Synchronous Sequential Computation with Molecular Reactions*

Hua Jiang
hua@umn.edu

Marc D. Riedel
mriedel@umn.edu

Keshab K. Parhi
parhi@umn.edu

Department of Electrical and Computer Engineering, University of Minnesota
200 Union St. S.E., Minneapolis, MN 55455

ABSTRACT

Just as electronic systems implement computation in terms of voltage (*energy per unit charge*), molecular systems compute in terms of chemical concentrations (*molecules per unit volume*). Prior work has established mechanisms for implementing logical and arithmetic functions including addition, multiplication, exponentiation, and logarithms with molecular reactions. In this paper, we present a general methodology for implementing synchronous sequential computation. We generate a four-phase *clock signal* through robust, sustained chemical oscillations. We implement *memory elements* by transferring concentrations between molecular types in alternating phases of the clock. We illustrate our design methodology with examples: a binary counter as well as a four-point, two-parallel FFT. We validate our designs through ODE simulations of mass-action chemical kinetics. We are exploring DNA-based computation via strand displacement as a possible experimental chassis.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling Methodologies*

General Terms

Design

Keywords

Molecular Computation, Synthetic Biology, Computational Biology, Digital Design, Synchronous Logic, Sequential Logic

1. INTRODUCTION

There has been a groundswell of interest in molecular computation in recent years [12, 14, 17, 23]. Broadly, the field strives for molecular implementations of computational processes – that is to say processes that transform input concentrations of chemical types into output concentrations of chemical types. Some of the early work in the field discussed molecular solutions to challenging combinatorial problems such as the Hamiltonian Path Problem and Boolean Satisfiability [1]. In spite of the claims of "massive parallelism" – 100 Teraflop performance in a test tube! – such applications were never compelling. Chemical systems are inherently slow and messy, taking minutes or even hours to finish, and producing fragmented results. Such systems will never be competitive with conventional silicon computers for tasks such as number crunching.

And yet the broad impetus of the field is not computation *per se*. Rather it is the design of "embedded controllers" – chemical reactions, engineered into biological systems such as viruses and bacteria, to perform useful molecular computation *in situ* where it is needed. For example, consider a system for chemotherapy drug delivery with engineered bacteria. The goal is to get bacteria to invade tumors and selectively produce a drug to kill the cancerous cells. Embedded control of the bacteria is needed to decide where and how much of the drug they should deliver. The computation could be as simple as: "If chemical type X is present, produce chemical type Y" where X is a protein marker of cancer and Y is the chemo drug. Or it could be more complicated: produce Z if X is present and Y is not present or vice-versa (i.e, an exclusive-or function). Or it could be time-varying computation: produce Z if the rate of change of X is within certain bounds (i.e., band-pass filtering). Exciting recent work along these lines includes [2] and [18].

There have been deliberate attempts to bring concepts from digital circuit design into the field [3, 4, 5, 20, 19, 22, 21]. Prior work has described a computational constructs for chemical reaction networks: logical operations such as copying, comparing and incrementing/decrementing [15]; programming constructs such as "for" and "while" loops [16]; arithmetic operations such as multiplication, exponentiation and logarithms [15, 16]; and signal processing operations such as filtering [9, 13]. Building on this prior work, we present a general methodology for implementing *synchronous sequential* computation.

1.1 Computational Model

A molecular system consists of a set of chemical reactions, each specifying a rule for how types of molecules combine. For instance,

$$A + B \xrightarrow{k} 2C \tag{1}$$

specifies that one molecule of A combines with one molecule of B to produce two molecules of C. This reaction fires at a rate proportional to a kinetic constant k. We model the molecular dynamics in terms of *mass-action kinetics* [7, 8]: reaction rates are proportional to (1) the quantities of the participating molecular types; and (2) the kinetic constants. Accordingly, for the reaction above, the rate

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2011, June 5-10, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0636-2 ...$10.00.

of change in the concentrations of A, B and C is

$$-\frac{d[A]}{dt} = -\frac{d[B]}{dt} = 2\frac{d[C]}{dt} = k[A][B], \tag{2}$$

(here $[\cdot]$ denotes concentration). Most prior schemes for molecular computation depend on specific values of the kinetic constants (the k's associated with each reaction.) This limits the applicability since the kinetic constants are not constant at all; they depend on factors such as cell volume and temperature.

