Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 278735, 11 pages
doi:10.1155/2012/278735

Research Article

A Hardware Design of Neuromolecular Network with Enhanced Evolvability: A Bioinspired Approach

Yo-Hsien Lin[1] and Jong-Chen Chen[2]

[1] *Department of Information Management, Yuanpei University, 306 Yuanpei Street, Hsinchu 30015, Taiwan*
[2] *Department of Information Management, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan*

Correspondence should be addressed to Jong-Chen Chen, jcchen@yuntech.edu.tw

Received 14 July 2011; Revised 30 August 2011; Accepted 30 August 2011

Academic Editor: Jiang Xu

Silicon-based computer systems have powerful computational capability. However, they are easy to malfunction because of a slight program error. Organisms have better adaptability than computer systems in dealing with environmental changes or noise. A close structure-function relation inherent in biological structures is an important feature for providing great malleability to environmental changes. An evolvable neuromolecular hardware motivated by some biological evidence, which integrates inter- and intraneuronal information processing, was proposed. The hardware was further applied to the pattern-recognition domain. The circuit was tested with Quartus II system, a digital circuit simulation tool. The experimental result showed that the artificial neuromolecularware exhibited a close structure-function relationship, possessed several evolvability-enhancing features combined to facilitate evolutionary learning, and was capable of functioning continuously in the face of noise.

1. Introduction

Effective programmability is an important feature inherent in computer systems, including software and hardware, which allows us to explore various problem domains. However, most of computer systems are brittle in the sense that a slight modification of a system's structure can inadvertently change its functions or cause it to malfunction [1]. This is because computer systems possess a mapping structure with fitness landscapes consisting of isolated peaks that are separated by wide, deep valleys. By contrast, organisms possess a mapping structure with fitness landscapes holding some degree of smoothness that a slight change in an organism's gene structure generally will not significantly alter its functions. Finding feasible solutions within a reasonable time may become much easier in a smooth landscape than in a rugged landscape [2]. In biological systems, the smoothness (gradualism) property is naturally represented in the close structure-function relationship.

In the early 1990s, some other researchers concentrated on applying evolutionary techniques to hardware design. They attempted to use a reconfigurable hardware to continually change the internal circuit structure until the desired structure appears. This field was called evolvable hardware (EHW). EHW brought an interdisciplinary integration. One such idea is to combine the merits of biological sys-tems and computer systems together and hopefully create hardware with better adaptability. For example, Sipper and Ronald [3] proposed an FPGA circuit to simulate the glob-al behaviour of a swarm of fireflies. Mange et al. [4] successfully applied evolutionary techniques into the design of a timer (stopwatch) and a full watch (biowatch) with digital circuits. Higuchi and his colleagues [5, 6] worked on the development of a number of evolvable hardware chips for various applications, including an analog chip for cellular phones, a clock-timing chip for Gigahertz systems, a chip for autonomous reconfiguration control, a data compression chip, and a chip for controlling robotic hands. Murakawa et al. [7] applied evolutionary techniques to reconfigure neural network topology, de Garis [8, 9] developed an artificial brain that assembled a group of cellular automata-based neural net modules to control a robot, and Torresen

[10] designed an evolutionary digital circuit to control prosthetic hands.

Our goal is to provide the digital machine with a representation of the internal structure-function relations of biological systems, to capture some of the dynamic modes of the processing of these systems, and to incorporate learning algorithms of the type used in natural systems. Redundancy, weak interactions, and compartmentalization are three important features inherent in biological structures that facilitate evolutionary learning [1]. The proposed system (artificial neuro molecular system, ANM) is a plastic architecture with rich dynamics that combines these three features into the system, in particular into the subneuronal level of processing. We note that redundancy allows an organism to absorb genetic changes and yet wait for other mutations to join together to make a significant change in its phenotypic traits. By virtue of redundancy, several genetic mutations do not have to occur simultaneously. Weak interaction is the other indispensable feature for facilitating evolution. When the interactions among the constituted components of a system are small, adding a component into (or removing it from) a system will not significantly alter its outputs (or functions). This allows a system to stabilize its current state in responding to structural changes or environmental changes. Compartmentalization is another evolution-friendliness feature. It allows a system to block off disturbance or noise in the environment.

2. Architecture of ANM

The ANM model was motivated from the molecular mechanisms inside real neurons. The model consists of two types of neurons: cytoskeletal neurons and reference neurons. Cytoskeletal neurons have significant intraneuronal information processing that might directly or indirectly relate to their firing behavior. They combine, or integrate, input signals in space and time to yield temporally patterned output signals. Reference neurons serve as pointers to other neurons in a way that allows for interneuronal memory manipulation.

In this section, we introduce the intraneuronal architecture that plays the role of integrating spatiotemporal signals inside a neuron and the interneuronal architecture that orchestrates groups of neurons for performing coherent tasks. We then explain the evolutionary learning algorithm used in this model.

2.1. Operation Hypotheses. The model is based on two hypotheses.

H1. There are some brain neurons in charge of the time-space information transition. This kind of neuron is called cytoskeletal neuron. Cytoskeletal neurons are based on the operation hypothesis between the nerve cell cytoskeletons and molecules, producing a time-space input signal and transducing it into a series of time outputs [1, 11].

