rightarrow} I_n\ , \quad 2I_n \overset{k_f}{\rightarrow} 4P_n\ , \quad P_{n-1} + I_n \overset{k_f}{\rightarrow} 3P_n \end{aligned} \tag{9}$$

Consider the first three reactions. Two molecules of P_1 combine with one molecule of P_n to produce three molecules of P_1. The first step in this process is reversible: two molecules of P_1can combine, but in the absence of any molecules of P_n, the combined form will dissociate back into P_1. So, in the absence of P_n, the quantity of P_1will not change much. In the presence of P_n, the sequence of reactions will proceed, producing one molecule of P_1 for each molecule of P_n that is consumed. Due to the first reaction, the transfer will occur at a rate that is superlinear in the quantity of P_1; this speeds up the transfer and so provides positive feedback. Suppose that the initial quantity of P_1 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 P_1, P_2, ..., P_n. One requirement for a clock in synchronous computation is that different clock phases should not overlap. In a two-phase clock for synchronous structures: concentrations of molecular types representing clock phase “0” and clock phase “1” should not be present at the same time. To this end, we choose two nonadjacent phases, P_1 and P_3 in a four-phase oscillator, as the clock phases. For a clearer illustration, we use *R*(ed) to denote P_1 and *B*(lue) to denote P_3.Our scheme for chemical oscillation works well.

2.2.2. RGB scheme

Unlike the synchronous scheme the *RGB* scheme does not require a global clock; rather it is “self-timed” in the sense that a new phase of the computation begins when an external sink removes the entire quantity of molecules Y, *i.e.*, the previous output value, and supplies a new quantity of molecules X, *i.e.*, the current input value.

The moving-average filter in this framework is shown in Figure 9 and can be implemented by the reactions in Figure 10. The molecular types corresponding to signals are X, A, C, R, G, B, and Y. To illustrate the design, we use colors to categorize some of these types into three categories: Y and R in red; G in green; and X and B in blue. The group of the first three reactions shown in the S1 column of Figure 10 transfers the concentration of X to A and to C, a fanout operation. The concentrations of A and C are both reduced to half, scalar multiplication operations. The concentration of A is transferred to the output Y, and the concentration of C is transferred to R. The transfer to R is the first phase of a delay operation. Once the signal has moved through the delay operation, the concentration of B is transferred to the output Y. Since this concentration is combined with the concentration of Y produced from A, this is an addition operation. The final group of three reactions shown in the S1 column of Figure 10 implements the delay operation. The concentration of R is transferred to G and then to B. Transfers between two color categories are enabled by the absence of the third category: red goes to green in the absence of blue; green goes to blue in the absence of red; and blue goes to red in the absence of green. The reactions are enabled by molecular types r, g, and b that we call absence indicators. The absence indicators ensure that the delay element takes a new value only when it has finished processing the previous value.

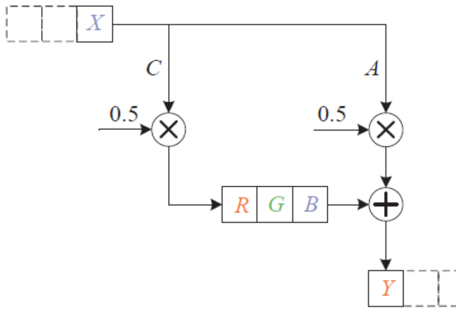


Figure 9. The moving average filter implemented in *RGB* scheme.

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

Figure 10. Set of molecular reactions for the *RGB* implementation of the moving average filter.

The simulation result for the RGB implementation of moving average filter is shown in Figure 11. Like the synchronous scheme the input signal contains both high frequency and low frequency components. Molecules X are injected into the system and molecules Y are collected from the system every 20 hours. We see that our design performs well, attenuating the high-frequency component. The simulated output concentration does not quite track the theoretical output concentration; it is higher than it should be for high input concentrations. The explanation for this is that, for high input concentrations, the reactions fire quickly, so the computational cycle completes early. Before the next cycle begins, some "leakage" of the output concentration occurs.

