

Reconfigurable Neuromorphic Computation in Biochemical Systems*

Hui-Ju Katherine Chiang^{1,2}, Jie-Hong R. Jiang^{1,3}, and François Fages²

Abstract—Implementing application-specific computation and control tasks within a biochemical system has been an important pursuit in synthetic biology. Most synthetic designs to date have focused on realizing systems of fixed functions using specifically engineered components, thus lacking flexibility to adapt to uncertain and dynamically-changing environments. To remedy this limitation, an analog and modularized approach to realize reconfigurable neuromorphic computation with biochemical reactions is presented. We propose a biochemical neural network consisting of neuronal modules and interconnects that are both reconfigurable through external or internal control over the concentrations of certain molecular species. Case studies on classification and machine learning applications using the DNA strand displacement technology demonstrate the effectiveness of our design in both reconfiguration and autonomous adaptation.

I. INTRODUCTION

Accelerating advances in synthetic and systems biology have enabled complex design and application of biochemical processes. For example, biochemically based computation and control [5], [8] can be engineered to recognize patterns among biomarkers [6], [13], [18] that are responsible for certain diseases and to further rectify problematic pathways. However most engineered systems have fixed functionality, and lacks flexibility in dynamic adaptation to its changing biochemical environment, which is intrinsically full of stochasticity and variability.

Not until recently, *programmable* [2] and *reconfigurable* [4], [3] biochemical systems have been proposed in digital logic and linear control domains, where reprogrammability is achieved through concentration control of predefined species. However, autonomous system adaptation remains an illusion as the reprogrammability achieved by prior work relies on external control. Prior work considers system scenarios identified before the design stage. Unfortunately, in many circumstances not all scenarios can be fully characterized or defined in advance. The ability to achieve on-site learning and to reconfigure accordingly is crucial to realize autonomous system adaptation. Bio-inspired neuromorphic computation provides an ideal scheme for learning from high-dimensional and noisy data for autonomous system reconfiguration.

Despite previous molecular implementations of neural networks [10], [17], the proposed architectures lack reconfigurability. Implementing neuromorphic systems with built-in learning capability remains challenging. We present in this paper the first *chemical reaction* construction of *reconfigurable* artificial neural networks. Similar to the silicon-based field programmable gate arrays (FPGAs), a module-based architecture is proposed, which consists of programmable neuron modules and their synaptic connections. The reconfigurability lies in the adjustable *firing thresholds* and *weighted connections* of the architecture. To demonstrate the feasibility of our method in potential real-world applications, we perform case studies on classification and machine learning, and exploit the DNA strand displacement (DSD) technology as the underlying experimental chassis realizing chemical reaction networks (CRNs) [16].

*This work was supported in part by MOST under grants 103-2221-E-002-273 and 103-2622-E-002-031.

¹GIEE, National Taiwan University, Taipei 106, Taiwan

²EPI Lifeware, Inria Paris-Rocquencourt, France

³CS Lab, Tomsk State University, Tomsk 634050, Russia

II. PRELIMINARIES

A. Model of Chemical Reaction Dynamics

A *chemical reaction network (CRN)* is composed of a set of reactants \mathbf{r} , a set of products \mathbf{p} , and a set of reactions describing the transformations from some subset of reactants to some subset of products. A reaction is often expressed in the form

$$\sum_{i=1}^n \alpha_i r_i \rightarrow \sum_{j=1}^m \beta_j p_j,$$

where species $r_i \in \mathbf{r}$ is the i^{th} reactant, $p_j \in \mathbf{p}$ the j^{th} product, and coefficients α_i 's and β_j 's specify the stoichiometric amounts. Under the classical chemical kinetic (CCK) model, we assume that the molecules involved in the reaction are of large quantities such that the spatial non-uniformity of molecule distribution is negligible and the intrinsic discrete and stochastic molecular interactions can be safely approximated with continuum and determinism. Specifically the dynamics of the above reaction, with rate constant k , can be described by

$$k \prod_{i=1}^n [r_i]^{\alpha_i} = -\frac{1}{\alpha_i} \frac{d[r_i]}{dt} = \frac{1}{\beta_j} \frac{d[p_j]}{dt},$$

where $[p_j]$ represents the concentration of species p_j . Accordingly the dynamics of a CRN can be characterized by a set of ordinary differential equations (ODEs).

