Parallel Processing Letters
Vol. 25, No. 1 (2015) 1540002 (18 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0129626415400022

Delay Line as a Chemical Reaction Network

Josh Moles
*Department of Electrical & Computer Engineering,
Portland State University
PO Box 751, Portland, Oregon 97207, United States*

Peter Banda
*Department of Computer Science,
Portland State University
PO Box 751, Portland, Oregon 97207, United States*

Christof Teuscher
*Department of Electrical & Computer Engineering,
Portland State University
PO Box 751, Portland, Oregon 97207, United States*

Received December 2013
Revised June 2014
Communicated by Guest Editor
Published 31 March 2015

ABSTRACT

Chemistry as an unconventional computing medium presently lacks a systematic approach to gather, store, and sort data over time. To build more complicated systems in chemistries, the ability to look at data in the past would be a valuable tool to perform complex calculations. In this paper we present the first implementation of a chemical delay line providing information storage that can reliably capture information over an extended period of time. The delay line is capable of parallel operations in a single instruction, multiple data (SIMD) fashion.

Using mass action and Michaelis-Menten kinetics, we describe the chemical delay line implementation featuring enzymes acting as a means to reduce copy errors. We also discuss how information is randomly accessible from any element on the delay line. Our work shows how the chemical delay line retains and provides a value from a previous cycle. The system's modularity allows for integration with existing chemical systems. We exemplify the delay line capabilities by integration with a threshold asymmetric signal perceptron to demonstrate how it learns all 14 linearly separable binary functions over a sliding window of size two. The delay line has applications in biomedical diagnosis and treatment, such as smart drug delivery.

Keywords: Chemical delay line; chemical computing; information storage; Michaelis-Menten kinetics; chemical perceptron; time series learning.

1. Introduction

The ability to store temporal data is a fundamental operation to many types of calculations and signal processing [1]. Access to the results from previous calculations or observations is essential to form more complex operations and devices, like first-in, first-out (FIFO) memories. Capturing the values of data over time in a chemical reaction network could enable sensing of concentrations over a window of time rather than just the present concentration observed. Chemistry provides an unconventional paradigm to solve computation tasks in a highly parallel fashion because the reactions changing the concentration of species inherently occur concurrently [2].

Chemical computing is rapidly growing field that would benefit from such a time delay line. Previous works on chemical computers address problems such as networking protocols [3], logic circuits [4–7], arithmetic [8], signal processing [9], tic-tac-toe [10] and chess [11]. Arkin and Ross [7] emphasized the need for "buffer" between the phases of logic elements. Chemical systems only have the capability to look into the present or previous value as inputs. With our delay line, the period of looking into the past could be arbitrary.

An example of biochemical application is smart medication [12–16]. Rather than have a fixed dosage of a specific type of medicine, a patient could be observed over a time window and then adapt the drug (in quantity or species) to best respond to their needs [17, 18]. Another use in the biochemistry field would be the detection of harmful species, e.g., chemicals produced by cancer cells in a host. With a time delay line, the detection would not be limited to a simple yes or no, but can be extended to measure a chemical concentration as well as capture at what point the event occurred. This could be used for monitoring or passed to another chemical computer (such as a chemical perceptron [6, 19]) that could then make decisions and react.

This work presents two variations of a time delay line implemented as a chemical reaction network. First, a manual copy model that requires additional signaling to indicate when it is time to propagate values through the delay line. Second model features an automatic backwards propagation of a copy signal. We will also demonstrate an example of this delay line by connecting it to a chemical perceptron capable of learning the 14 lineally separable two-input logic functions [6].

The buffer presented in this work is novel compared to prior work. Jiang *et al.* [9] introduced the concept of a delay element. The delay element is primarily used as a storage area for holding data in between each computation cycle. The data then returns and is examined in computing during the next iteration of the calculation. Jiang's buffer is primarily a signal processing application looking only at the previous value. Our delay line has the ability to delay not only multiple steps in time, but also allows access to any of the past values besides the most recent. We could create a FIFO [1] out of the delay line by removing the intermediate stages and providing only the final output.

Other areas, such as networking, use chemical reaction networks as a mechanism to control scheduling and queuing of packets [3]. The work discusses a methodology to use the law of mass action as a means to schedule packets. With a buffer like the one we are describing, the Meyer's systems could also be extended to actually implement a means to queue packets in a chemical system. This system would reduce cost and complexity by having a single implementation medium.

This paper will start by providing an overview of chemical reaction networks to understand the concepts discussed next in the design (Section 3) and results (Section 4) sections. We then describe a deoxyribozyme cascading implementation (Section 5) followed by our concluding remarks (Section 6).

2. Artificial Chemistry Background

We model the delay line by using several observations of nature, such as chemical kinetics and the law of conservation of mass. We follow the standard formalism for representing chemistry called chemical reaction network, which is an instance of an artificial chemistry [20]. It consists of a set of species and reactions with associated rates. In our system, molecular species are symbolic and unstructured, and there is no notion of space because we assume the solution is well stirred. Actually, we do not need to handle the position of individual molecules, but rather transform all molecules of the same type (species) using rates generated by kinetic laws: mass-action [21, 22] for regular and Michaelis-Menten [23–25] for catalytic reactions.

Dittrich [20] describes an artificial chemistry made up of a finite set of molecular species and a finite set of reactions. The set of molecular species are represented by symbols (e.g., S, S_1, P_1, $X1_{signal}$). The reactions are formed through multiple sets of species (reaction left side) that react to form products (reaction right side) [6]. A reaction looks like $S_1 + S_2 \rightarrow P$ where reactants S_1 and S_2 form the product P.