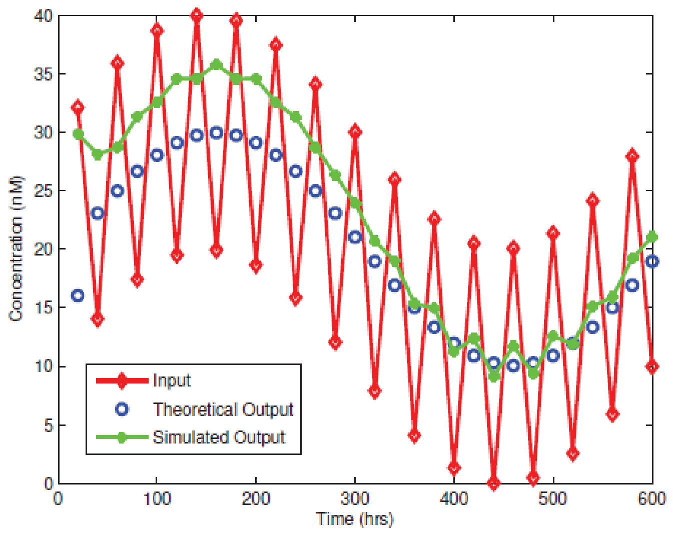


Figure 11. DNA-level simulation of the moving average filter implemented in *RGB* scheme.

2.2.3. More Complex Applications

Based on the previous frameworks we investigated the biomolecular implementation of two more complex applications: biquad filter and fast Fourier transform (FFT). Due to lack of space in the proposal the details of these implementations are not included. The details of these reactions have been presented in [34] and [38]. The biquad filter is an infinite-impulse response filter and contains feedback. The FFT computes the Fourier-domain representation of a time-domain signal. Although the computations are feed-forward, this requires molecular implementation of switches or multiplexors.

Table 1 compares the simulation results of the three operations,namely, the moving-average filter, the biquad filter, and the real FFT (RFFT) transform, in both the synchronous and *RGB* frameworks. The real FFT computes the FFT of a real signal. The error in this table is computed as the difference between the output value obtained by simulation, O_s, and the theoretical output, O_t. Table 1 shows that the synchronous scheme has lower error. On the other hand, it is slower than the *RGB* scheme. Although the in-vitro simulation results using DNA strands validate the functionality of the methods, it is essential to improve their speed and robustness. Proposed efforts to overcome these barriers are described in next section.

Table 1. Comparison for moving average, biquad, and RFFT in synchronous and RGB schemes.

system	scheme	no. of reactants	no. of reactions	Sample period (hrs)	% error $(\left\lvert\frac{O_s-O_t}{O_t}\right\rvert \times 100$
moving average	Sync.	22	29	80	4.21
	RGB	16	24	20	5.67
biquad	Sync.	37	44	80	8.63
	RGB	32	46	50	12.79
RFFT	Sync.	119	202	900	7.8
	RGB	213	225	425	22

3. Proposed work

Prior work in our group has provided a framework for synthesizing DSP operations with molecular reactions. The proposed project will build upon our prior work and will explore novel methodologies to improve robustness, accuracy and speed of the molecular DSP systems. First, digital implementations of signal processing functions are proposed. These systems require A/D and D/A conversions. Molecular implementations of these systems have never been attempted before. Second, new scheduling approaches are presented to reduce the sample period in discrete-time signal processing systems.

3.1. Fully-Digital Signal Processing

Designing a robust DSP system with biomolecular reactions is very challenging. For the discrete-time systems described in previous sections, the speed and output error vary depending on the input and intermediate signal values. This is because the reaction rates are proportional to the concentration of the participating molecular types. To improve robustness, we propose to develop a framework to implement DSP operations in a fully-digital manner. Figure 12 illustrates the general structure of our design. Proposed objectives include: design and implementation of robust and fast arithmetic operations in this framework, analog to digital converter (ADC), and digital to analog converter (DAC). Based on our preliminary work, we present a proof of concept for molecular implementation of these building blocks. It may be noted that the molecular reactions are implemented using a clock generated using a synchronous scheme.

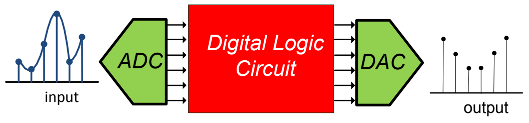


Figure 12. Proposed digital Framework for biomolecular discrete-time signal processing.

3.1.1. DAC and ADC

Digital to Analog converter (DAC). Figure 13 illustrates the proposed DAC circuit for a 3-bit data. The required arithmetic operations (addition and multiplication) for this circuit are well defined in prior work [24]. The conversion must be fired to completion when the clock is nonzero. The output value is collected when the clock is zero. Reactions in (10) represent an abstract implementation of DAC.