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

1.2 Organization

The rest of the paper is organized as follows. In Section 2, we present a design methodology for synchronous sequential computation, based on clocking with chemical oscillations. We implement memory elements – flip-flops – by transferring concentrations between molecular types in alternating phases of the clock. In Sections 3 and 4, we present two detailed design examples: a three-bit binary counter and a four-point, two-parallel FFT. In Section 5, we present simulations. Finally, in Section 6, we conclude the paper with a discussion of possible experimental applications and future directions.

2. SYNCHRONOUS SEQUENTIAL COMPUTATION

The general structure of our design is illustrated in Figure 1. As in an electronic system, our molecular system consists of separate reactions that implement *computation* and *memory*. A clock signal synchronizes transfers between computation and memory. For the computational reactions, we refer the reader to prior work [9, 15, 16]. Operations such as addition and scalar multiplication are straightforward. Operations such as multiplication, exponentiation, and logarithms are trickier. These can be implemented with reactions that implement iterative constructs analogous to "for" and "while" loops. (They do so robustly and exactly, without any specific dependence on the rates.)

The contribution of this paper consists of a new method for clock signal generation and for implementing memory.

2.1 Clock Generation

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 the chemical concentrations. With such oscillations, a low concentration corresponds to logical value of zero; a higher concentration corresponds to a logical value of one. Techniques for generating chemical oscillations are very well-known in the literature. Classic examples include the Lotka-Volterra, Brusselator and Arsenite-Iodate-Chlorite systems [6, 10]. However, none of these schemes are quite suitable for synchronous sequential computation. We require that the clock signal be perfectly symmetrical, with abrupt transitions between the phases.

Here we present a new design for a 4-phase chemical oscillator. The clock phases are represented by molecular types $R(ed)$, $G(reen)$, $B(lue)$, and $Y(ellow)$. First consider the reactions

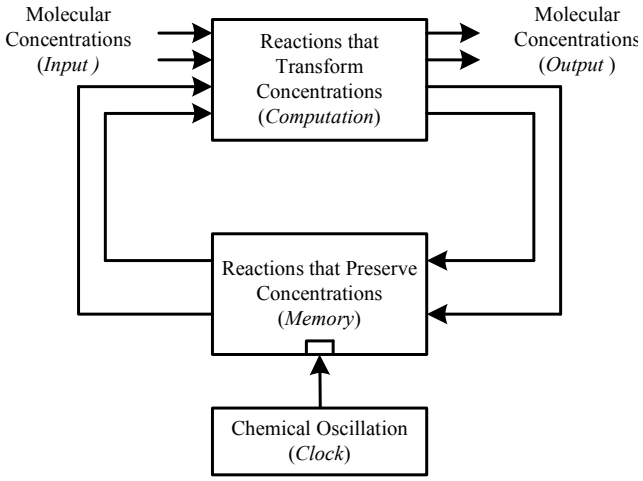


Figure 1: Block diagram of a synchronous sequential system.

$$\begin{array}{lcl} \varnothing & \xrightarrow{\text{slow}} & r \\ \varnothing & \xrightarrow{\text{slow}} & g \\ \varnothing & \xrightarrow{\text{slow}} & b \\ \varnothing & \xrightarrow{\text{slow}} & y \end{array} \tag{3}$$

and

$$\begin{array}{lcl} R + r & \xrightarrow{\text{fast}} & R \\ G + g & \xrightarrow{\text{fast}} & G \\ B + b & \xrightarrow{\text{fast}} & B \\ Y + y & \xrightarrow{\text{fast}} & Y. \end{array} \tag{4}$$