We combine mass action kinetics with the ideas of artificial chemistry to express reaction rates for ordinary (non-catalytic) reactions. Epstein [26] expresses this through a series of differential equations. Given a generic chemical reaction $aS_1 + bS_2 \rightarrow cP$, the rate of reaction, v, is expressed by

$$v = -\frac{1}{a}\frac{d[S_1]}{dt} = -\frac{1}{b}\frac{d[S_2]}{dt} = \frac{1}{c}\frac{d[P]}{dt} = k[S_1]^a[S_2]^b\,, \tag{1}$$

where $[S_1]$, $[S_2]$, and $[P]$ are the concentrations of the reactants, S_1 and S_2, and the product, P. Symbols a and b are stoichiometric constants, and k is the reaction rate constant. Reactions could also be reversible, but in this paper, for simplification, we assume the reverse rate is always zero.

Michaelis-Menten kinetics describes the rate of a catalytic reaction where a substrate (S) is transformed into a product (P) through the use of an enzyme or catalyst (E) in a reaction modeled as $E + S \rightleftharpoons ES \rightarrow E + P$. The catalyst E speeds up the rate of the reaction without being consumed in the process. The rate of a catalytic reaction is defined by

$$v = \frac{k_{cat}[E][S]}{K_m + [S]}\,, \tag{2}$$

where k_{cat} and K_m are rate constants. Copeland [27] discusses the work of Henri, Michaelis, and Menten in greater detail.

3. Delay Line Design

To introduce the time delay line design, we will first examine a delay line constructed of only two stages in two different styles. One is a manual copy delay line that requires experimenter participation to indicate when it is time to move values between stages. The second model automatically propagates the signaling species backwards, hence it is more autonomous, but it comes at the cost of additional and cumulative error in the resulting output values.

3.1. *Manual Copy Delay Line*

First, we will introduce the delay line of two stages with manual copy of the signaling species shown in Figure 1. A delay line of two stages is composed of seven species: X, $X1C$, $X1$, $X2$, $X2C$, $X2_{signal}$, and $X1_{signal}$. The species X represents the input value of the delay line. The signals, $X1_{signal}$ and $X2_{signal}$, are the catalysts that start the reaction conversion of X into corresponding stages. The primary function of $X1_{signal}$ is to trigger and accelerate the copy reaction which converts of X to $X1C$ and $X1$. Species $X2_{signal}$ performs a similar action for the conversion of $X1C$ to $X2$.

Species $X1C$ and $X2C$ are delayed copies of X that move to the next stage of the system (for example, $X1$ to $X2$ and $X2C$). Species $X2C$ is shown for completeness

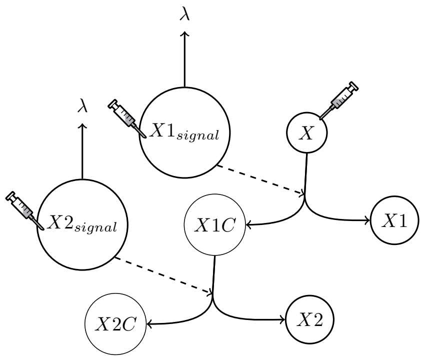


Fig. 1. Manual copy delay line with two stages. The syringe is used to indicate the species where inputs are presented and $X1$ and $X2$ represent the output species from the delay line. Species $X2C$ is used to cascade a value to a delay line of greater than two stages. The signal species, $X1_{signal}$ and $X2_{signal}$, catalyze the copy reactions and are removed from the system by decay (λ).

and is used to cascade the system to a delay line of more than two stages. For a two stage delay line, $X2C$ is waste and flushed. The outputs of the system are the $X1$ and $X2$ species, i.e., $X1$ and $X2$ represent the current and previous values of X that are consumed as the inputs of another system.

Species $X1C$ is the internal transition storage species. The storage species act as a buffer for the value that will transition into $X2$ on next activation of the system with an $X2_{signal}$ passed in. Ideally, the concentration of $X1C$ will be the same as $X1$ prior to its consumption. This process is represented by a set of reactions using the previously mentioned species. Reactions (3) and (4) (below) represent the conversion of the input species, X, through to the output species, $X1$ and $X2$.

$$X \xrightarrow{X1_{signal}} X1 + X1C \tag{3}$$

$$X1C \xrightarrow{X2_{signal}} X2 + X2C \tag{4}$$

Reactions (5) and (6) show the decay (represented by lambda, λ) of the catalyst species, $X1_{signal}$ and $X2_{signal}$.

$$X2_{signal} \rightarrow \lambda \tag{5}$$

$$X1_{signal} \rightarrow \lambda \tag{6}$$

Now, using these reactions, we can examine data moving through the delay line. For this manual copy delay line, actions must occur at two moments (in time). First, at time zero, we present a random value to the input X and reset $X1$ and $X2$ to zero. The reset of $X1$ and $X2$ simulate consumption by the underlying system the delay line is integrated with. Species $X2_{signal}$ is set to one to copy the value stored in $X1C$ to $X2$. In the ideal case for the initialization and first run of the delay line, $X2$ should be zero until these actions repeat. After 25 time steps, $X1_{signal}$ is injected to the system. The wait is to fully allow the transition from $X1C$ to $X2$ before beginning the transformation of X to $X1C$. These injections repeat every 1,000 time steps and are summarized in Table 1. Table 2 shows an example of these injections repeating every 1,000 time steps with example data moving through.