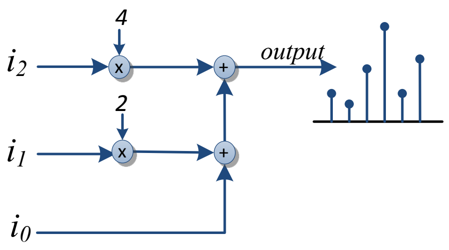


Figure 13 . Proposed DAC circuit.

$$i_0 + clk \xrightarrow{k_f} clk + output \ , \quad i_1 + clk \xrightarrow{k_f} clk + 2\,output \ , \quad i_2 + clk \xrightarrow{k_f} clk + 4\,output \qquad (10)$$

Analog to digital converter (ADC). For the molecular ADC we apply the idea of successive-approximation-register (SAR) [43] used in electronic ADCs and adapt it in a way that can be efficiently implemented by molecular reactions. Compared to other ADCs, SAR has some advantages for electronic as well as molecular implementations. First, this approach does not require a counter. Second, the time to convert an analog value to digital is independent of the signal value. Figure 14 shows the proposed ADC in detail. In this figure *SR* is a shift register with the initial value of “100”. *R* is a register initialized by “000”. The DAC inputs are connected to the bit-wise addition of *SR* and *R*. During the first clock pulse the output of the DAC, or the inverting input of the comparator, is 4. Thus, the sample of a discrete-time input signal is compared with 4 and if it is greater than 4, the comparator’s output is set to ‘1’. In this case the nonzero value of SR_2 (bit 2 of register *SR*) is copied to R_2 (bit 2 of register *R*) and sets its value to ‘1’. Otherwise, if the DAC output is less than 4, R_2 remains ‘0’. Transfer of SR_2 to R_2 is implemented by the following chemical reaction:

$$SR_2 + En \rightarrow SR_2 + En + R_2 \qquad (11)$$

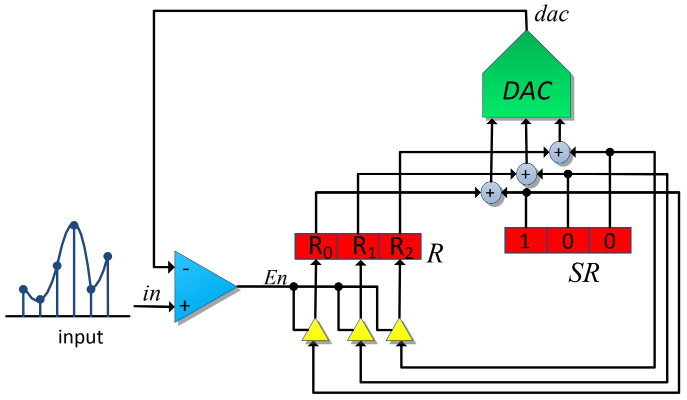


Figure 14 . Proposed ADC circuit.

In reaction (11), En corresponds to an enable signal. At the next clock pulse the content of the *SR* changes to “010”. Then the DAC input is “010” or “110” depending on whether R_2 is ‘0’ or ‘1’, respectively. Therefore, the analog input signal is compared to 2 or 6 depending on the R_2 value. If the value of input signal is greater than ‘1’ SR_1 is copied to R_1. Otherwise R_1 remains ‘0’. Finally, in the same manner, the value of R_0 is determined during the third clock pulse. Therefore, the 3-bit digital equivalent for the current sample of the discrete time input is available as "$R_2R_1R_0$". Note that for each new input sample, *R* must be reset at the beginning of the conversion.

Each element in the ADC circuit will be implemented by molecular reactions targeting the DNA-strand displacement reactions. We will use a clock similar to the two-phase synchronous clock. Register, *R*, and shift register, *SR*, can be implemented using D flip-flop. In prior work, we have demonstrated implementation of D flip-flops by molecular reactions [39].

For the comparator, based on our prior work [44], we develop a construct that compares the quantities of two input types and produces an output type if one is greater than the other. We now describe how this can be achieved. First we sample the input signal, *in*, and the DAC output signal, *dac*, in Figure 14 using the reactions in (12). The sampling reactions should be fast enough to complete by the current clock phase.

$$clk + in \xrightarrow{k_f} clk + S_{in} \quad , \qquad clk + dac \xrightarrow{k_f} clk + S_{dac} \tag{12}$$

