Reconfigurable Neuromorphic Computation in Biochemical Systems

Hui-Ju Katherine Chiang, Jie-Hong R. Jiang, and François Fages

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, 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].

II. PRELIMINARIES

A. Model of Chemical Reaction Dynamics

A chemical reaction network (CRN) is composed of a set of reactants \( r \), a set of products \( p \), 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 \( r_i \in r \) is the \( i \)-th reactant, \( p_j \in p \) the \( j \)-th product, and coefficients \( \alpha_i \)'s and \( \beta_j \)'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

\[
\frac{d[r_i]}{dt} = -\frac{1}{\alpha_i} \sum_{j=1}^{m} \beta_j [p_j],
\]

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: \( t_1, \ldots, t_n, \theta \in \mathbb{R}^+ \cup \{0\} \) is determined by the activation function

\[
f(\overrightarrow{w}) = \begin{cases} 
1, & \text{if } \sum_{j=1}^{n} w_{ij} \geq \theta, \\
0, & \text{otherwise}, \end{cases}
\]

where \( w_{ij} \in \mathbb{R} \) 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 \( w_{ij} \) of a neuron is positive (resp. negative), \([w_{ij}]\) (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
threshold value $\theta$ 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 $i_v$ and the threshold input $a_{\theta}$) for neuron $v$, we rely on the following reactions, with $x = 1, 2, 3$ for three inputs:

\[
\begin{align*}
\text{Weighted input } i_v & \text{ to neuron } v: \\
\text{Threshold as negatively weighted input: }
\end{align*}
\]

\[
\begin{align*}
&w_x p + i_v \rightarrow v_{(1)\text{buf}} + w_x p + i_v & (I.1) \\
&w_x n + i_v \rightarrow v_{(0)\text{buf}} + w_x n + i_v & (I.2) \\
&\theta + a_{\theta} \rightarrow k_v v_{(0)\text{buf}} + \theta + a_{\theta} & (I.3)
\end{align*}
\]

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

\[
\begin{align*}
&v_{(0)\text{buf}} + v_{(1)\text{buf}} \rightarrow k_v \theta & (II.1) \\
&v_{(0)\text{buf}} + v_{(1)} \rightarrow v(0) & (II.2) \\
&k_v v_{(1)\text{buf}} + v(0) \rightarrow k_v v_{(1)} & (II.3) \\
&S_v + v(0) \rightarrow 3v(0) & (II.4) \\
&S_v + v(1) \rightarrow 3v(1) & (II.5) \\
&S_v + v(1) \rightarrow 3v(1) & (II.6)
\end{align*}
\]

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 $(v(0),v(1)) = (0,1)$) signaling neuron output 1; $(1,0)$ signaling neuron output 0) and one unstable steady state $(v(0),v(1)) = (0.5,0.5))$. 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 $\theta$ 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 $k \times w_p, k \times w_n, or \theta$ unique to that reaction, we can relax the original rate constant constraint $k_{(1,1)} = k_{(1,2)} = k_{(1,3)} = k \times (k_{(1,1)} \times w_p^2) = k_{(1,2)} \times w_n^2 = k_{(1,3)} \times \theta^2$, 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

A directed interconnect requires two reactions: one for the positively weighted inputs and the other for the negatively weighted inputs. The presence of the module, and the weights they are associated with, are designated 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.

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

\[
\begin{align*}
&w_{ij(p)} + v_{i(1)} \rightarrow v_{j(1)\text{buf}} + w_{ij(p)} + v_{i(1)} \\
&w_{ij(n)} + v_{i(1)} \rightarrow v_{j(0)\text{buf}} + w_{ij(n)} + v_{i(1)}
\end{align*}
\]

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 $[(x+1) + 6 \times (y+z)] + [2 \times (x+y+z)]$ species and $[7 \times (y+z)] + [2 \times (x+y+z)]$ reactions. (When working as a classifier, it may classify up to $2^x$ 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

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:

\[ (5u_1 - u_2 \geq 3) \lor \left( (-u_1 + 2u_2 \geq 1.5) \land (u_1 + u_2 \geq 1.5) \right) \]

(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:

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

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 \( h_1 \lor (h_2 \land h_3) \). 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 assignment 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. 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 upper bound 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 \( \theta \) 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 \( \hat{w}_p \) without changing \( \theta \) is one that satisfies: \((\hat{w}_p \times \text{input} > 3) \equiv (\text{input} > 6)\). Hence, our target result is \((\hat{w}_p, \hat{w}_n) = (0.5, 0)\). The system keeps comparing its current response with the correct output continuously in time as inputs of the training set are fed serially into
the system, and updating the weights according to the formula:

\[
\omega_{i+1} = \omega_i + \alpha \times \text{input} \times (\text{out}_{\text{correct}} - \text{out}_{\text{real}}),
\]

where \(\omega_i\) is the updated weight after the \(i^{th}\) training input is fed. The positive \(\alpha\) determines the learning rate of the system. The impact of an erroneous output on \(\omega_i\) 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:

\[
\begin{cases}
\text{input} + v(1) & \rightarrow k_{\text{learn}} \text{input} + v(1) + w_n \\
\text{input} + \text{correct} & \rightarrow k_{\text{learn}} \text{input} + \text{correct} + w_p \\
w_n + w_p & \rightarrow \emptyset,
\end{cases}
\]

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 \(k_{\text{learn}}\) for a time period \(\Delta t\) 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 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