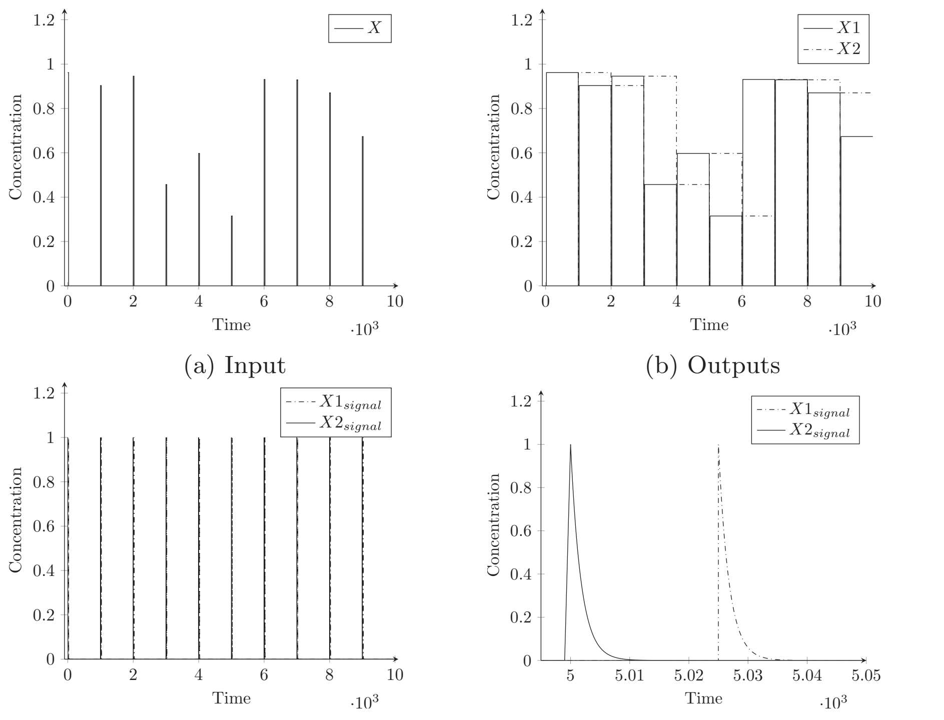
Figure 2 shows the results of running the actions in Table 1 for 10 iterations (10,000 time steps). Valid data is available for examination on output species $X1$ and $X2$ every time steps after each cycle. Figure 2a shows the input values injected to the manual delay line. During the first cycle, species $X2$ remains at zero since

Table 1. Actions for two stage manual copy delay line simulations.

Time	Species	Value
0	X	$0.0 \leq rand() \leq 1.0$
0	$X1$	0
0	$X2$	0
0	$X2_{Signal}$	1
25	$X1_{Signal}$	1

Table 2. Pipeline view of data moving through manual signaling delay line from Table 1. Bold items show those injected to the system. **A**, **B**, and **C** are inputs and **1** is a concentration (presence) of Xm_{signal}.

Species	Time = 0	25	1000	1025	2000	2025
X	**A**	$A \to 0$	**B**	$B \to 0$	**C**	$C \to 0$
X1signal		$\mathbf{1} \to 0$		$\mathbf{1} \to 0$		$\mathbf{1} \to 0$
X2signal	$\mathbf{1} \to 0$		$\mathbf{1} \to 0$		$\mathbf{1} \to 0$	
X1	**0**	$0 \to A$	**0**	$0 \to B$	**0**	$0 \to C$
X1C		$\to A$	$A \to 0$	$0 \to B$	$B \to 0$	$0 \to C$
X2	**0**		$\mathbf{0} \to A$	A	$\mathbf{0} \to B$	B
X2C			$\to A$	A	$A \to B$	B



(a) Input

(b) Outputs

(c) Copy Signals

(d) Copy Signals (Zoomed on Figure 2c)

Fig. 2. Two stage manual copy delay line showing inputs and outputs. Data arrives as input (a) and is available on outputs (b) with $X1$ being the current and $X2$ being the previous X. The copy of this data is triggered by $X1_{signal}$ and $X2_{signal}$ (c). Figure 2d shows the copy signals zoomed in from Figure 2c.

there is no previous value as seen in Figure 2b. Figure 2c shows the catalysts, $X2_{signal}$ and $X1_{signal}$, sequentially injected each cycle. Figure 2d presents the sequence of actions where $X2_{signal}$ is injected at time zero followed by $X1_{signal}$ 25 time steps later.

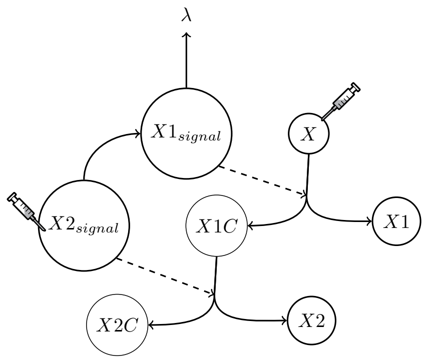


Fig. 3. Backwards propagating delay design with two stages. The syringe is used to indicate an injection of the input species X and the copy signal $X2_{signal}$. The species $X1$ and $X1$ represent the output species from the delay line. The signal $X2_{signal}$ is propagated backwards to $X1_{signal}$ without user intervention and then decays (λ).

3.2. *Backwards Signal Propagation Delay Line*

The backwards signal propagation delay line handles the signal species differently. More specifically, the only input signaling species is $X2_{signal}$ and rather than decay, $X2_{signal}$ reacts to $X1_{signal}$. The advantage of this model is that the user is only required to perform actions at the beginning of the cycle and then the system transforms the species internally (without external help). Figure 3 shows a revision of the manual copy delay line for this model. This reduces the number of injections to two: the input (X) and the final copy signal ($X2_{signal}$ for two stage). The change leaves reactions (7) and (8) unchanged.

$$X \xrightarrow{X1_{signal}} X1 + X1C \tag{7}$$

$$X1C \xrightarrow{X2_{signal}} X2 + X2C \tag{8}$$

Revising the remaining reactions requires modifying only reaction (5). Removing the decay from reaction (5) so that $X2_{signal}$ reacts to $X1_{signal}$ gives the updated reactions (9) and (10).

$$X2_{signal} \rightarrow X1_{signal} \tag{9}$$

$$X1_{signal} \rightarrow \lambda \tag{10}$$

All actions in the system occur instantaneously and are the same as actions employed by the manual delay line at time zero. At the beginning of every cycle, $X1$ and $X2$ are set to zero to simulate the next block of the system consuming their values. Also, a random value (X) and signal ($X2_{signal}$) are injected to the system. Table 3 summarizes these actions, which repeat every 1,000 time steps to ensure enough time for all reactions to reach steady state.