Then we compare the samples S_{in} and S_{dac} by consuming them via the following reaction:

$$S_{in} + S_{dac} \xrightarrow{k_f} src. \tag{13}$$

We assume that the reaction fires to completion. The result is that there are only molecules of S_{in} left, or only molecules of S_{dac} left, or no molecules of S_{in} or S_{dac} left. Molecules of the type that originally had a larger quantity will persist. If the quantities are equal, then both types are annihilated. In order to determine which type is completely annihilated, we need to generate the absence indicators a_{in} and a_{dac}, respectively, for samples S_{in} and S_{dac}.

$$src \xrightarrow{k_s} a_{in} + a_{dac} \quad , \; S_{in} + a_{in} \xrightarrow{k_f} S_{in} \;\; , \quad S_{dac} + a_{dac} \xrightarrow{k_f} S_{dac} \tag{14}$$

Reaction (15) produces the molecule *En* if the sampled value from the input is greater than the sampled value from the DAC's output.

$$S_{in} + a_{dac} \xrightarrow{k_s} S_{in} + a_{dac} + En \tag{15}$$

Finally, for robustness and to enable start of the next comparison, we use the following reaction to destroy En in the case that the asserted condition is not true:

$$S_{dac} + En \xrightarrow{k_s} S_{dac} \quad , \qquad a_{in} + En \xrightarrow{k_s} a_{in}. \tag{16}$$

For the central part of our structure, we use a very robust molecular implementation of logical circuits based on the approach discussed in Section 2.1 for binary representation. One should note that the proposed complementary bit representation can be easily applied to the ADC and DAC parts.

3.1.2. Digital Filter design. Constructs for ADC, digital logic, and DAC will be used in combination with a synchronous clock to demonstrate a digital filter implementation. A simple 3-tap FIR filter is shown Figure 15. The main components of the digital filter are delay units, adders, and constant coefficient multiplication. Here, delay elements are *W*-bit registers where *W* is the number of bits, also referred to as word-length. A *W*-bit register consists of *W* D flip-flops. Therefore, delays and adders can be implemented based on constructs from our prior work.

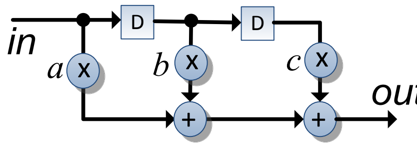


Figure 15. A 3-tap FIR filter.

Consider the number $A = a_{W-1}.a_{W-2} \ldots a_1\, a_0$ as the constant coefficient in the FIR filter. Now the multiplication $A \times x$, where x is a variable data can be expressed as $A \times x = \sum_{i=0}^{W-1} a_{W-1-i} x 2^{-i}$. It is clear that constant multiplication can be carried out by adding a number of partial product terms corresponding to the nonzero bit positions in the constant multiplier. The number of add operations required equals one less than the number of nonzero bits in the constant coefficient. The constant coefficient can be encoded such that it contains the fewest number of nonzero bits. This can be accomplished using canonic signed digit (CSD). In CSD representation each bit

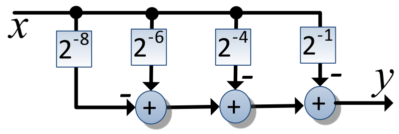


Figure 16. CSD Implementation of $x \times 0.\bar{1}00\bar{1}010\bar{1}.$

is in the set {-1,0,1} instead of {0,1} for common 2's complement representation. In fact, the multiplication for CSD multiplier is calculated by adding or subtracting partial product terms. The properties of CSD representation, an algorithm for computing the CSD format of a number and how to convert a 2's complement representation to CSD are described in [45]. As an example for the number 1.01110011 in 2's complement format, the CSD representation is $0.\bar{1}00\bar{1}010\bar{1}$. (Here $\bar{1}$ denotes the bit value of -1). In this case the number of nonzero bits is reduced from 6 to 4. The multiplication $y = x \times 0.\bar{1}00\bar{1}010\bar{1}$ can be computed as illustrated in Figure 16. We can apply the Horner's rule for precision improvement and tree-height reduction technique for latency reduction [46]. The improved CSD multiplier is shown in Figure 17. Molecular implementation for other types of multipliers such as Baugh-Wooly [46] can be investigated.

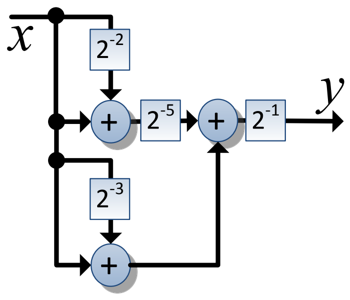


Figure 17. Improved circuit for implementation of $x \times 0.\bar{1}00\bar{1}010\bar{1}$.