In the sequel, to simplify notation, we do not distinguish a species (treated as a signal) and its concentration (as the non-negative value of the signal) when they are clear from the context.

B. Neuron Model

There are various existing neuron models under different levels of abstraction appropriate for different uses. We adopt the well-known binary neuron model. Under this model, the output of a neuron with n inputs and threshold: $i_1, \dots, i_n, \theta \in \mathbb{R}^+ \cup \{0\}$ is determined by the activation function

$$f(\vec{i}) = \begin{cases} 1, & \text{if } \sum_{j=1}^n w_j i_j \geq \theta, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where $w_j \in \mathbb{R}$ is the corresponding synaptic weight of input i_j . Note that, by implementing neuronal behavior with biochemical reactions, the above step function f is approximated with a *sigmoid* function, which is differentiable thus advantageous in learning such as the backpropagation algorithm.

III. ARCHITECTURE

Similar to FPGAs, our proposed neuromorphic architecture consists of reconfigurable neuron modules and interconnects.

A. Neuron Module

To represent a real-valued signal x , two species x_p and x_n are designated with $x = [x_p] - [x_n]$, similar to [14]. When the input weight w_i of a neuron is positive (resp. negative), $[w_{ip}]$ (resp. $[w_{in}]$) is set to the absolute value of positive (resp. negative) weight and $[w_{in}]$ (resp. $[w_{ip}]$) is set to zero. By interpreting non-negative

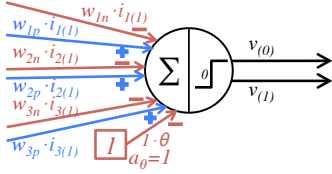
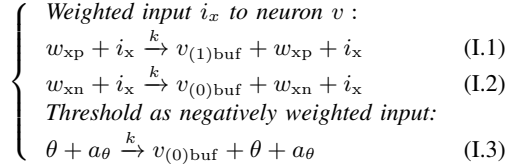


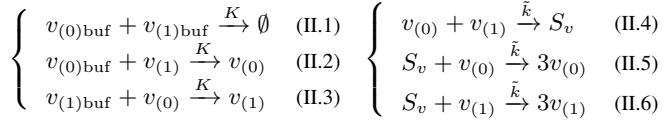
Fig. 1. A neuron with three inputs.

threshold value θ as an auxiliary input $a_{\theta} \equiv 1$ with negative weight equal to $-\theta$, a neuron can always be transformed into an equivalent one with threshold equal to 0, as the one illustrated in Fig. 1. Therefore it suffices to implement a single bistable reaction system for a neuron whose output toggles at the *zero* threshold point. We use the three-input neuron as depicted in Fig. 1 to explain the two main components of CRN implementation listed below:

(I) To compute the weighted sum (represented by the generation rate difference between molecules $v_{(1)\text{buf}}$ and $v_{(0)\text{buf}}$ of inputs (including i_x and the threshold input a_{θ}) for neuron v , we rely on the following reactions, with $x = 1, 2, 3$ for three inputs:



(II) To determine whether the weighted sum exceeds 0, we depend on the bistability created with the following reactions.



We start our discussion from (II). Reactions (II.4)~(II.6) create a bistable system [11] with two *stable* steady states (represented by dual-rail output $(v_{(0)}, v_{(1)}) = (0, 1)$ signifying neuron output 1; $(1, 0)$ signifying neuron output 0) and one *unstable* steady state (at $(v_{(0)}, v_{(1)}) = (0.5, 0.5)$). To decide whether the weighted sum of inputs is larger than zero (i.e., whether the sum of the positively weighted inputs is larger than the absolute value of the sum of the negatively weighted inputs) and to require $(v_{(0)}, v_{(1)}) = (0, 1)$ (resp. $(1, 0)$) when the sum of the positively weighted inputs is larger (resp. smaller) than the sum of the negatively weighted inputs, we establish the correspondence between $v_{(0)}$ (resp. $v_{(1)}$) and *negatively* (resp. *positively*) weighted inputs by the reactions in (I) and (II.1)~(II.3). It should be clarified that reactions (I.1) and (I.2) are *not* an intrinsic part of the module, but rather their presence depends on the existence of their corresponding *interconnects* between modules, as to be detailed in Sec. III-B.