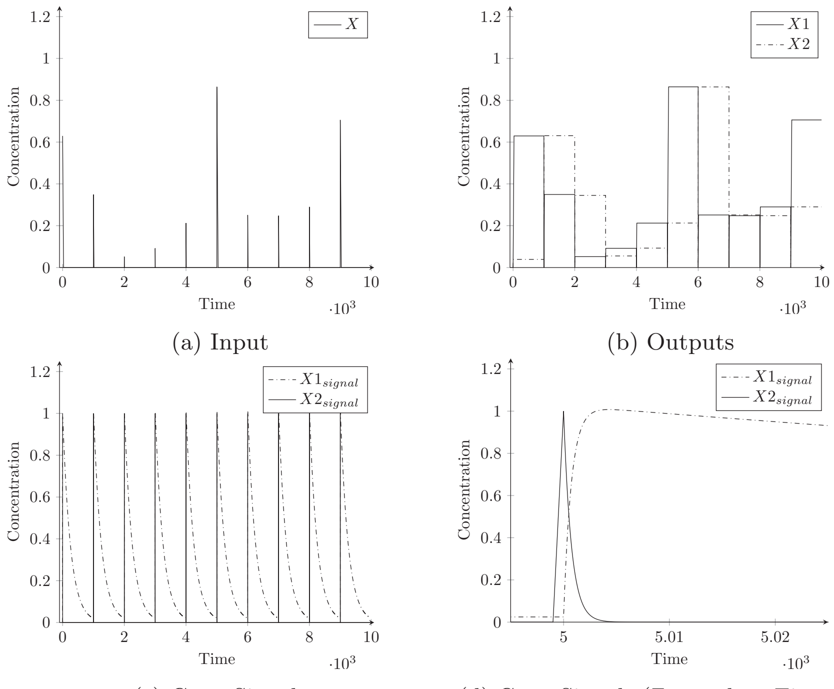
The simulations of the backwards signal propagation delay line run for 10,000 time steps (same as for the manual delay line). Valid data is also produced at the

Table 3. Actions for two stage back propagation delay line simulations. These actions repeat every 1000 similar to Table 2.

Time	Species	Value
0	X	$0.0 \leq rand() \leq 1.0$
0	$X1$	0
0	$X2$	0
0	$X2_{Signal}$	1

same point (every 50 steps) on the output species $X1$ and $X2$. The value produced on the first cycle of $X2$ ideally should be zero, but leakage from $X1C$ is generally seen from steps zero to 1,000 (see Fig. 4b). An input is introduced to the system at species X (Fig. 4a) and then is reacted in the same cycle to species $X1$ (Fig. 4b). After the next cycle (i.e., the next introduction of $X2_{signal}$), the value injected at X previously is now presented at $X2$ (Fig. 4b).

Notice that the backwards propagation introduces an error to the system with some of the $X2$ values not lining up exactly with the previous $X1$. This difference



(a) Input (b) Outputs (c) Copy Signals (d) Copy Signals (Zoomed on Figure 4c)

Fig. 4. Two stage backwards propagation delay line showing inputs and outputs. Data arrives as input (a) and is available on outputs (b) with $X1$ being the current and $X2$ being the previous X. The copy is started by $X1_{signal}$ and $X2_{signal}$ (c). Figure 4d shows the signals controlling propagation zoomed in from Fig. 4c.

is due to the time window that the reactions for X to $X1$ and $X1$ to $X2$ are simultaneously active. Looking at Fig. 4d, $X1_{signal}$ and $X2_{signal}$ are large enough for both catalyses to occur. So, for this small window of time, there is effectively a direct path from X to cascade down to $X2$. This overlap is not inherently a problem. It allows the desired parallelism of this system. We can afford this error in a small number of stages, but the inaccuracy can grow with a larger number of stages.

3.3. *Inherit Single Instruction, Multiple Data (SIMD)*

With the nature of chemistry, one of the advantages of our unconventional delay line implementation is the ability to perform single instruction, multiple data [28] operations. The main factor is finding a unique set of species to hold each delay line that will not react with surrounding buffers to allow such parallel operations. Figure 5 shows an example of a two-stage set of backwards propagation and manual copy delay lines that are producing a vector of three values for the current and previous cycles.

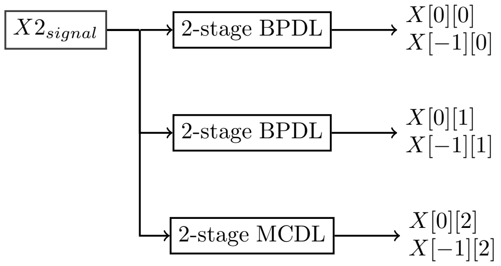


Fig. 5. Time delay design single instruction, multiple data (SIMD) representation showing simultaneous output of previous ($X[-1][n]$) and current ($X[0][n]$) X for parallel data processing. The signaling can be used with multiple instances of a delay line, both for the manual copy and the back propagation type.

3.4. *More than Two Stages*

Extending the buffer for more than two stages is straightforward. For each stage we add one output species (Xm), transition species (XmC), and catalyst species (Xm_{signal}). This allows the system to flexibly provide a buffer of desired length. As an example, Fig. 6 shows a backwards signal propagation delay line with three stages. The total number of species required in the system grows at a rate of $3m+1$, where m is equal to the number of stages in the system. One trade-off to note is that as the number of stages in the system increases, so does the period of time to cascade values through the delay line. Ideally, each reaction runs to full completion prior to Xm_{signal} propagating backwards to begin the next conversion.