H2. There are some brain neurons in charge of memory control and neuron group organization. This kind of neuron is called reference neurons. The purpose of reference neurons is to form a common-goal information processing group from cytoskeletal neurons. By the memory screening of reference neurons, each workgroup would have neurons of different internal structures, thus being able to finish the group task [1, 11].

2.2. Intraneuronal Architecture. It has been firmly established by now that information processing inside a neuron is significant. The objective of the present study is not to identify the precise nature of these mechanisms, but rather to capture the working hypothesis that the cytoskeleton serves as a signal integration system. Our model is restricted to the membrane components. In the present implementation, the membrane of the cytoskeleton is abstracted as a macromolecular network (a cytoskeletal network) comprising a number of components capable of initiating, transmitting, and integrating cytoskeletal signals. Our assumption is that an inter-neuronal signal impinging on the membrane of a neuron is converted to an intraneuronal signal (a cytoskeletal signal) transmitting on the cytoskeleton. This process was called "transduction"; therefore, a cytoskeletal neuron could be considered a transducer with a specific structure. Cytoskeletal neurons are platforms of message processing, and they are inspired by the signal integration and memory function of the cytoskeleton.

This research utilized 2D cellular automata (CA) [11–13] to conduct the experiment of cytoskeletal neurons, and the wraparound fashion links were adopted for the CA arrangement. A cytoskeleton has multiple molecule networks of microtubules, microfilaments, and neurofilaments. In order to simulate these networks, we defined three kinds of fibers to make a cytoskeleton type (C-type), and the fibers were named C1, C2, and C3. Each of the cytoskeletal elements will have its own shape, thus forming the cytoskeletal molecule networks. The conformation of each cytoskeletal element is variable; therefore, molecule-mass-like groups may possibly be formed. Different types of cytoskeletal elements have different signal transmission features. C1 has the strongest signal bearing capacity, but it has the slowest transmission speed. C3 has the weakest signal bearing capacity, but it has the fastest speed. C2's performance is between C1 and C3. The illustration of cytoskeletal neurons structure is shown in Figure 1. Each cytoskeletal neuron has its unique cytoskeletal fiber structure. The types of signal flows depend on the different structures and different transmission characteristics. Some signal flow would execute the transduction tasks with a diffusion-like method, sometimes fast and sometimes slow.

When an external stimulus hits a cytoskeletal neuron membrane, it will activate the readin enzyme at that location. The activation will cause a signal flow to transmit along the route of the same cytoskeletal elements. For example, after the on-location (3, 2) readin received the external input, it will transmit the signals to its eight neighbors that have the same cytoskeletal element locations. The illustration shows that it can transmit the signal to C2 at locations (2, 2) and (4, 2). Any cytoskeletal element that receives this kind of signal will do the same, thus forming the phenomenon of

a signal flow. In order to ensure it is a one-way transmission, meaning there will not be any signal backflow or loop formed, the cytoskeletal element will enter a temporal resting state after the transmission. This is called a refractory state. The additional remark is that after a signal was transmitted by a cytoskeletal element, the signal did not disappear immediately within the element. Instead, the signal would decrease progressively until it finally disappeared. The decreasing signal and the new-coming signals would cause a time-space integration reaction, and that is a very important mechanism that decides when a firing will occur.

There could be some interactions among different cytoskeletal fibers. Microtubule-associated proteins (MAPs) have the ability to connect different cytoskeletal fibers, thus causing cross-fiber signal flow channels. This will help the flow of microsubstances within neurons. For instance, when the input signal originated from location (3, 2) goes along the C2 elements of the second column, it will meet an MAP-linked C1 element at location (5, 2). The C2 signal will be transmitted to C1 through MAP, and another signal flow will be formed in C1. However, due to different types of cytoskeletal fibers and different transmission features, there might be some energy transition problems when signals going through different mediums. Hence, regarding the cross-fiber signals, this research defined the signal bearing capacity of C1, C2, and C3 as S, I, and W, meaning strong, intermediate, and weak. Because the linking function provided by MAP allows the signals to flow among different molecule elements, there exist information processing behaviors within the neurons.

When a time-space integrated cytoskeletal signal arrives at a location of a readout enzyme, the activation will lead to a neuron firing. For example, the signal flows started at locations (1, 5) and (8, 7) may be integrated at location (5, 5), and the readout enzyme at that location would be activated, thus causing a neuron firing. Because the integrated cytoskeletal signals may continuously appear, the firing outputs become a series of signals that happened in different time points. This research collected these signals to serve as the reference for transduction efficiency assessments.

2.3. Digital Hardware Design of Cytoskeletal Neuron. In the process of digitalization, each grid in cytoskeletal neuron is called processing unit (PU). Figure 2 shows the conceptual architecture of a PU, including four control parts and four signal processing blocks. The input department controler is responsible for controlling the conversions of the signals arriving at the input department into signals for the process department. The output department controler is responsible for controlling the layout of the information for signals sent out from output department to its neighboring cells. The processor department controller has two purposes. Firstly, it will control the countdown of an accumulator so that its value will degrade at a certain speed. Secondly, it will control the timing of the signals sent from the accumulator to its corresponding bounder, which in turn determines the transmitting speed of a cell. The following explains how to implement signal initiation, transmission, and integration on the cytoskeleton of a neuron.