When the weight of the x^{th} input i_x is positive (effectively $w_{xn} = 0$), only the reaction with w_{xp} involved is activated and thus $v_{(1)\text{buf}}$ is generated at rate $(k \cdot w_{xp} \cdot i_x)$. For the y^{th} input i_y with a negative weight, the same reasoning applies and $v_{(0)\text{buf}}$ is generated at rate $(k \cdot w_{yn} \cdot i_y)$. Reaction (I.3) effectively subtracts the threshold value θ from the weighted sum of inputs. With the reactions in (I), the *generation rates* of molecules $v_{(1)\text{buf}}$ and $v_{(0)\text{buf}}$ correspond respectively to the intended sums of the *positively* and *negatively* weighted inputs. Reactions (II.1)~(II.3) then convert the comparison between the *generation rates* of $v_{(1)\text{buf}}$ and $v_{(0)\text{buf}}$ to the comparison between the *concentrations* of $v_{(0)}$ and $v_{(1)}$. Finally, reactions (II.4)~(II.6) enforce the concentrations of $v_{(0)}$ and $v_{(1)}$ at equilibrium stabilize to one of two the stable steady states discussed in the previous paragraph. Note that the conversion

achieved by reactions (II.1)~(II.3) is crucial in *preserving the total number of output molecules* ($[v_{(0)}] + [v_{(1)}]$), so the system does not require constant replenishment of species from outside. The effort not only makes the system more practical, but also avoids deviation of system behavior resulted from inaccurate replenishment.

To guarantee that the ratio of the positively to negatively weighted sums of inputs is the same as the ratio of the generation rate of $v_{(1)\text{buf}}$ to the generation rate of $v_{(0)\text{buf}}$, all the reactions in (I) would require the same rate constant k . This requirement is unrealistic and can be overcome by our engineered reconfigurability [4]. Because the rate of each reaction in (I) can not only be regarded as a function of k but also as a function of $k \times w_p$, $k \times w_n$, or $k \times \theta$ for species w_p , w_n , or θ *unique* to that reaction, we can relax the original rate constant constraint $k_{(I.1)} = k_{(I.2)} = k_{(I.3)} = k$ to $(k_{(I.1)} \times w'_{xp}) = (k_{(I.2)} \times w'_{xn}) = (k_{(I.3)} \times \theta')$, where the primed version w' of w signifies that the value of w' corresponds not exactly to an original input weight as w , but to an input weight adjusted for the purpose of rate matching.

B. Programmable Interconnect

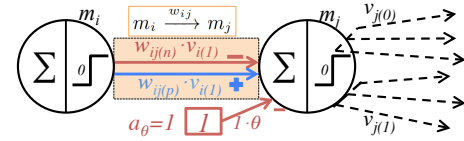
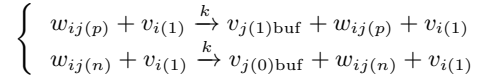


Fig. 2. Interconnect configuration for positive and negative weight. Other interconnects to the neurons are left out for clarity.

Once the set of available neuron modules \mathbf{m} are constructed, a directed interconnect from a module $m_i \in \mathbf{m}$ to another module $m_j \in \mathbf{m}$ with reconfigurable weighting $w_{ij} \in \mathbb{R}$, as shown in Fig. 2, can be realized with the following reactions.



A directed interconnect requires two reactions: one for the positively weighted input and the other for the negatively weighted input. Corresponding to the two reactions are two species ($w_{ij(p)}$, $w_{ij(n)}$) whose concentrations are used to control the weight. Therefore implementing an interconnect costs 2 reactions and 2 species. Note that the reactions for an interconnect from neuron m_i to m_j are designed such that they will not alter the equilibrium of the source module m_i —the concentration of $v_{i(1)}$ is not affected by downstream reactions.