The reaction set of the delay line also scales in a straightforward fashion. Each intermediate delay stage has a reaction similar to reaction (7) and the final delay

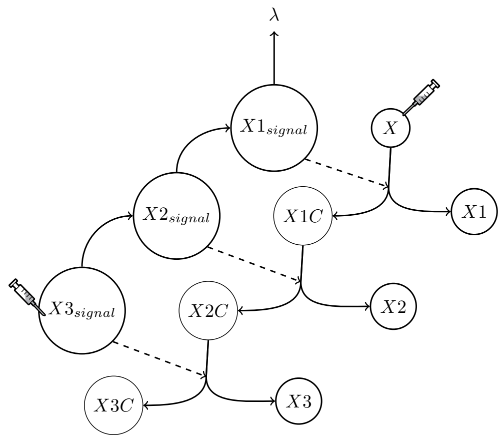


Fig. 6. Backwards signal propagation delay line with three stages. The syringe is used to indicate an injection of the input X and the signal $X3_{signal}$. Species $X1$, $X2$, and $X3$ represent the output species from the delay line. Lambda (λ) shows decay of backwards propagation signal.

stage (the mth delay) has a reaction similar to reaction (8). This remains true for extending both the manual and backwards propagating delay line. Extension of the catalysts depends on the implementation. For the manual copy delay line, simply adding the species and a subsequent input is required. Extending the backward propagating delay line has the advantage that it does not increase the number of injections, but it still increases the overall number of species.

4. Results

We will highlight the results for the two stage buffer and its extension beyond two stages. We employed Genetic Algorithms (GAs) [29] to optimize the rate constants (mapped to chromosomes) of the backwards propagation model. We only used the algorithm to optimize the backwards propagation model since the manual copy was straightforward to optimize by hand. The GA used an elite selection of the top 20 chromosomes from the population of 100, which undergo cross-over and mutation to form the next generation. The goal (fitness function) of this evolutionary algorithm was to minimize the error of the delay line.

We defined error as the difference between the actual input value (X) and the value occurring at $X1$ on this cycle and then $X2$ on the next cycle. We performed this test 50 time steps after X is injected into the delay line. Equation (11) shows the calculation of this error where $X[n]$ represents the current value of X and $X[n-1]$ represents the value of X on the previous input cycle. Adding both differences for the two stage delay line provided the overall error.

$$error = |X_1 - X[n]| + |X_2 - X[n-1]| \,. \tag{11}$$

The genetic algorithm performed perturbation mutation that changed each chromosome's element with 30% chance by ±30% using a uniform distribution. We ran

Table 4. Rate constants of two stage manual copy delay line found by GA.

Reaction	Forward Rate	K_m
$X \xrightarrow{X1_{signal}} X1 + X1C$	0.0757	2.0000
$X1C \xrightarrow{X2_{signal}} X2 + X2C$	0.0757	2.0000
$X2_{signal} \to \lambda$	0.5643	(None)
$X1_{signal} \to \lambda$	0.5643	(None)

the GA for 100 generations to produce the results for the two stage delay line. The algorithm was configured to target a transition of the input species, X, to the current time species, $X1$, as fast as possible, and convert the intermediate species, $X1C$ to the previous time species, $X2$, as fast as possible while minimizing the amount of leakage between the phases of the design.

4.1. *Two Stages*

Table 4 shows the rate constants for the manual propagation delay line reactions. Rates for the conversion of input species, X, down the chain is the same rate with the presence of $X1_{signal}$ and $X2_{signal}$ both increasing the rate by the same amount because the forward copy reactions should be as fast as possible. Figure 2 shows the plots using these rate constants in a two stage system, which can be replicated for a manual copy system of any size.

For a different size, the back propagation delay line has different rate constants. In addition, the rate constants were not grouped like the manual propagation delay line because it would drastically decrease the performance. Looking at the constants in Table 5, the reaction for species $X1C$ to $X2$ is the fastest. This is directly due to the rapid rate that $X2_{signal}$ is reacting to $X1_{signal}$. Effectively, to meet the first requirement of getting X into $X1$ as fast as possible, the lower level transition of species (Reaction (8)) must complete before. Figure 4 shows the output of a two stage backwards signal propagation delay line with the rate constants in Table 5.

Table 5. Rate constants of two stage backwards signal propagation delay line found by GA.

Reaction	Forward Rate	K_m
$X \xrightarrow{X1_{signal}} X1 + X1C$	0.0020	0.0225
$X1C \xrightarrow{X2_{signal}} X2 + X2C$	0.0706	2.0000
$X2_{signal} \to X1_{signal}$	1.3648	(None)
$X1_{signal} \to \lambda$	0.0039	(None)

To compare the accumulated error of the two delay lines we used symmetric mean absolute percentage error (SAMP) defined as

$$SAMP = 100 * \left\langle \frac{|y - \hat{y}|}{y + \hat{y}} \right\rangle, \tag{12}$$

where $\langle . \rangle$ is the mean of a set of multiple values, y is the actual value, and $\hat{y}$ is the expected value. We calculated an average SAMP per stage (unit size) by dividing cumulative SAMP with m. More specifically, using n to represent a discrete time series sample and m to represent the number of stages:

$$SAMP = \frac{100}{m} * \sum_{k=1}^{m} \left\langle \frac{|Xk - X[n-(k-1)]|}{Xk + X[n-(k-1)]} \right\rangle. \tag{13}$$

For instance, SAMP for two stages ($m = 2$) is given by

$$SAMP = \frac{100}{2} * \left\langle \frac{|X1 - X[n]|}{X1 + X[n]} + \frac{|X2 - X[n-1]|}{X2 + X[n-1]} \right\rangle. \tag{14}$$

We evaluated performance (error) over 10,000 runs, each repeating the sequence of actions defined in Table 1 and Table 3 for 200 iterations (200,000 time steps). Figure 7 shows the results for a delay line of size two as well as for larger sizes (discussed in next section). The difference in values from expected values for the two stage delay line are quite small. This shows that for a two stage delay line, both types operate well. One thing to note is that the backwards delay line has a larger initial error which can accumulate over time.