Reactions 3 generate molecular types r, g, b, and y slowly and constantly. Here the symbol $\varnothing$ indicates "no reactants" meaning the products are generated from a large or replenishable source. In Reactions 4, the types R, G, B, and Y quickly consume the types r, g, b, and y, respectively. Call R, G, B, and Y the *phase signals* and r, g, b, and y the *absence indicators*. The latter are only present in the absence of the former. The reactions

$$\begin{array}{lcl} R + y & \xrightarrow{\text{slow}} & G \\ G + r & \xrightarrow{\text{slow}} & B \\ B + g & \xrightarrow{\text{slow}} & Y \\ Y + b & \xrightarrow{\text{slow}} & R \end{array} \tag{5}$$

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

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

$$\begin{array}{rcl} 2G & \underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} & I_G \\ R + I_G & \xrightarrow{\text{fast}} & 3G \\ 2B & \underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} & I_B \\ G + I_B & \xrightarrow{\text{fast}} & 3B \\ 2Y & \underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} & I_Y \\ B + I_Y & \xrightarrow{\text{fast}} & 3Y \\ 2R & \underset{\text{fast}}{\overset{\text{slow}}{\rightleftharpoons}} & I_R \\ Y + I_R & \xrightarrow{\text{fast}} & 3R. \end{array} \tag{6}$$

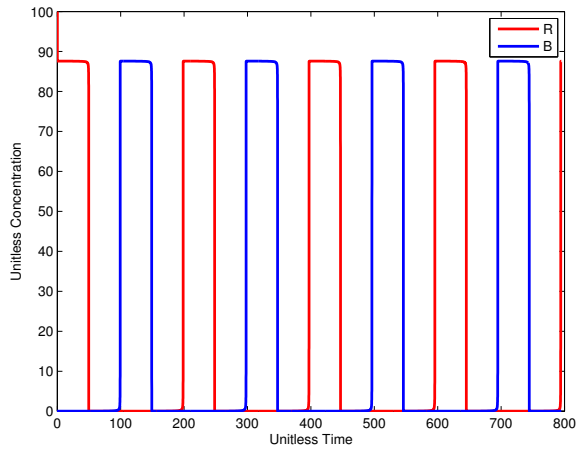


Figure 2: ODE simulation of the chemical kinetics of the proposed clock.

Consider the first two reactions. Two molecules of G combine with one molecule of R to produce three molecules of G. The first step in this process is reversible: two molecules of G can combine, but in the absence of any molecules of R, the combined form will dissociate back into G. So, in the absence of R, the quantity of G will not change much. In the presence of R, the sequence of reactions will proceed, producing one molecule of G for each molecule of R that is consumed. Due to the first reaction $2G \xrightarrow{\text{slow}} I_G$, the transfer will occur at a rate that is super-linear in the quantity of G; this speeds up the transfer and so provides positive feedback.[1]

Suppose that the initial quantity of R is set to some non-zero amount, and the initial quantity of the other types is set to zero. We will get an oscillation among the quantities of R, G, B, and Y. We choose two nonadjacent phases, R and B, as the clock phases.

Our scheme for chemical oscillation works remarkably well. Figure 2 shows the concentrations of R and B as a function of time, obtained through ordinary differential equation (ODE) simulations of the reactions 3, 4, 5 and 6. We note that the R (red) and B (blue) phases are non-overlapping.

2.2 Memory

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

The blue phase reactions are:

$$\begin{aligned} B + D_i &\xrightarrow{\text{fast}} Computations + B \\ Computations &\xrightarrow{\text{fast}} D'_j. \end{aligned} \tag{7}$$

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

[1]A rigorous discussion of chemical kinetics is beyond the scope of this paper. Interested readers can consult [6].