While these filter optimizations are well known in electronic circuits, the impact of these tradeoffs in molecular implementations has not been explored. Various optimization approaches will be implemented in DNA and their tradeoffs with respect to speed and accuracy will be investigated.

Adaptive digital filters cannot be implemented using fixed coefficients. In these systems, the coefficients are adapted. Thus, the multipliers need to be programmable. Various programmable multipliers such as modified Booth recoded Wallace tree have been used in electronic implementations. We propose to simulate these multipliers using molecular reactions for implementing an adaptive digital filter.

3.2. High-Speed Discrete-time Systems using DNA

The proposed work will investigate implementations of parallel signal processing systems using DNA to increase achievable sample rates. We argue that computation of multiple outputs in parallel will reduce the sample period since the number of delays does not change in these computations and the completion time for the transfer reactions stays about the same. Since computations are fast, the effective sample period can be reduced as the level of parallelism increases. This, however, increases the number of reactions. The basic approach involves mapping computation of different outputs to various phases of the computation cycle. Multiple outputs can be computed in parallel either using the RGB scheme or a synchronous scheme. We illustrate computation of the parallel FIR filter using the RGB scheme. We also propose a new 4-phase scheme to implement discrete-time signal processing systems. Then we describe how parallel outputs can be computed using the proposed 4-phase scheme.

3.2.1. Fast DSP using a 4-Phase Scheme

We illustrate a new 4-phase clock-free scheme where we implement delay elements using two molecular types. A synthesis approach for mapping any DSP algorithm to molecular reactions in the 4-phase scheme is described below:

1- Draw the data flow graph (DFG) according to the block diagram of the DSP algorithm. Replace the input node x by nodes x and x', output node y by nodes y and y', and each delay element D_k by a pair of nodes D_k and D'_k.

2- Assign phase 1 to the outgoing edges of the node x', y' and the outgoing edges of each D'_k node.

3- Assign phase 3 to the fan out edges of the nodes x, y and All edges between D_k and D'_k.

4- Consider additional nodes m_2 and m_4. m_2 transfers to m_4 and m_4 transfers to m_2 at phases 2 and 4, respectively.

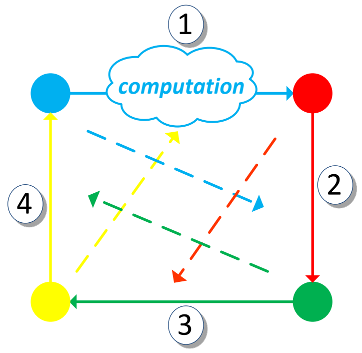


Figure 18. 4-phase scheduling.

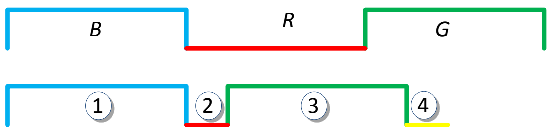
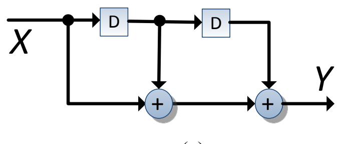


Figure 19. RGB and 4-phase timing.

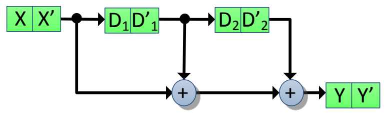
5- The molecular reactions for absence indicators, computations, and signal transfers are synthesized according to the assigned scheduling phases.

It noticeable that in an asynchronous scheme, edges of two consecutive phases cannot be directly connected [37]. Here phases 2 and 4 are introduced to separate phases 1 and 3. Figure 18 illustrates the a 4-phase scheduling scheme. Signal transfers in each phase are triggered by the absence indicators (dashed arrows) of the previous phase. The concept for absence indicators is described in [37]. The concentrations for m_2 and m_4 are independent of the input signal value. Therefore, they can be very small. This is *key* to faster phases 2 and 4. However, in the *RGB* scheme the complete signal value transfers in each phase. Thus, none of the phases can be shrunk. Figure 19 illustrates how a computing iteration in the proposed 4-phase scheme can be faster compared to the *RGB* scheme. We illustrate the synthesis method for an FIR filter.