C. Resource Requirements and Scalability

For the required resources, each neuron (interconnect) requires 6 (2) species and 7 (2) reactions. A feedforward neural network with x inputs, z outputs, and one hidden layer of y nodes requires $[(x+1) + 6 \times (y+z)] + [2 \times (xy + yz)]$ species and $[7 \times (y+z)] + [2 \times (xy + yz)]$ reactions. (When working as a classifier, it may classify up to 2^z classes in x -dimensional input space [19]). On the other hand, given any neural network, the mapping of its neurons and interconnects to our architecture is doable in linear time by assigning reaction species of (I.1) (I.2) and (II.1) (II.6) for each interconnect and neuron, respectively.

IV. CASE STUDIES

We perform two case studies mapping classifier and learning applications to our proposed architecture. All synthesized CRNs are first simulated on Biocham [7] for verification, and further

mapped into *two-domain* DSD reactions [1], which are suitable for modularized composition among reactions. The simulation results of the mapped DSD reactions are provided to justify the feasibility of our design.

A. Classification

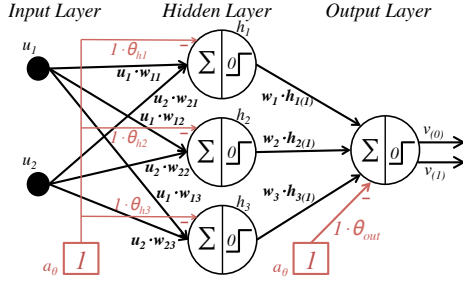


Fig. 3. A feedforward network with one hidden layer.

We demonstrate the reconfigurability of our neuromorphic architecture with the mapping of a classification example. Consider the feedforward network with one input, one hidden, and one output layer as shown in Fig. 3, which implements a classifier that separates the input space spanned by $u_1, u_2 \in \mathbb{R}^+ \cup \{0\}$ into two classes based on whether the criterion below is satisfied:

$$(5u_1 - u_2 \geq 3) \vee [(-u_1 + 2u_2 \geq 1.5) \wedge (u_1 + u_2 \geq 1.5)]$$

(Note that any arbitrary classification task can be achieved by a feedforward neural network with one hidden layer [9], and can be realized in our proposed system by setting corresponding concentrations in the following systematic way, thus *reconfigurable*.) The input layer consists of two inputs u_1 and u_2 ; the output layer requires $\lceil \log_2(\text{number of classes}) \rceil$ neurons. Each neuron in the hidden layer can define a separating hyperplane in the input space, so the number of required neurons equals the number of *distinct* inequalities involved in the criterion specified. Accordingly the parameters of Fig. 3 are assigned as follows:

$$\begin{cases} w_{11} = 5, & w_{21} = -1, & \theta_{h1} = 3 \\ w_{12} = -2, & w_{22} = 4, & \theta_{h2} = 3 \\ w_{13} = 2, & w_{23} = 2, & \theta_{h3} = 3 \end{cases}$$

Each neuron h_i in the hidden layer checks the satisfiability of its corresponding inequality. So for the output layer, the criterion to realize is the Boolean formula $h_1 \vee (h_2 \wedge h_3)$. The last step of the mapping procedure is to transform a logic formula into a linear inequality with binary variables. In this example, one possible assignments is $(w_1, w_2, w_3, \theta_{out}) = (6, 4, 2, 5)$.

Fig. 4(a) shows the *Biocham* simulation result of the corresponding CRN; Fig. 4(b) summarizes the inequalities implemented by each neuron; Fig. 4(c) plots the partition of input space given the classification constraints. Due to space limitation, Fig. 6 only shows part of the DSD reactions mapped from the CRN. The DSD simulation results by *Visual DSD* [12] under *static* inputs $(u_1, u_2) = (0.5, 2)$ and $(0, 0)$ are shown in Fig. 5.

B. Supervised Learning

We justify our claim that autonomous learning ability can be embedded into the proposed biochemical reaction-based module by realizing autonomous weight update. In real-world application, the input vector and *correct* classification result can both be time-series data of species concentrations read from the environment.