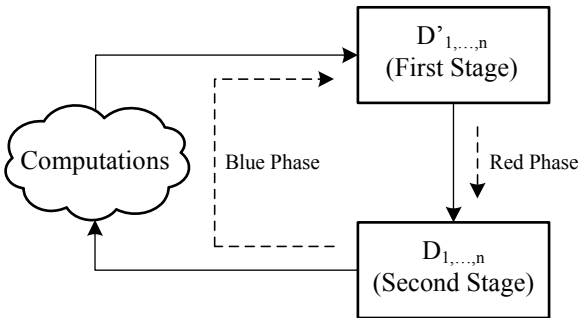


Figure 3: The two-phase memory transfer scheme.

The red phase reactions are

$$R + D'_j \xrightarrow{\text{fast}} D_k + R. \tag{8}$$

Every unit S_j transfers the signal it stores in D'_j to D_k, preparing for the next cycle. For the equivalent of delay (D) flip-flops in digital logic, $j = k$. For other types of memory units, j and k can be different. For example, for a toggle (T) flip-flop, S_k is the complementary bit of S_j: $D'_j \longrightarrow D_k$ and $D'_k \longrightarrow D_j$ toggle the pair of bits in each clock cycle. The transfer diagram for our memory design is shown in Figure 3.

3. A BINARY COUNTER

As our first design example, we present a three-bit binary counter. (It can readily be generalized to an arbitrary number of bits.). The counter consists of three identical stages, each of which processes a single bit. The block diagram of a stage is shown in Figure 4a.

In the nth stage, there are two memory units T_1^n and T_0^n that form a T flip-flop. The presence of molecules of T_1^n indicates that this bit is logical one; the presence of molecules of T_0^n indicates that this bit is logical zero. If we provide a non-zero initial concentration to one of the two types, then either T_0^n or T_1^n will always be present. Applying the memory implementation discussed in Section 2.2, we have types $T_1'^n$ and $T_0'^n$ as the first stages of the memory units.

The red phase reactions are

$$\begin{aligned} R + T_1'^n &\xrightarrow{\text{fast}} T_0^n + R \\ R + T_0'^n &\xrightarrow{\text{fast}} T_1^n + R. \end{aligned} \tag{9}$$

These toggle each bit. The blue phase reactions are

$$\begin{aligned} I^{n-1} + T_0^n &\xrightarrow{\text{fast}} T_0'^n \\ I^{n-1} + T_1^n &\xrightarrow{\text{fast}} T_1'^n + I^n. \end{aligned} \tag{10}$$

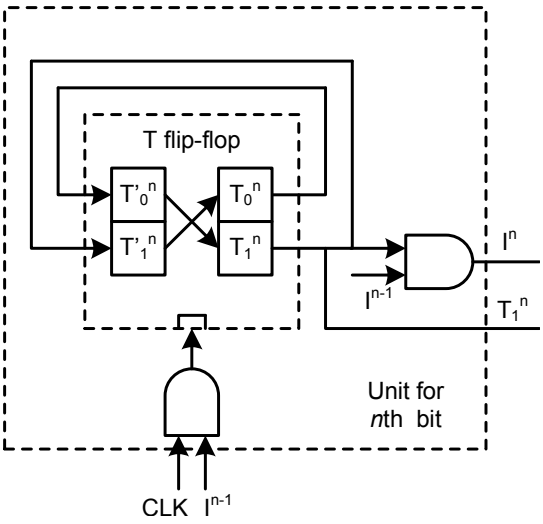
These simply feed the output of each T flip-flop back to its input.

Note that the T flip-flops transfer molecules only when there are molecules of I^{n-1} injected from the previous stage. If the bit is logical one, i.e., T_1^n is present, then molecules of I^n are injected into the next stage.