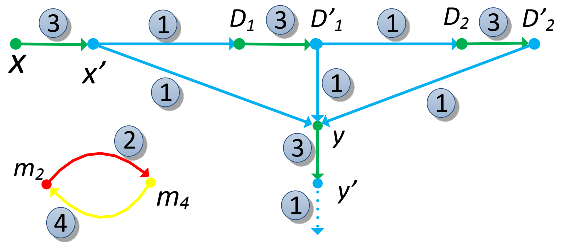
FIR filter. Figure 20(a) shows a three-tap FIR filter. For simplicity, all tap coefficients are assumed to be 1. The flow graph in Figure 20(b) illustrates the phase assignments. Reactions in (17) provide absence indicators for the flow graph. First reaction in (17) slowly generates the absence indicators for each phase. In the remaining reactions of (17), the source molecules of signals are transferred in each phase quickly, and the absence indicator of that phase is consumed.



(a)



(b)



(c)

Figure 20. A three-tap FIR filter: (a) Block diagram, (b) 4-phase scheme (c) Data flow graph and scheduling based on the proposed method.

$$src \overset{k_s}{\rightarrow} a_1 + a_2 + a_3 + a_4$$

Phase 1:
$$a_1 + x' \overset{k_f}{\rightarrow} x'$$
$$a_1 + D'_1 \overset{k_f}{\rightarrow} D'_1$$
$$a_1 + D'_2 \overset{k_f}{\rightarrow} D'_2$$
$$a_1 + y' \overset{k_f}{\rightarrow} y'$$

Phase 2:
$$a_2 + m_2 \overset{k_f}{\rightarrow} m_2$$

Phase 3:
$$a_3 + D_1 \overset{k_f}{\rightarrow} D_1$$
$$a_3 + D_2 \overset{k_f}{\rightarrow} D_2$$
$$a_3 + x \overset{k_f}{\rightarrow} x \qquad (17)$$
$$a_3 + y \overset{k_f}{\rightarrow} y$$

Phase 4:
$$a_4 + m_4 \overset{k_f}{\rightarrow} m_4$$

The reactions (18) provide the signal transfers associated with corresponding absence indicators. Signal transfers of each phase are enabled by the absence indicator of the previous phase. Note that these are all slow reactions.

Phase 1:
$$x' + a_4 \overset{k_s}{\rightarrow} D_1 + y$$
$$D'_1 + a_4 \overset{k_s}{\rightarrow} D_2 + y$$
$$D'_2 + a_4 \overset{k_s}{\rightarrow} y$$

Phase 4:
$$m_4 + a_3 \overset{k_s}{\rightarrow} m_2$$

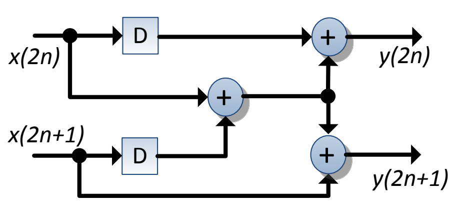
Phase 2:
$$m_2 + a_1 \overset{k_s}{\rightarrow} m_4$$

Phase 3:
$$D_1 + a_2 \overset{k_s}{\rightarrow} D'_1 \qquad (18)$$
$$D_2 + a_2 \overset{k_s}{\rightarrow} D'_2$$
$$y + a_2 \overset{k_s}{\rightarrow} y'$$
$$x + a_2 \overset{k_s}{\rightarrow} x'$$

According to reactions (17) and (18), molecules of x', y', D'_1, and D'_2 transfer in the first phase. After all molecules of x', y', D'_1, and D'_2 are transferred, phase 2 starts and m_2 is transferred to m_4. In phase 3, x, y, $D_1$and D_2 transfer, respectively, to x', y', D'_1 and D'_2. Concentration of $D'_1$and D'_2 are stored to be used for the computation of the next output. Thus, each pair of D_i and D'_i $(i = 1,2)$ function as a delay element.

3.2.2. High-Speed by Parallelism.

Using the circuit-level techniques [47], we can speed up the molecular reactions by parallel computing either in the context of RGB scheme or the proposed 4-phase scheme or a synchronous scheme.



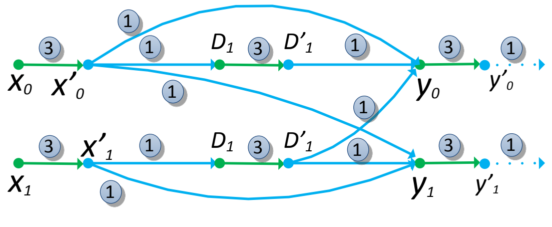
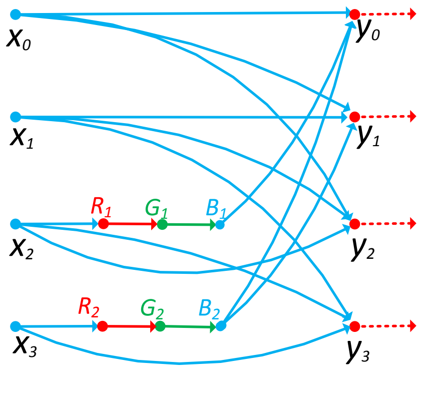


Figure 21. 2-parallel flow graph of the 3-tap FIR filter.