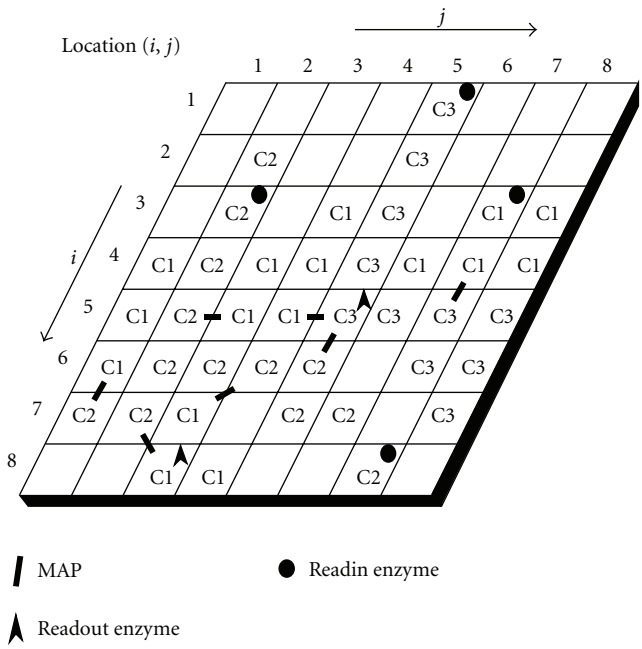


Figure 1: Structure of cytoskeletal neuron.

2.3.1. Signal Initiation. In the present implementation, there are two possible mechanisms to initiate a cytoskeletal signal. One is directly initiated by an external stimulus. When a PU receives an external stimulus and there is a readin enzyme sitting on it, a new cytoskeletal signal is initiated. The other mechanism is combining some specific combinations of cytoskeletal signals in space and time to turn a PU into a highly activated state, which in turn initiates a new signal. Each PU processes the signals sent from its eight neighboring PUs through the input block (Figure 3). We note that a PU will change its state when it receives a signal from its neighboring PU (through MAP). Different types of cytoskeletal signals are initiated and transmitted by different types of PUs. We assume that there are four possible PU types: C1, C2, C3, and none. The first three represent different types of cytoskeletal components for transmitting signals (i.e., different signal flows) whereas the last one represents the lack of a component. In Figure 3, a signal from a C1-type, C3-type, and C2-type neighboring PU is labeled as S, W, and I, respectively.

2.3.2. Signal Transmission. The following explains how to implement signal transmission on an 8×8 grid of PUs. We assume that signal transmission occurs through the neighboring PUs of the same type. An activated PU will activate its neighboring PUs of the same type at its next time step, which in turn activates its neighboring PUs of the same type at the following next time step. This process continues as long as there is a neighboring PU belonging to the same type. To assure unidirectional signal transmission, an activated PU will enter a refractory state. The refractory period depends on the update time of each PU type (to be described in the next section). It will then go back to the quiescent state after the refractory period is over. A PU in the refractory state

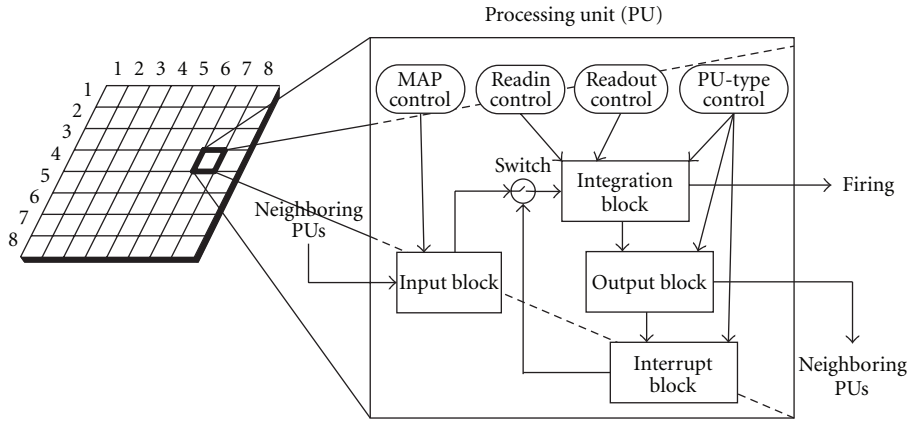


Figure 2: Conceptual architecture of a PU. The input block is illustrated in Figure 3, the interrupt block in Figure 4(a), the output block in Figure 4(b), and the integration block in Figure 6.