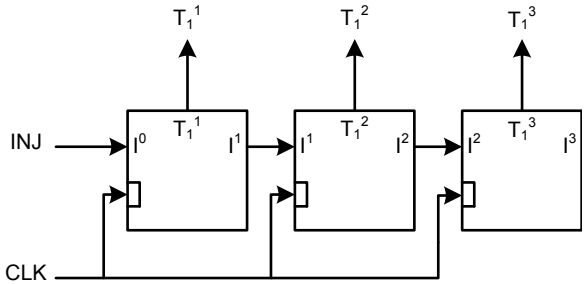
Figure 4b illustrate three connected stages. Reaction

$$B + Inj \xrightarrow{\text{fast}} I^0 + B \tag{11}$$

transfers the external injection Inj to I^0 in the blue phase. Since this reaction is the very first of all computational reactions, all injection signals I^n are generated in the blue phase. Accordingly, B is not required in Reaction 10, so long as I^n is among the reactants.



(a) Block diagram of one stage of the counter.



(b) Connecting the stages.

Figure 4: The binary counter.

4. A TWO-PARALLEL FFT DESIGN

We present a second design example, a four-point two-parallel fast Fourier transform (FFT). The FFT operation is canonical in signal processing. It can have a parallel pipelined architectures for high throughput [11]. A block diagram is shown in Figure 5. Assume that the system starts at clock cycle 1. The first two inputs are sampled in cycle 1; the last two inputs are sampled in cycle 2. The system generates the first outputs in cycle 3; it generates the last two outputs in cycle 4.

There are four switches in this design. Each selects one of the two incoming signals alternatively in different cycles. To achieve this switching functionality in our molecular design, we use two alternating selection signals. We generate these with a pair of D-flip-flops, as shown in Figure 6. If there is a non-zero initial concentration of S_1', then S_0' and S_1' will be "turned on" once every two cycles, in alternating fashion, starting with S_1'. We implement this computation with following reactions:

$$\begin{array}{rcl} R + S_0' & \xrightarrow{\text{fast}} & S_0 + R \\ R + S_1' & \xrightarrow{\text{fast}} & S_1 + R \\ B + S_1 & \xrightarrow{\text{fast}} & S_0' + B \\ B + S_0 & \xrightarrow{\text{fast}} & S_1' + B \end{array} \tag{12}$$

The transfer reactions enabled by S_1' or S_0' implement the switches. Note that it is S_1' and S_0', not S_1 and S_0, that enable the switches, because they are generated in blue phase.

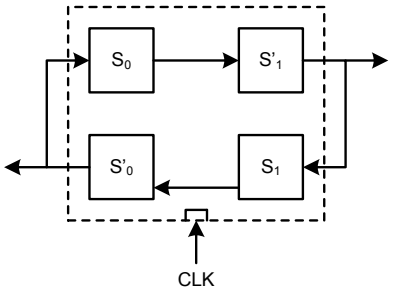


Figure 6: Generating the selection signals.

In this system, signals are complex numbers. Both the real and the imaginary parts can be negative numbers. To represent the signals, each number X is assigned four molecular types X_p, X_n, X_p^*, and X_n^*. The first two are assigned to the real parts: X_p represents the positive component and X_n the negative component. The last two are assigned to the imaginary parts: X_p^* represents the positive component and X_n^* the negative component. Therefore, $X = [X_p] - [X_n] + j([X_p^*] - [X_n^*])$.

Adders are implemented by assigning input edges and output edges to the same molecular type [9]. Note that there are two negative input edges in the lower two adders. Signals from the negative input edge will be transferred to the opposite component. For example, at the $n+1$st clock cycle, M_1 is transferred to O_1 and O_2 as

$$\begin{array}{rcl} S_1' + M_{1,p} & \xrightarrow{\text{fast}} & O_{1,p} + O_{2,n} + S_1' \\ S_1' + M_{1,n} & \xrightarrow{\text{fast}} & O_{1,n} + O_{2,p} + S_1' \\ S_1' + M_{1,p}^* & \xrightarrow{\text{fast}} & O_{1,p} + O_{2,n}^* + S_1' \\ S_1' + M_{1,n}^* & \xrightarrow{\text{fast}} & O_{1,n} + O_{2,p}^* + S_1' \end{array} \tag{13}$$