4.2. *Over Two Stages*

In this section, we will examine the use of a delay line with five stages. Five stages was selected and executed for both the manual copy and back propagating delay line. Figure 7 shows the final error when evaluated for 10,000 runs for 200 iterations each (same as for $m = 2$ in previous section). The maximum error over the entire evaluation is shown in Table 6. There are a few observations to note on this plot. The error on a backwards propagation delay line (B) increases as the number of stages in the delay line increases. For a smaller delay line, this error would generally be negligible, but for larger sizes this could be a concern. The manual copy has a significantly smaller error as shown in Figure 7.

As for the backwards propagating delay line, the error starts to accumulate to a noticeable value rapidly. Even by phase three, the delay line is starting to produce error that is in excess of the manual copy delay line with ten stages. Looking back to Fig. 4d there is a period of time where both $X1_{signal}$ and $X2_{signal}$ overlap which can explain how error that starts quite small in stage one of the delay system accumulates to a large value by the time it reaches the later stages of the buffer. Depending on the desired properties of the delay line, this is worth considering for the application.

Table 6. Maximum and average SAMP obtained through performance runs of 200 iterations and varying configurations of stages and manual copy and backwards propagation. Maximum and average excludes the initial values where the delay line is filling (first m points with low SAMP).

Backwards DL	Max	Average	Manual DL	Max	Average
B5	14.35%	14.09%	M10	0.0059%	0.0016%
B4	11.66%	11.25%	M5	0.0049%	0.0024%
B3	5.26%	4.84%	M2	0.0033%	0.0008%
B2	2.28%	1.97%			

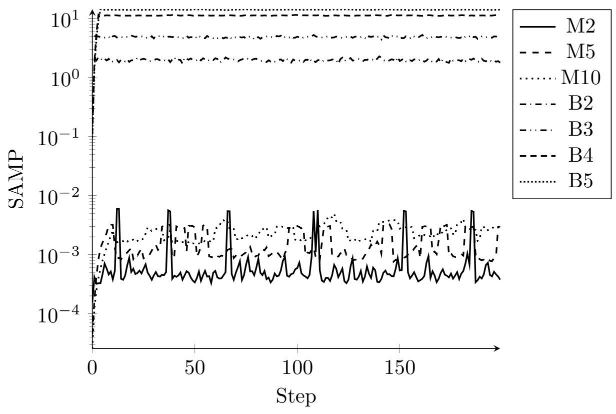


Fig. 7. SAMP calculated for delay lines. Mm and Bm are the m^{th} stage of manual copying and back propagation delay line.

4.3. *Time Series Perceptron Integration*

To demonstrate the capabilities of the delay line to fit into other designs, we integrated it with a chemical perceptron called a threshold asymmetric signal perceptron introduced in [19]. This perceptron learns through reinforcements and is inspired by biological neurons. Integration with the delay line and the perceptron shows how the delay line can easily fit with other systems to act as an input stream without any design modifications. Previously, the perceptron received both values simultaneously as two inputs. Now, we are showing that, without change to the perceptron or delay line, the two integrate together and function well. Figure 8 shows an example of this integration.

We trained the perceptron using reinforcement learning on 14 linearly separable binary function. Figure 9 shows the results of this learning. The binary time series perceptron learns 11 of the 14 functions with an accuracy of greater than 85%. Figure 10 shows the buffer and perceptron accurately producing the output for OR.

NAND, IMPL, and NOTX1 are all heavily dependent on the last input to resolve in the time delay line, $X1$. The input species $X1$ is not provided to the system until typically 50 time steps later than value $X2$. The original model of the perceptron

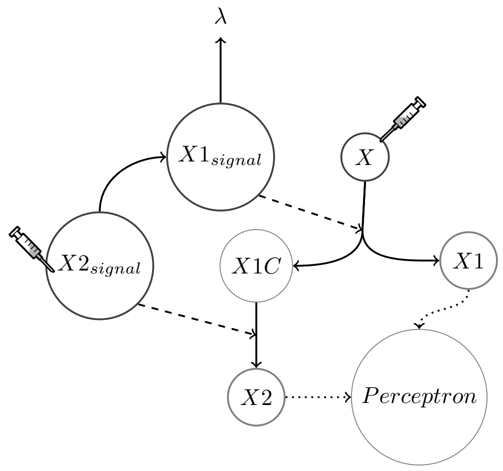


Fig. 8. Perceptron integration with backwards propagating delay line of two stages. The delay line outputs ($X1$ and $X2$) are fed to the perceptron without modification of the delay line.

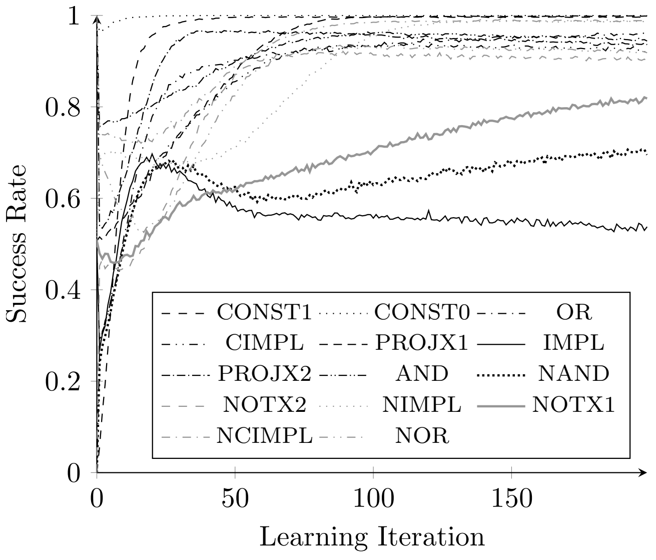


Fig. 9. Success rate of binary time series chemical perceptron. The perceptron learns 11 of the 14 linearly separable functions with an accuracy of greater than 85%.

was optimized for instantaneous and simultaneous injection of both inputs. Because input $X1$ is not ready, the performance is lower because that input plays a larger role on the correct performance for these logic functions. This makes the system capable of obtaining an average success rate of approximately 90% compared to the perceptron's 99% success rate [19].