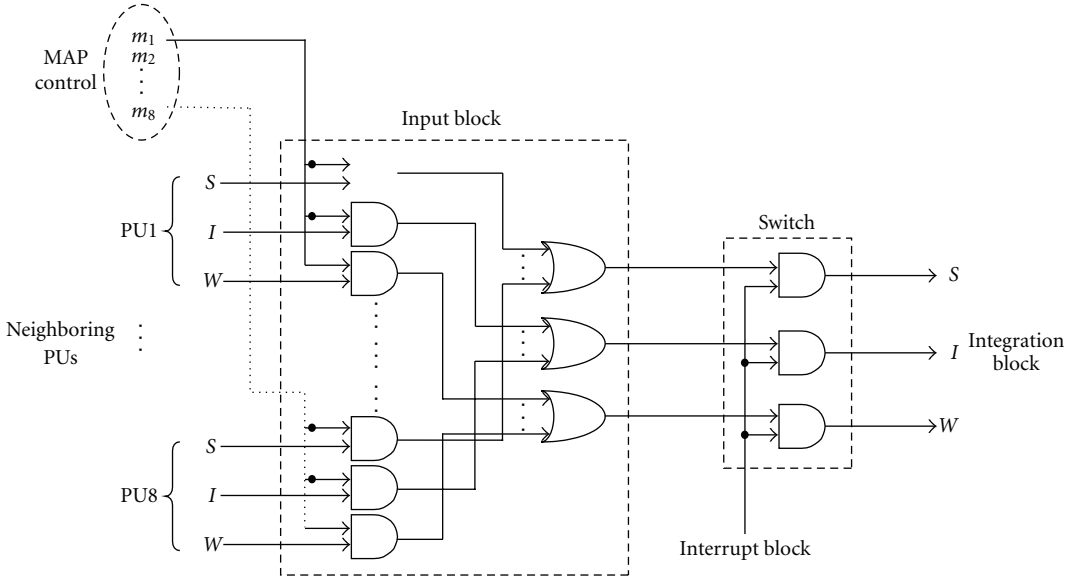


Figure 3: Conceptual architecture of the input block.

will ignore any stimuli during the refractory period. This would prevent a signal from bouncing repeatedly between two neighboring PUs. We note that the refractory time is an important parameter that may affect the performance in the present hardware design. That is, different output is possible when length of the refractory time of each type is varied. But our goal at this stage is to fulfill the function that a PU serves as a signal integrator that combines different signals in space and time (the detail will be described in the next section). That is, the refractory time is fixed and will not be involved in the evolutionary change. In the present implementation, we have not performed a systematical experiment along this line. But it would be interesting to perform the experiment in the near future.

A switch controlled by the interrupt block is used to regulate the signal flow from the input block to the integration block (Figure 4(a)). The timing of control is through the output block (Figure 4(b)). When a PU is in the state of being ready to take any signals from its neighboring PUs, its output block will turn on the switch (through the interrupt block) by sending it a high-voltage signal. This indicates that any signal from its neighboring PU is allowed to change the state of a PU. However, the switch will be turned off if a PU is either in the state of processing a neighboring signal or in the refractory state.

2.3.3. Signal Integration. As mentioned earlier, there are three types of PUs for transmitting signals. Our implementation of signal integration is that, to fire a neuron, it requires at least two different types of signals to rendezvous at a PU within a short period of time. In other words, a PU serves as a signal integrator that combines different signals in space and time. To capture this feature, two hypotheses are used. The first is that different PU types have different transmission speeds. The second hypothesis is that an activated PU can influence the state of its neighboring PU through MAP

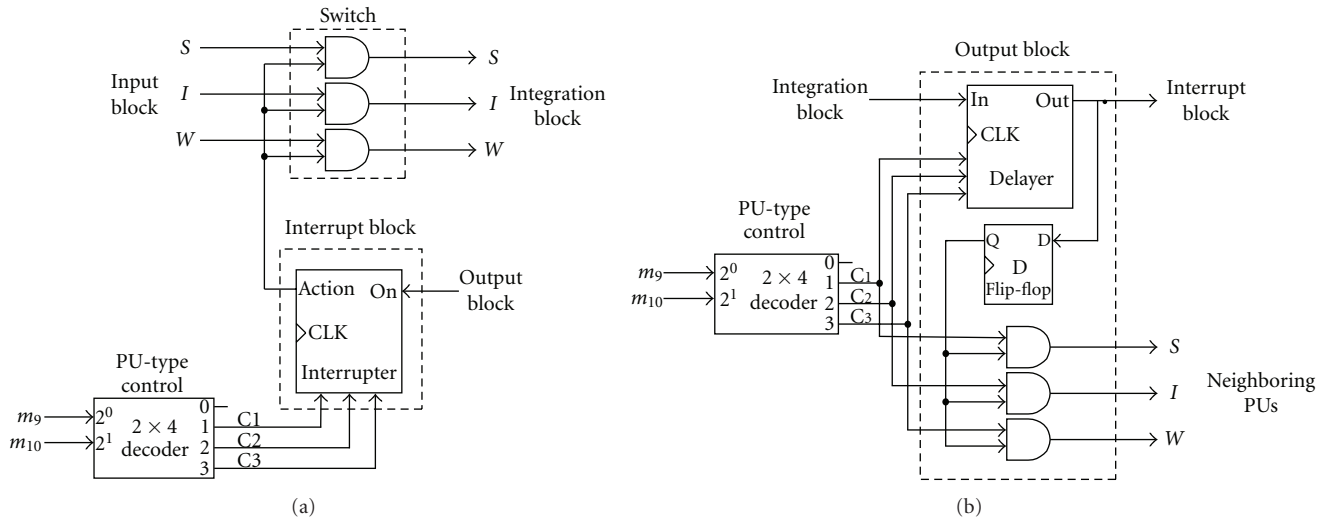


Figure 4: The interrupt block (a) and the output block (b).