or simply

$$S_1' + M_1 \xrightarrow{\text{fast}} O_1 + O_2^- + S_1'. \tag{14}$$

Also, for each number X, the reactions

$$\begin{array}{rcl} X_p + X_n & \xrightarrow{\text{fast}} & \varnothing \\ X_p^* + X_n^* & \xrightarrow{\text{fast}} & \varnothing \end{array} \tag{15}$$

are required. They cancel out equal concentrations of positive and negative components by transferring them to an external sink.

There is a $-j$ multiplication in the system. It is implemented by

$$\begin{array}{rcl} S_1' + M_{2,p} & \xrightarrow{\text{fast}} & D_{4,n}'^* + S_1' \\ S_1' + M_{2,n} & \xrightarrow{\text{fast}} & D_{4,p}'^* + S_1' \\ S_1' + M_{2,p}^* & \xrightarrow{\text{fast}} & D_{4,n}' + S_1' \\ S_1' + M_{2,n}^* & \xrightarrow{\text{fast}} & D_{4,p}' + S_1' \end{array} \tag{16}$$

or simply

$$S_1' + M_2 \xrightarrow{\text{fast}} D_4'^{-*} + S_1' \tag{17}$$

which transfers real/imaginary parts to imaginary/real parts with opposite polarity.

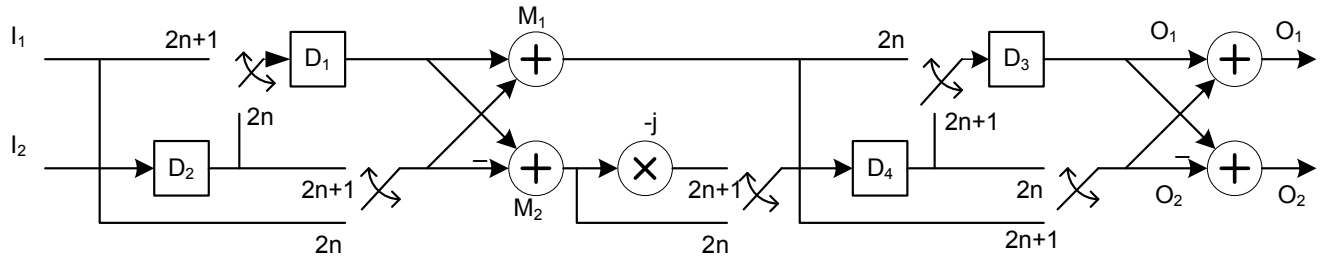


Figure 5: Block diagram of a 4-point pipelined FFT design.

Based on the computational operations discussed above, we have the blue phase reactions

$$\begin{aligned}
I_1 + S_1' &\xrightarrow{\text{fast}} D_1' + S_1' \\
I_1 + S_0' &\xrightarrow{\text{fast}} M_1 + M_2^- + S_0' \\
I_2 + B &\xrightarrow{\text{fast}} D_2' + B \\
D_2 + S_0' &\xrightarrow{\text{fast}} D_1' + S_0' \\
D_2 + S_1' &\xrightarrow{\text{fast}} M_1 + M_2^- + S_1' \\
D_1 + B &\xrightarrow{\text{fast}} M_1 + M_2 + B \\
M_2 + S_1' &\xrightarrow{\text{fast}} D_4'^{-*} + S_1' \\
M_2 + S_0' &\xrightarrow{\text{fast}} D_4' + S_0' \\
M_1 + S_0' &\xrightarrow{\text{fast}} D_3' + S_0' \\
M_1 + S_1' &\xrightarrow{\text{fast}} O_1 + O_2^- + S_1' \\
D_3 + B &\xrightarrow{\text{fast}} O_1 + O_2 + B \\
D_4 + S_0' &\xrightarrow{\text{fast}} O_1 + O_2^- + S_0' \\
D_4 + S_1' &\xrightarrow{\text{fast}} D_3' + S_1'.
\end{aligned} \tag{18}$$

Note that S_0' and S_1' are generated in the blue phase. It is not necessary to list B if a reaction is enabled by S_0' or S_1'.