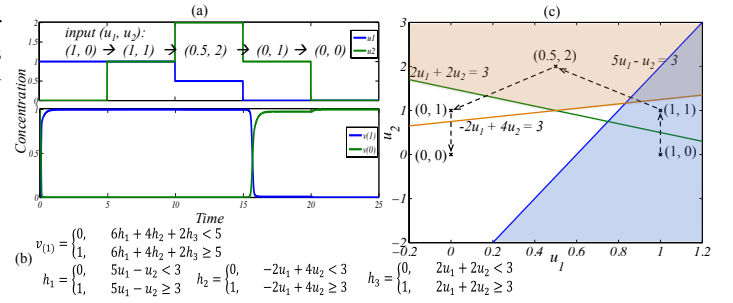


Fig. 4. (a) Simulation result. (b) Functions implemented by each neuron. (c) Input space and separating hyperplanes. The union of the shaded areas defines the set of inputs with output 1; the arrows indicate the trace of (u_1, u_2) in the simulation.

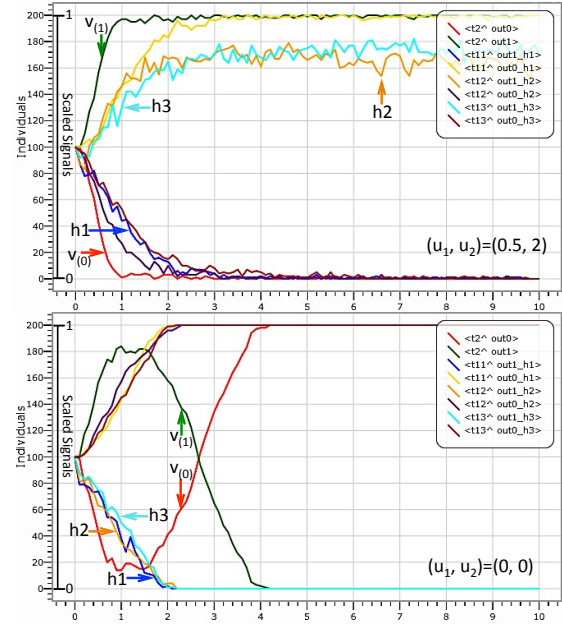


Fig. 5. DSD stochastic simulation results for classification of static inputs.

For example, the input vector can be the concentrations of a set of potential biomarkers for diabetes, and the *correct answer* corresponds to recent statistics of blood glucose value (which can be obtained by cascading a reaction-based, constant-leakage integrator with a neuron whose threshold equals the upperbound of normal value). The system can then be trained into diabetes diagnostic device based on biomarkers.

For clarity, we demonstrate autonomous adaptation by using the *perceptron learning algorithm* [15] to train the composing neuron into a one-dimensional classifier *on positive real* that outputs 0 when the input is smaller than 6, and outputs 1 otherwise. The training pairs of input and its corresponding correct answer are presented as concurrent concentrations to the neuron with the network structure shown in Fig. 7. The threshold value θ of the neuron is *arbitrarily* initialized to 3 and remains fixed; the training target is the input weight represented by its positive and negative components w_p, w_n . Let the input weight be initialized to 2, i.e., $(w_p, w_n) = (2, 0)$. Given our goal, the target training result \widehat{w}_p without changing θ is one that satisfies: $(\widehat{w}_p \times \text{input} > 3) \equiv (\text{input} > 6)$. Hence, our target result is $(\widehat{w}_p, \widehat{w}_n) = (0.5, 0)$. The system keeps comparing its current response with the correct output *continuously in time* as inputs of the training set are fed serially into

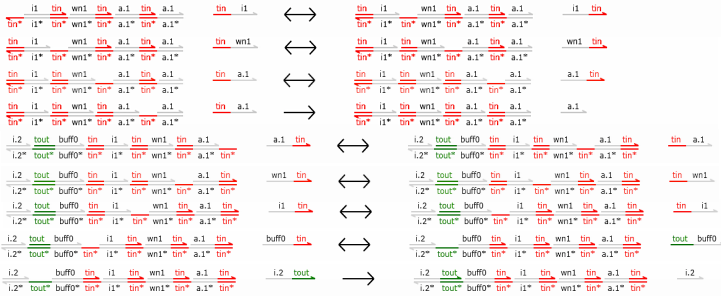


Fig. 6. DSD reactions for a weighted input of a neuron; corresponding to CRN component (I) described in Sec. III-A.