Figure 21(a) illustrates a 2-parallel version of the 3-tap filter in Figure 20(a). The scheduling to compute two parallel outputs is shown in the Figure 21(b). Instead of one input sample, two input samples are processed per each 4-phase cycle. The parallelization level can be increased arbitrarily. For example Figure 22 illustrates a 4-parallel implementation of the 3-tap filter using *RGB* and 4-phase schemes. Both of these schedules include two delay elements. The number of delay elements is same as the original flow graph for the 3-tap filter. Here the overall computational rate increases by factor of 4, compared to the original implementation. Unlike electronic circuits, parallelism in molecular systems *doesn't* increase the cost of implementations as reactants are available in abundance.



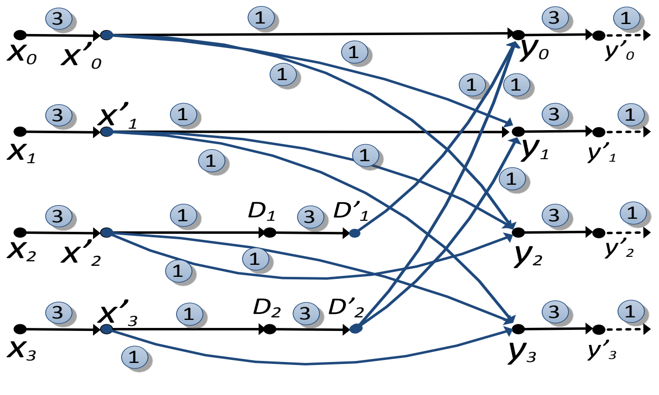


Figure 22. 4-parallel flow graph for the 3-tap filter in: (a) *RGB* scheme, (b) 4-phase scheme.

3.3. Power Spectral Density (PSD) Computation.

PSD represents the power of the input signal over a range of frequencies. Molecular implementation of the PSD has a potential role in many applications such as protein monitoring, drug delivery, disease state measurement, etc. The PSD of a signal is the Fourier transform of the auto-correlation of the signal. There exist different methods to compute or estimate the PSD. Due to the availability of computationally-efficient FFT algorithms, the periodogram approach is often preferred to other approaches. The widely used method to compute PSD is the Welch method [48] which is a modified periodogram approach. FFT is the core part of the Welch method. In general, as

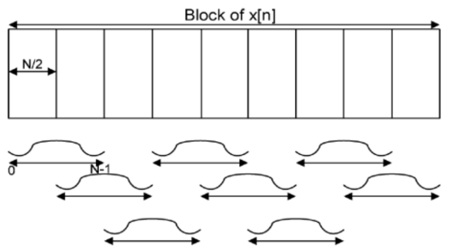


Figure 23. Dividing a block of input signal into overlapped segments.

illustrated in Figure 23 an overlap of 50% is used when dividing the input signal into multiple segments. In [33], we propose an architecture that reduces the number of operations required to compute the PSD. The proposed architecture requires a $\frac{N}{2}$-point FFT instead of an *N*-point FFT block, where *N* is the length of the window.

The main idea is to reuse the $\frac{N}{2}$-point FFT from the previous segment by moving the windowing operation into the frequency domain. This is only practical when the window functions are represented by raised cosine functions. The window operation in time domain is converted to a convolution operation in the frequency domain and is implemented by a symmetric 3-tap or 5-tap FIR filter using 2 or 3 multipliers, respectively. The low-complexity of the frequency-domain convolution with a short filter is the key to reduction in complexity of the PSD computation. Figure 24 illlustrates this idea. For FFT blocks efficient architectures for real-valued signal presented in [49] and [50] are used.

In the proposed approach, the even samples are computed exactly, while the odd samples require a shift by a half-sample delay. The method uses a *bidirectional* filter approach to implement the half-sample delay filter. Figure 25 shows the structure of the bidirectional filter used in [33].