The red phase reactions are

$$\begin{aligned}
R + D_1' &\xrightarrow{\text{fast}} D_1 + R \\
R + D_2' &\xrightarrow{\text{fast}} D_2 + R \\
R + D_3' &\xrightarrow{\text{fast}} D_3 + R \\
R + D_4' &\xrightarrow{\text{fast}} D_4 + R.
\end{aligned} \tag{19}$$

So the full design of the four-point, two-parallel FFT consists of Reactions 12, 18 and 19, together with the positive/negative canceling reactions as well as the clock generation reactions.

5. SIMULATIONS

We present simulation results for the binary counter and for the FFT. For each design, we list our choice of kinetic constants corresponding to "slow" and "'fast" as well as the initial concentrations of the molecular types. We assume that an external source sets the concentrations of the input types to new values at specific intervals. We setup ordinary differential equations corresponding to the mass-action kinetics of the reactions and solve these numerically with MATLAB.

5.1 Transient Response

5.1.1 Counter

For the three-bit counter, we set the initial concentrations of T_0^0, T_0^1, and T_0^2 to 10 (corresponding to bits "000") and R to 100. We set the initial concentrations of all the other molecular types to 0. We set the concentration of type Inj to 10 at time points 50, 500, 1000, 1500, 2000, 2500, 3000. We set k_{fast} to 100; and k_{slow} to 1. The results of a MATLAB ODE simulation are shown in Figure 7. We see that the bit signals T_1^0, T_1^1, and T_1^2 toggle at the correct

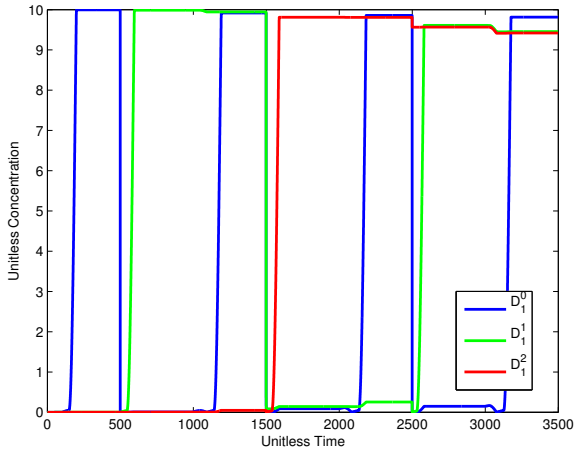


Figure 7: Transient simulation result of the counter.

time points. The system counts the number of injection events from "000" to "111" correctly.

One observation from Figure 7 is that the concentrations of T_1^0, T_1^1, and T_1^2 when the corresponding bit is "1" degrade slowly over time. This is because of slightly overlapped clock phases. There is always a slight leakage amount of R in the blue phase and a slight amount of B in the red phase. This error accumulates over time, due to the feedback loop in each stage of the counter. To mitigate against this, we could select a higher ratio of $\lambda = \frac{k_{\text{fast}}}{k_{\text{slow}}}$.

5.1.2 FFT

For our FFT design, we the set initial concentration of S_0' to 50 and that of R to 100. Recall that S_0' is transferred to S_0 in the first red phase and S_0 is transferred to S_1' in first blue phase. So the computation begins with S_1'. We set the initial concentrations of all the other types to 0. We inject I_1 and I_2 in the red phase. The output types O_1 and O_2 are built in clock cycle 3, in a blue phase. We clear them out in following red phase. We set k_{fast} to 100; and k_{slow} to 1. The results of a MATLAB ODE simulation are shown in Figure 8. The inputs are a sequence of real numbers {10, 15, 10 ,0}. The outputs are {35, -15j, 5, 15j}, as shown in the figure.