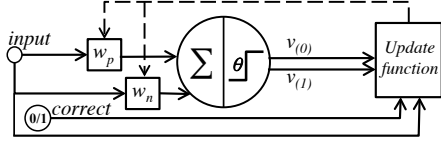


Fig. 7. Structure of the one dimensional classifier to be trained.

the system, and updating the weights according to the formula:

$$w_{i+1} = w_i + \alpha \times \text{input} \times (\text{out}_{\text{correct}} - \text{out}_{\text{real}}),$$

where w_i is the updated weight after the i^{th} training input is fed. The positive α determines the *learning rate* of the system. The impact of an erroneous output on w_i grows faster under higher learning rate. To implement the update function of the perceptron learning algorithm, the following reactions are added to the neural network CRN.

$$\begin{cases} \text{input} + v_{(1)} \xrightarrow{k_{\text{learn}}} \text{input} + v_{(1)} + w_n \\ \text{input} + \text{correct} \xrightarrow{k_{\text{learn}}} \text{input} + \text{correct} + w_p \\ w_n + w_p \xrightarrow{k} \emptyset, \end{cases}$$

where the rate constant k here has value similar to the one in neuron implementation, without particular requirement. The reactions work as follows. When the system's output is correct ($v_{(1)} = \text{correct}$), w_n and w_p are generated in the same rate by the first two reactions, and the impact will be canceled out by the third reaction. Hence the weight value will not be changed. When $v_{(1)} = 1$ but $\text{correct} = 0$, error occurs. The weight's negative component w_n is produced at rate $K_{\text{learn}} = k_{\text{learn}} \times \text{input} \times v_{(1)}$ when no positive component is produced. Combined with the third reaction, the weight is reduced at constant rate K_{learn} for a time period Δt before the next training input comes in. The weight is thus updated by $(K_{\text{learn}} \times \Delta t)$, an increase that conforms with the update rule. Finally, when an error occurs in another direction with $v_{(1)} = 0$ and $\text{correct} = 1$, w_p is produced and no negative component is produced in the same period. Following similar reasoning, the weight is updated by $(-k_{\text{learn}} \times \text{input} \times \Delta t)$, a decrease.

The CRN simulation results of the training process under input series in Fig. 8(a) are shown. Note that the proposed system allows *online* learning, so can be tuned in real-time as the training inputs come in. Fig. 8(b) nicely approximates the correct training result with appropriate learning rate, and the module's output value $v_{(1)}$ conforms better with expected output as training proceeds. Fig. 8(c) and Fig. 8(d) show the system's behaviors when the learning rate is too large or small, leading to oscillation or slow convergence respectively. Since the Visual DSD tool currently does not support simulation for time-varying inputs, we do not show DSD simulation results. However our mapped DSD reactions were verified to have

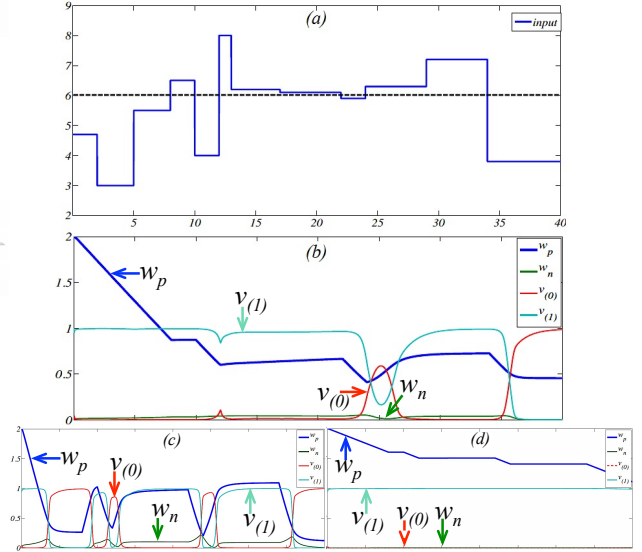


Fig. 8. (a) Training input series. The applied period of each input pattern can vary, as long as the corresponding *desired output* is presented concurrently. (b)-(d) Simulation results under different learning rates k_{learn} : (b) appropriate, under k , (c) too fast, under $3k$ (d) too slow, under $k/3$.