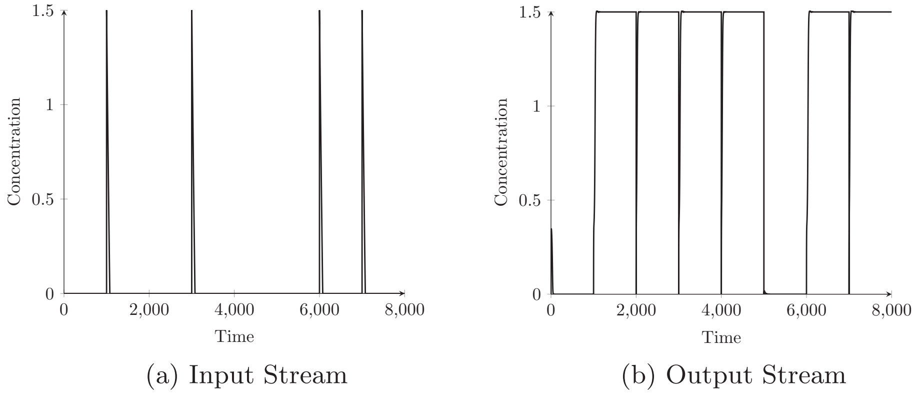


Fig. 10. Example concentration traces of binary time series chemical perceptron that successfully learns OR function. Left shows input stream 0, 1, 0, 1, 0, 0, 1, 1. Right shows correct output stream of 0, 1, 1, 1, 1, 0, 1, 1. Two zeros on the input stream at 4,000 and 5,000 successfully produce zero at time 5,000 on output stream.

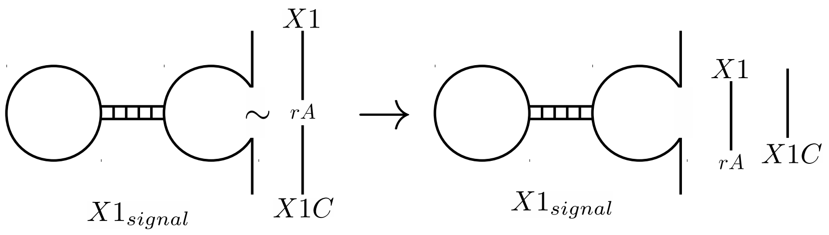


Fig. 11. Deoxyribozyme cascading example. Deoxyribozyme $X1_{signal}$ cleaves X at embedded ribonucleotide (rA) to form $X1$ and $X1C$. A similar process occurs on $X1C$ to produce $X2$ and $X2C$.

5. Deoxyribozyme Cascading Implementation

Now, we would like to present how such a delay line could be realized in a system employing deoxyribozyme catalysis [10, 30, 31]. Figure 11 shows an example of a two stage manual delay line with the signals being the deoxyribozymes $X1_{signal}$ and $X2_{signal}$ which cleave the substrate X at the embedded ribonucleotide. This produces $X1$ ready for the next system to consume. Subsequently, $X1C$ embedded with another ribonucleotide is able to get cleaved by deoxyribozyme $X2_{signal}$ to form the next input to the system, $X2$.

6. Conclusion

We have presented a novel implementation of a delay line as a chemical reaction network capable of storing past concentrations. By arranging our delay lines in a SIMD-like layout, we could delay multiple segments of data simultaneously with a shared control signal for either model of delay line. We have introduced two types of a chemical delay line: manual copy delay line and backwards propagation delay line. A manual copy delay line can precisely carry values in a delayed state, but requires more intervention from the user (growing at a rate of m) to propagate values through the system. The second model, backwards propagating delay line, automatically moves values through the system with a single signaling injection with reasonable accuracy.

The integration of the backwards propagating delay line with the threshold asymmetric signal perceptron resulted in the first chemical model capable of learning binary time series. Also, this example is a proof-of-concept that our delay line is a modular block ready for use in other systems. For systems requiring a smaller window of past values, either model of the delay line gives sufficient accuracy for data storage. The manual copy delay line shows potential for longer chains with the amount of calculated SAMP passed between phases remaining below 0.01% for a delay line of 10 stages. The backwards propagating delay line provides a much simpler user interface at the sacrifice of accuracy. A backwards propagation of five stages keeps the calculated SAMP below 15%. Systems requiring a large number of delays will have to weigh accuracy and simplicity to make a selection for a particular implementation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant No. 1028120.

References

[1] N. Kanopoulos and J. J. Hallenbeck, "A first-in, first-out memory for signal processing applications," *Circuits and Systems, IEEE Transactions on*, vol. 33, no. 5, pp. 556–558, 1986.

[2] B. De Lacy Costello and A. Adamatzky, "On multitasking in parallel chemical processors: experimental findings," *International Journal of Bifurcation and Chaos*, vol. 13, no. 02, pp. 521–533, 2003.

[3] T. Meyer and C. Tschudin, "Flow management in packet networks through interacting queues and law-of-mass-action scheduling," Technical Report CS-2011-001, University of Basel, Tech. Rep., 2011.