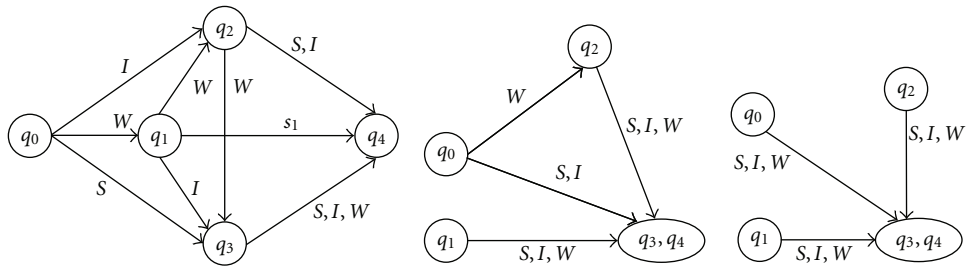


Figure 5: Transition rules of a PU. S, I, and W indicate a signal from a highly activated C1-, C2-, and C3-type PU, respectively. For example, if C1-type PU in the state q_0 receives an I signal it will enter the moderately activated state q_2. If it receives a W signal it will enter the more activated state q_3. For example, a C1-type PU in the state q_0 will enter the less active state q_1 when it receives a signal from a neighboring C3-type PU (i.e., signal W). We note that if a PU does not receive a stimulus before its next update time, it will go into state q_1 if it was in state q_2, or enter state q_0 if it was in q_1.

linking them together. That is, the latter will make a state transition when it receives a signal from the former.

We assume that a PU has six possible states: quiescent (q_0), active with increasing levels of activity (q_1, q_2, q_3, and q_4), and refractory (q_r). Certainly, the complexity of intraneuronal dynamics will be greater when a larger number of PU states are allowed. Correspondingly, it will increase the complexity of the hardware design. We note that six states are sufficient for present use. The following describes the transition rules of each PU. A PU in the highly active state (q_3 or q_4) will return to the refractory state (q_r) at its next update time, and then go into the quiescent state (q_0) at the following next update time. The next state for a less active PU (q_0, q_1, or q_2) depends on the sum of all stimuli received from its active neighboring PUs (Figure 5). If a PU does not receive a stimulus before its next update time, it will go into state q_1 if it was in state q_2, or enter state q_0 if it was in q_1.

In the present implementation, a signal traveling along C1-type PUs has the slowest speed, but also has the greatest degree of influence on the other two PU types. In contrast, a signal traveling along C3-type PUs has the fastest speed, but also has the least degree of influence on the other two PU types. The speed and the degree of influence of a C2-type signal are between those of a C1- and C3-type signal. We note that the degrees of influence between two different PU types are asymmetrical. For example, a C1-type PU in the state q_0 will enter the less active state q_1 when it receives a signal from a neighboring C3-type PU (i.e., signal W). By contrast, a C3-type PU in the state q_0 will enter a highly activated state q_3 if it receives a signal from a neighboring C1-type PU (i.e., signal S). Roughly speaking, the signal with the greatest degree of influence serves as the major signal flow in a neuron while the other two types of signals provide modulating effects.

The integration block is the major component of the ANM design that integrates signals transmitting in space and time (Figure 6). The comparator is responsible for processing either an external signal linked with its application domain or a cytoskeletal signal from its neighboring PU. For each external signal sent to a PU, we assume that the latter will directly go into a highly active state (q_3) if there is a readin enzyme sitting at the same site. This allows the initiation of a new cytoskeletal signal. As to a signal sending from its

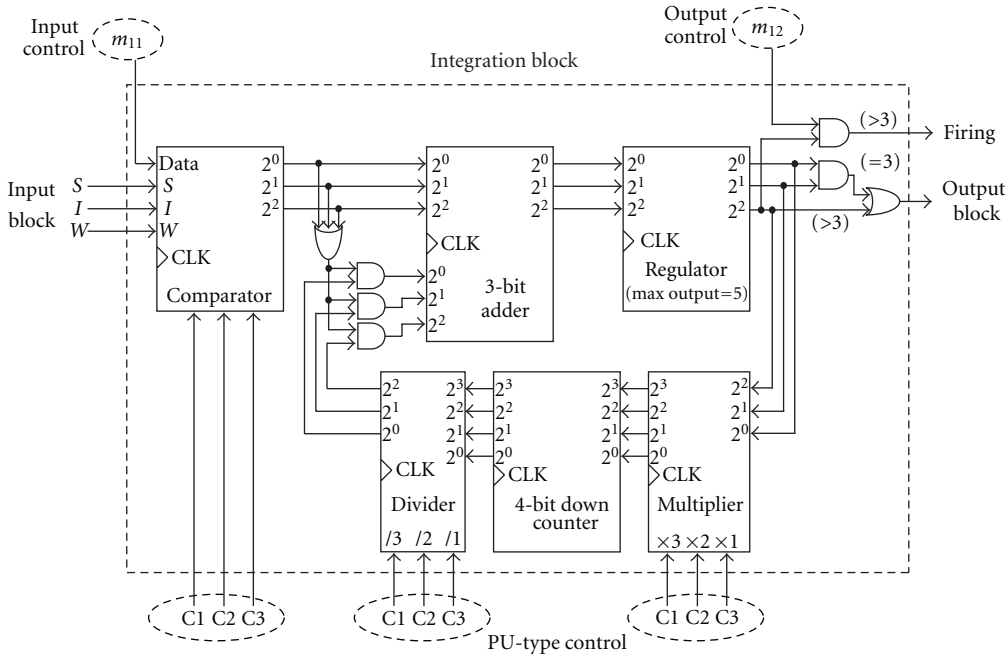


Figure 6: Conceptual architecture of the integration block.

neighboring PU, the comparator determines the state change of a PU. The adder serves as a recorder that keeps a record of a PU's present state. Different numbers represent different states. Different types of PUs have different transmission speeds. The present implementation is that the ratio of transmission speeds of C1-, C2-, and C3-type PUs is 1 : 2 : 3. To comply with the transmission speeds, the ratio of state update times of C1-, C2-, and C3-type PUs is 3 : 2 : 1. Our hardware implementation of speed control comprises a multiplier, a downcounter, and a divider. The multiplier magnifies a signal sent from the regulator and generates an output for the downcounter. The degree of magnification is determined by its update time. For each clock, the downcounter decreases by one. The divider restores the magnified signal in the downcounter back to its original range.