Since there is no feedback in the FFT architecture, no error due to leakage of R and B accumulates.

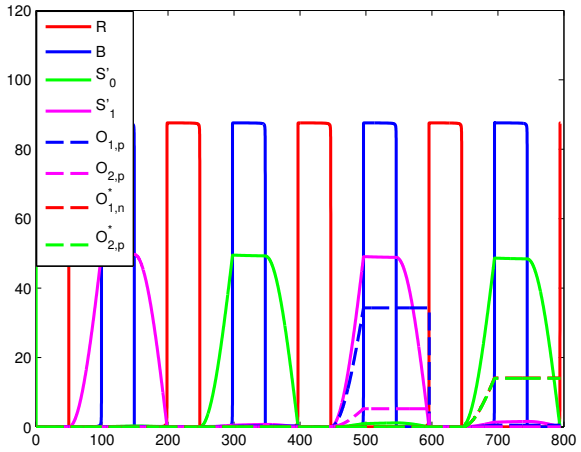


Figure 8: Transient simulation result of the FFT design.

Table 1: Relative error in simulations.

λ	Counter (Average error per cycle per bit)	FFT
10	0.9871%	28.107%
100	0.2078%	3.5428%
1000	0.0169%	0.2691%

5.2 Error Analysis

We analyze computational errors of the counter and the FFT design in terms of the fast-to-slow ratio λ. For the counter, the error is defined as the differences of concentrations from a perfect "0" or "1". We consider the average error accumulated in one clock cycle for one bit. For the FFT design, we consider the relative error of the simulated outputs compared to theoretical outputs. These errors are listed in Table 1. As expected, we see that the error decreases as λ increases: with a higher fast-to-slow ratio, fewer reactions misfire – that is, fire in the wrong clock phase.

6. REMARKS

This paper presents the first robust, rate-independent methodology for synchronous computation. Here "rate-independent" refers to the fact that, within a broad range of values for the kinetic constants, the computation is exact and independent of the specific rates. The results in this paper are complementary to prior results on *self-timed* methodologies for molecular computation [9]. Those had a distinct asynchronous flavor. As in electronic circuit design, there are advantages and disadvantages to asynchronous and synchronous design styles for molecular computing. On the one hand, a synchronous style leads to simpler designs with fewer reactions. On the other hand, errors can accumulate across clock cycles.

We are exploring the mechanism of DNA strand-displacement as an experimental chassis [17]. DNA strand-displacement reactions can emulate chemical reactions with nearly any rate structure. Reaction rates are controlled by designing sequences with different binding strengths. The binding strengths are controlled by the length and sequence composition of "toehold" sequences. With the right choice of toehold sequences, reaction rates differing by as much as 10^6 can be achieved. Our contribution can be positioned as the "front-end" of the design flow – analogous to technology-independent design. DNA assembly can be considered the "back-end" – analogous to technology mapping to a specific library.

7. REFERENCES

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

[2] J. C. Anderson, E. J. Clarke, A. P. Arkin, and C. A. Voigt. Environmentally controlled invasion of cancer cells by engineered bacteria. *Journal of Molecular Biology*, 355(4):619–627, 2006.

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

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

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

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

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

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

[9] H. Jiang, A. P. Kharam, M. D. Riedel, and K. K. Parhi. A synthesis flow for digital signal processing with biomolecular reactions. In *IEEE International Conference on Computer-Aided Design*, pages 417–424, 2010.

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

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

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

[13] M. Samoilov, A. Arkin, and J. Ross. Signal processing by simple chemical systems. *Journal of Physical Chemistry A*, 106(43):10205–10221, 2002.

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

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

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

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

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

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

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

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

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

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