[4] N. Matsumaru, F. Centler, P. Speroni Di Fenizio, and P. Dittrich, "Chemical Organization Theory as a Theoretical Base for Chemical Computing," *Workshop on Unconventional Computing*, pp. 71–82, 2005.

[5] E. Katz, *Biomolecular Information Processing: From Logic Systems to Smart Sensors and Actuators*. Weinheim, Germany: Wiley-VCH Verlag & Co, 2012.

[6] P. Banda, C. Teuscher, and M. Lakin, "Online learning in a chemical perceptron," *Artificial life*, vol. 219, pp. 195–219, 2013.

[7] A. Arkin and J. Ross, "Computational functions in biochemical reaction networks." *Biophysical journal*, vol. 67, no. 2, pp. 560–78, Aug. 1994.

[8] E. Katz, *Molecular and Supramolecular Information Processing: From Molecular Switches to Logic Systems*. Weinheim, Germany: Wiley-VCH Verlag & Co, 2012.

[9] H. Jiang, S. Salehi, M. Riedel, and K. Parhi, "Discrete-Time Signal Processing with DNA," *ACS synthetic biology*, vol. 2, no. 5, pp. 245–254, May 2013.

[10] M. N. Stojanovic and D. Stefanovic, "A deoxyribozyme-based molecular automaton," *Nature biotechnology*, vol. 21, no. 9, pp. 1069–1074, 2003.

[11] D. Faulhammer, A. R. Cukras, R. J. Lipton, and L. F. Landweber, "Molecular computation: RNA solutions to chess problems," *Proceedings of the National Academy of Sciences*, vol. 97, no. 4, pp. 1385–1389, 2000.

[12] G. Neat, H. Kaufman, and R. Roy, "A hybrid adaptive control approach for drug delivery systems," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 1988, pp. 1305–1306.

[13] M. F. Abbod, D. A. Linkens, M. Mahfouf, and G. Dounias, "Survey on the use of smart and adaptive engineering systems in medicine," *Artificial Intelligence in Medicine*, vol. 26, no. 3, pp. 179–209, 2002.

[14] J. Halámek, V. Bocharova, S. Chinnapareddy, J. R. Windmiller, G. Strack, M.-C. Chuang, J. Zhou, P. Santhosh, G. V. Ramirez, M. A. Arugula, J. Wang, and E. Katz, "Multi-enzyme logic network architectures for assessing injuries: Digital processing of biomarkers." *Molecular BioSystems*, vol. 6, no. 12, pp. 2554–2560, Dec. 2010.

[15] J. Wang and E. Katz, "Digital biosensors with built-in logic for biomedical applications–biosensors based on a biocomputing concept." *Analytical and Bioanalytical Chemistry*, vol. 398, no. 4, pp. 1591–603, Oct. 2010.

[16] M. Zhou, N. Zhou, F. Kuralay, J. R. Windmiller, S. Parkhomovsky, G. Valdés-Ramírez, E. Katz, and J. Wang, "A self-powered "sense-act-treat" system that is based on a biofuel cell and controlled by boolean logic." *Angewandte Chemie (International ed. in English)*, vol. 51, no. 11, pp. 2686–9, Mar. 2012.

[17] S. Mailloux, J. Halámek, and E. Katz, "A model system for targeted drug release triggered by biomolecular signals logically processed through enzyme logic networks." *The Analyst*, vol. 139, no. 5, pp. 982–986, Mar. 2014.

[18] S. Mailloux, O. Zavalov, N. Guz, E. Katz, and V. Bocharova, "Enzymatic filter for improved separation of output signals in enzyme logic systems towards 'sense and treat' medicine," *Biomaterials Science*, vol. 2, no. 2, p. 184, 2014.

[19] P. Banda, C. Teuscher, and D. Stefanovic, "Training an asymmetric signal perceptron through reinforcement in an artificial chemistry," *Journal of The Royal Society Interface*, vol. 11, no. 93, p. 20131100, 2014.

[20] P. Dittrich, J. Ziegler, and W. Banzhaf, "Artificial chemistries–a review." *Artificial life*, vol. 7, no. 3, pp. 225–275, Jan. 2001.

[21] F. Horn and R. Jackson, "General mass action kinetics," *Archive for Rational Mechanics and Analysis*, vol. 47, no. 2, pp. 81–116, 1972.

[22] P. Érdi and J. Tóth, *Mathematical models of chemical reactions: theory and applications of deterministic and stochastic models*. Manchester, UK: Manchester University Press, 1989.

[23] V. Henri, *Lois générales de l'action des diastases*. Librairie Scientifique A. Hermann, 1903.

[24] L. Michaelis and M. L. Menten, "Die Kinetik der Invertinwirkung," *Biochem. Z*, vol. 49, no. 333-369, p. 352, 1913.

[25] V. Leskovac, *Comprehensive Enzyme Kinetics*, V. Leskovac, Ed. New York, New York, USA: Kluwer Academic/Plenum Publishers, 2003.

[26] I. R. Epstein and J. A. Pojman, *An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos*. New York, New York, USA: Oxford University Press, 1998.

[27] R. A. Copeland, *Enzymes: a practical introduction to structure, mechanism, and data analysis*. Hoboken, New Jersey, USA: John Wiley & Sons, 2004.

[28] M. J. Flynn, "Some computer organizations and their effectiveness," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 948–960, 1972.
[29] M. Mitchell, *An introduction to genetic algorithms.* Cambridge, Massachusetts, USA: MIT press, 1998.
[30] M. N. Stojanovic, P. de Prada, and D. W. Landry, "Homogeneous assays based on deoxyribozyme catalysis," *Nucleic acids research*, vol. 28, no. 15, pp. 2915–2918, 2000.
[31] J. Liu, Z. Cao, and Y. Lu, "Functional nucleic acid sensors," *Chemical reviews*, vol. 109, no. 5, pp. 1948–1998, 2009.