All of the digital circuit modules were design with Verilog using Quartus II software, a digital circuit design tool developed by the Altera Corporation (San Jose, CA). The final design of circuits were downloaded into an FPGA device which produced by the Altera Corporation.

2.4. Interneuronal Architecture-Orchestral Learning. The reference neuron scheme is basically a Hebbian model, in which the connection between two neurons is strengthened when they are active simultaneously. This model also has a hierarchical control feature. With this feature, reference neurons are capable of assembling cytoskeletal neurons into groups for performing specific tasks. Orchestration is an adaptive process mediated by varying neurons in the assembly which selects appropriate combinations of neurons to complete specific tasks. Currently, cytoskeletal neurons are divided into a number of comparable subnets. By comparable subnets, we mean that neurons in these subnets are similar in terms of their inter-neuronal connections and intraneuronal structures. Neurons in different subnets that have similar inter-neuronal connections and intraneuronal structures are grouped into a bundle.

Two levels of reference neurons are used to manipulate these bundles of neuron. The two levels form hierarchical control architecture (Figure 7). The first is referred to as the low-level reference neurons that directly control the bundles of cytoskeletal neurons. Each of these controls a specific bundle (we note that only the bundles activated by the reference neurons are allowed to perform information processing). The second level is referred to as the high-level reference neurons that play the role of grouping the low-level reference neurons. The activation of a high-level reference neuron will fire all of the low-level reference neurons that it controls, which in turn will activate some of these bundles of cytoskeletal neurons (i.e., neurons in different subnets that have similar intraneuronal structures). For example, when R_2 fires, it will fire r_1 and r_{32}, which in turn causes E_1 and E_{32} in each subnet to fire.

The connections among low-level reference neurons and cytoskeletal neurons are fixed. However, the connections between high-level reference neuron and low-level reference neuron layers are subjected to change during evolutionary learning. The above process is called orchestral learning.

2.5. Evolutionary Learning. Processing units are responsible for transmitting and integrating cytoskeletal signals. Evolution at the level of PU configurations is implemented by copying (with mutation) the PU configurations of neurons in the best-performing subnets to those of comparable neurons in the lesser-performing subnets. Variation is implemented by varying the PU configurations during the copy procedure.

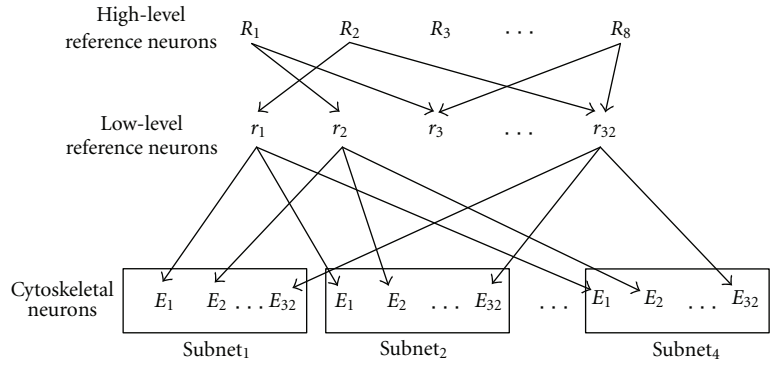


Figure 7: Hierarchical inter-neuronal control architecture.

(1) **Generate** at random the initial *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the reproduction subnet. Each neuron is denoted by neuron (s, b) where s is the subnet number and b is the bundle number.
(2) **Copy** the *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the reproduction subnet to those of comparable neurons in the competition subnets.

$$\text{Copy neuron } (5, b) \text{ to } \left\{ \begin{array}{c} \text{neuron } (1,b) \\ \text{neuron } (2,b) \\ \text{neuron } (3,b) \\ \text{neuron } (4,b) \end{array} \right\}, \text{ for } b = 1, 2, \ldots, 32$$

(3) **Vary** the *MAP* pattern of each neuron in the first subnet, the *PU-type* pattern in the second subnet, the *readin enzyme* pattern in the third subnet, and the *readout enzyme* pattern in the fourth subnet.

$$\text{Vary} \left\{ \begin{array}{c} \text{the } MAP \text{ pattern of neuron } (1,b) \\ \text{the } PU\text{-}Type \text{ pattern of neuron } (2,b) \\ \text{the } readin\ enzyme \text{ pattern of neuron } (3,b) \\ \text{the } readout\ enzyme \text{ pattern of neuron } (4,b) \end{array} \right\}, \text{ if } U \leq P, \text{ for } b = 1, 2, \ldots, 32$$

where P is the mutation rate and U is a random number generated between 0 and 1.
(4) **Evaluate** the performance of each competition subnet and select the best-performing subnet.
(5) **Copy** the *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the best-performing subnet to those of comparable neurons in the reproduction subnets, if the former shows better performance than the latter.
(6) **Go to Step2** unless the stopping criteria are satisfied.