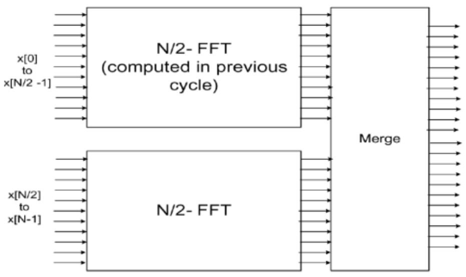


Figure 24. Using two consecutive $\frac{N}{2}$-point FFTs in order to compute one *N*-point FFT.

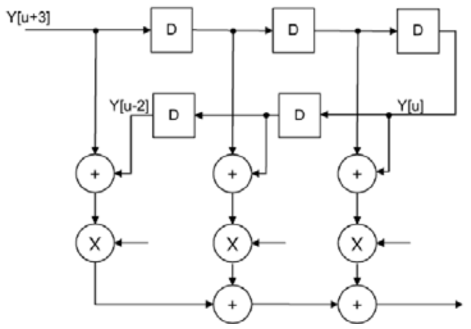


Figure 25. Bidirectional filter proposed in [33] for the PSD computation.

4. Schedule

This three year project will address molecular implementation of digital and discrete-time systems. Year-1 will be demonstrate ADC and DAC implementations by molecular implementations, and faster implementations by using the proposed 4-phase scheme. Year 2 will be devoted demonstration of fully-digital building blocks for functions such as multipliers, dividers, digital filters, and FFTs using DNA; and parallel implementations using RGB and 4-Phase schemes. Year-3 will be devoted to the demonstration of PSD using discrete-time and digital approaches.

5. Broader Impact of Proposed Research

If successful, the proposed research will transform molecular computing research for domains such as drug monitoring and drug delivery. Currently, an ineffective and ad-hoc approach prevails, where only very simple circuits implementing logical conditions such as “if-then-else” are considered. Our research will open up the field by permitting robust computation of time-varying functions. The full expertise and experience of the discipline of digital signal processing (DSP) will be brought to bear on important problems. Instead of approaching drug therapy as an exercise in collecting data, computing offline (i.e., electronically or by consulting a human expert), and then delivering drugs, new systems of *autonomous* molecular therapy will be engineered.

6. Minority Involvement Plan/Outreach/Education

The investigators will work with the University of Minnesota’s 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--Minnesota’s 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 University's laboratories. The PIs will engage a student from the North Star program.

6.1 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 hands on 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 one week sessions are devoted to girls only. The PIs will participate in this program to increase the involvement of high school students in the engineering program. This involvement will inspire high school students to be interested in science and technology.

6.2 Integration of Research into Course Curriculum

The results of this research will eventually be integrated into two graduate courses: EE-5393 "Circuits, computation and Biology" and EE-5329 "VLSI Signal Processing" taught by the investigators at the ECE department at the University of Minnesota. These classes typically involve course projects. Portions of the research can be used as course projects in these classes.

7. Results of Prior NSF Support

Parhi: The PI has recently completed two NSF grants. Award CCF-0811456: Collaborative Research: CPA-DA: Noise-Aware VLSI Signal Processing: A New Paradigm for Signal Processing Integrated Circuit Design in Nanoscale Era, 9/1/2008-8/31/2011. The EAGER grant CCF-0946601: EAGER: Synthesizing Signal Processing Functions with Biochemical Reactions (with M. Riedel) started on 8/1/09 and ended on 7/31/2011. 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. The robust demodulation work enabled removing sparse impulse noise without assuming a prior model of the probability density function (pdf) of the impulse noise as assumed in prior models. These results have been published in [51]-[57]. The EAGER grant on bimolecular signal processing allowed us to prove that signal processing can be implemented using chemical and molecular reactions in general and DNA strands in particular. This was the first attempt to prove that discrete-time signal processing systems can be implemented in DNA strands. This work also led to new digital logic reactions and new approaches to realizing flip flops in bimolecular reactions. Linear feedback shift registers and FFTs were also demonstrated in bimolecular reactions. These results have been published in [34]-[39], [58]-[60]. The PI received a new NSF grant on stochastic digital filters and transforms that started in September 2013.

Riedel: Grant 0845650, CAREER Award: Computing with Things Small, Wet, and Random Design Automation for Digital Computation with Nanoscale Technologies and Biological Processes"; 9/2009-8/2014; This 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. Broader impacts: The circuit-design community has unique expertise that can be brought to bear on the challenging computational problems encountered in synthetic biology. Applications in biology, in turn, offer a wealth of interesting problems in modeling and algorithmic development. With its cross-disciplinary emphasis, this project will bring new perspectives to both fields. The results have been published in [44], [61]-[76].