correct weight adjusting behavior given static input pairs.

V. CONCLUSIONS

We proposed a reconfigurable neuromorphic architecture implementable with modularized biochemical reactions. Case studies were performed to demonstrate the supported system reconfigurability and autonomous adaptability. Our method may provide a step forward to system engineering enabling neuromorphic computation in potential biomedical applications.

REFERENCES

- [1] L. Cardelli, "Two-domain DNA strand displacement," *Mathematical Structures in Computer Science*, Vol. 23, pp. 247–271, Feb. 2013.
- [2] Y.-J. Chen et al., "Programmable chemical controllers made from DNA," *Nature Nanotechnology*, Vol. 8, pp. 755–762, Oct. 2013.
- [3] T.-Y. Chiu et al., "Configurable Linear Control of Biochemical Systems," *Proc. International Workshop on Bio-Design Automation*, 2014.
- [4] H.-J. K. Chiang, et al., "Building reconfigurable circuitry in a biochemical world," *Proc. IEEE Biomedical Circuits and Systems Conference*, 2014.
- [5] R. Danial, et al., "Synthetic analog computation in living cells," *Nature*, 2013.
- [6] M. Frahm et al., "Discriminating between latent and active tuberculosis with multiple biomarker responses," *Tuberculosis*, Vol. 91, No. 3, pp. 250–256, 2011.
- [7] François Fages et al., "BIOCHAM 3.5 Reference Manual," Nov. 2013.
- [8] J. Hemphill and A. Deiters, "DNA computation in mammalian cells: microRNA logic operations," *J. Am. Chem. Soc.*, Vol. 135, No. 28, pp. 10512–10518, 2013.
- [9] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, Vol. 4, No. 2, pp. 251–257, 1991.
- [10] A. Hjelmfelt, et al., "Chemical implementation of neural networks and Turing machines," *Proceedings of the National Academy of Sciences*, Vol. 88, No. 24, pp. 10983–10987, 1991.
- [11] H. Jiang, et al., "Digital logic with molecular reactions," *Proc. International Conference on Computer-Aided Design*, pp. 721–727, Nov. 2013.
- [12] M. R. Lakin, et al., "Visual DSD: a design and analysis tool for DNA strand displacement systems," *Bioinformatics*, Vol. 27, No. 22, pp. 3211–3213, 2011.
- [13] WR W. Martin, et al., "Midbrain iron content in early Parkinson disease A potential biomarker of disease status," *Neurology*, Vol. 70, No. 16 Part 2, pp. 1411–1417, 2008.
- [14] K. Oishi and E. Klavins, "Biomolecular implementation of linear I/O systems," *IET Syst. Biol.*, Vol. 5, No. 4, pp. 252–260, 2011.
- [15] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Cornell Aeronautical Laboratory, Psychological Review*, Vol. 65, No. 6, pp. 386–408, 1958.
- [16] D. Soloveichik, et al., "Computation with finite stochastic chemical reaction networks," *Natural Computing*, Vol. 7, No. 4, pp. 615–633, 2008.
- [17] L. Qian, et al., "Neural network computation with DNA strand displacement cascades," *Nature*, Vol. 475, pp. 368–372, 2011.
- [18] A. Wieckowska et al., "In vivo assessment of liver cell apoptosis as a novel biomarker of disease severity in nonalcoholic fatty liver disease," *Hepatology*, Vol. 44, No. 1, pp.27–33, 2006.
- [19] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, Issue. 4, pp. 451–462, Nov. 2000.