Algorithm 1: Evolutionary learning algorithm.

We note that different PU configurations exhibit different patterns of signal flows.

In the present implementation, the ANM system has 256 cytoskeleton neurons, which are divided into eight comparable subnets. As we mentioned earlier, comparable subnets are similar in terms of their inter-neuronal connections and in-traneuronal structures. Thus, they also can be grouped into 32 bundles. The copy process occurs among neurons in the same bundle. The initial patterns of readin enzymes, readout enzymes, MAPs, and PU-types of the reproduction subnet are randomly decided. That is, the initial value of each bit is randomly assigned as 0 or 1. The evolutionary learning algorithm is shown in Algorithm 1. Note that the mechanism controlling the evolutionary process does not have to be so rigid. Instead, there are several possible alternatives to train this system. In this study, we simply pick out one of these alternatives and precede our experiments. In the future it would be interesting to investigate the impacts of varying the number of learning cycles assigned to each level and the level opening sequence on the learning.

Evolution of reference neurons is implemented by copying (with mutation) the patterns of low-level reference neuron activities loaded by the most fit high-level reference neurons to less fit high-level reference neurons (details can be found in [14]). The copying process is implemented by activating a most fit high-level reference neuron, which in turn reactivates the pattern of low-level reference neuron firing. This pattern is then loaded by a less fit high-level reference neuron. Variation is implemented by introducing noise into the copying process. Some low-level reference neurons activated by a most fit high-level reference neuron may fail to be loaded by a less fit high-level reference neuron. Or some low-level reference neurons that are not activated may fire and be "mistakenly" loaded by a less fit high-level reference neuron. In the present implementation, evolutionary learning at the reference neuron level is turned

off, as we have not yet implemented it on digital circuits. The realization of the evolutionary learning on digital circuit at the reference neuron level is definitely a must-do step, but undoubtedly a complicated job. But in the present implementation, we are not ready to fulfill the design and prefer to focus our study on the internal dynamics, instead of the interneuronal dynamics.

3. Input/Output Interface and Application Domain

We applied the chip to the IRIS dataset, one of the best known datasets found in the pattern recognition literature. The dataset was taken from the machine learning repository at the University of California, Irvine. The dataset contains 3 classes (Iris Setosa, Iris Versicolour, Iris Virginica) of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other. There are four parameters in each instance: sepal length, sepal width, petal length, and petal width.

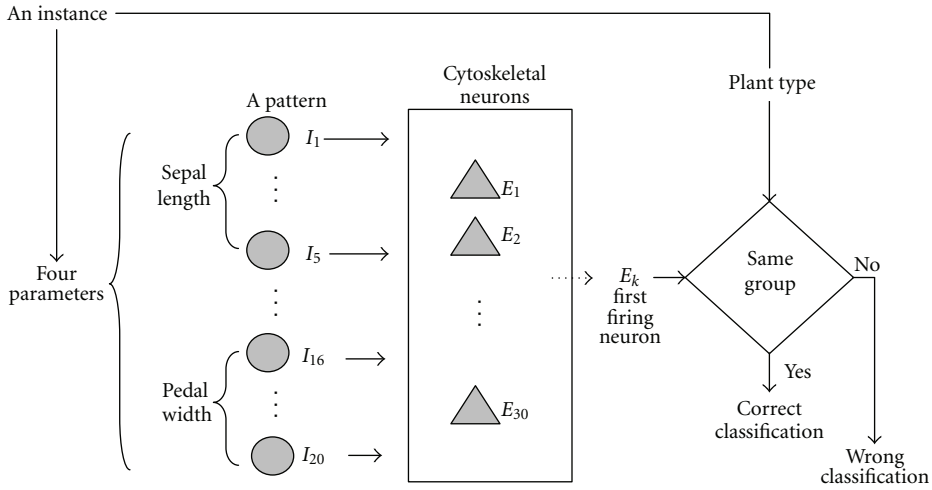
The initial connections between these 4 parameters and cytoskeletal neurons were randomly decided, but subject to change as learning proceeded. Through evolutionary learning, each cytoskeletal neuron was trained to be a specific input-output pattern transducer. That is, each of these neurons became responsible for processing only a small subset of stimuli generated from these 4 parameters. We used five bits to encode each of these 4 parameters. In total, there were 20 bits required to encode all of them. For each parameter, the minimal and maximal values of these 150 instances were determined (to be denoted by MIN and MAX, resp.), and the difference between these two values was divided by 5 (to be denoted by INCR). The transformation of each actual parameter value (to be denoted by ACTUAL) into the corresponding 5-bit pattern was shown to be

$$\begin{aligned}
&00001, \quad \text{if} \quad \text{MIN} \le \text{ACTUAL} < (\text{MIN} + \text{INCR}),\\
&00010, \quad \text{if} \quad (\text{MIN} + \text{INCR}) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 2),\\
&00100, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 2) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 3),\\
&01000, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 3) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 4),\\
&10000, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 4) \le \text{ACTUAL} \le \text{MAX}.
\end{aligned} \tag{1}$$

Each bit corresponded to a specific pattern of stimuli for cytoskeletal neurons. All cytoskeletal neurons that had connections with a specific bit would receive the same pattern of stimuli simultaneously. When a readin enzyme received an external stimulus, a cytoskeletal signal was initiated. It was randomly decided in the beginning and subject to change during the course of learning as to which readin enzymes of a neuron would receive the stimuli from a parameter. For each instance, all stimuli were sent to cytoskeletal neurons simultaneously. In other words, all cytoskeletal signals were initiated at the same time. The cytoskeleton integrated these signals in space and time. For each instance, the class of the first firing cytoskeletal neuron was assigned as its output. Cytoskeletal neurons were equally divided into three classes, corresponding to these three different groups of instances. For each instance, we defined that the chip made a correct response when the class of the first firing neuron was in accordance with the group shown in the dataset (Figure 8). The ANM design was tested with each of these 150 instances in sequence. The greater the number of correct responses made by the chip, the higher its fitness.

4. Experimental Results

4.1. Evolvability. The proposed hardware architecture incorporated several parameters PU, MAP, readin, and readout (denoted as P, M, I, and O, resp.) that allowed us to turn them on or off independently for evolutionary learning. We first study the manner in which problem-solving capability (or evolvability) depended on each level of parameter changes. And then we investigated the effects of increasing the number of evolutionary learning parameters (levels) opened for evolution. For each case of parameter changes (to be described later), five runs were performed. For each run, 100 out of these 150 instances in the IRIS set were selected at random as the training set whereas the remaining 50 instances were grouped as the testing set. All the results reported below were the average differentiation rates of five runs. We first trained the chip for 1200 cycles (at which point learning appeared to slow down significantly). Then, the chip after learning for 1200 cycles was tested with the testing set. The IRIS dataset includes 150 instances. In the evolvability experiment, the dataset were performed for five runs. For each run, the IRIS dataset was randomly divided into two parts: training set and testing set. The training set includes 100 instances and testing set 50 instances. We trained the chip for 1,200 cycles per run. It takes about 30 seconds to complete a learning cycle. The total running time of this experiment takes about 50 hours (30 s $*$ 1200 cycles $*$ 5 runs). When compared with BP neural network and SVM, the time needed to perform the experiment with our system is much longer than we expect. This is because the whole system has been simulated in a computer that simulation has to be performed in a step-by-step manner. When the hardware design has been totally realized with a real digital chip, it might take only microsec-onds to accomplish the assigned tasks.



($E_{1–10}$: Iris Setosa; $E_{11–20}$: Iris Versicolour; $E_{21–30}$: Iris Virginica)

Figure 8: Interface of the ANM design with the IRIS dataset.

To investigate the significance of each parameter, we first allowed only one parameter to change during the course of learning whereas evolution at the other three levels was turned off. In total, there were four experiments performed. Among these four parameters, the chip after learning when only readin enzymes were allowed to evolve alone achieved the highest recognition rate (i.e., 91.6%), when only PU types were allowed the second highest (i.e., 90.0%), when only MAPs were allowed the third highest (i.e., 84.4%), and when only readout enzymes were allowed the lowest rate (83.6%). This provided us some preliminary information about which levels of parameter changes might be friendlier to evolution than others.

The following experiment was to study the manner in which problem-solving capability (or evolvability) depended on different combinations of parameter changes. We first increased the number of parameter changes to two (i.e., two parameters were allowed to evolve simultaneously). There were six combinations of two parameter changes. As shown in Figure 9, the chip after learning when two levels were allowed to evolve simultaneously achieved higher recognition rates than when only one level was allowed to evolve alone. For example, the recognition rate was higher when PUs and MAPs were allowed to evolve simultaneously than when either PUs or MAPs were allowed to evolve alone. This implied that each level of parameter changes more or less contributed in facilitating evolutionary learning, and that synergy occurred among different levels of learning. We then performed the experiment that allowed three parameters to evolve at the same time. There were four possible combinations when three parameters were allowed to evolve simultaneously. The chip after learning when PMI (PU, MAP, and readin enzyme) were allowed to change at the same time achieved the best recognition rate (94.0%) among these four possible combinations (note that a parameter that was not opened for evolutionary changes will be held constant during the course of learning). The implication was that synergies among different levels of evolution became more important as more levels of parameter changes were allowed; implying that learning at one level opened up opportunities for another. However, it did not necessarily mean that learning with more levels of evolutionary changes could always achieve effective performance, in particular when the limitation of learning time was imposed. This was because the power of a multilevel system was not attained by simply summing up the contributions of each constituting element together, but by developing the synergy that might occur among the interactions of different levels. Learning with more levels of evolutionary changes would enhance the repertoire of the system, but did not guarantee that effective learning could be achieved with a limited amount of time. When we looked into what levels (operators) of evolution contributed to the learning progress and how the interactions occurring between different levels exerted control over the tempo of evolution, the result showed that each operator more or less contributed to learning progress, and that learning proceeded in an alternate manner. That is, synergies percolated through different combinations of evolutionary learning operators, implying that learning at one level opened up opportunities for another. We noted that synergy was more likely to occur when a comparatively small number of parameter changes was involved. However, synergy occurred only in a selective manner when more parameter changes were involved.

4.2. Comparison with Other Neural Models. For comparison we applied SVM and BP to the same training and testing sets. As above, five runs were performed. In the former model the average differentiation rates of the training and testing sets were 95.6% and 91.9%, respectively, while in the latter were 96.7% and 91.7, respectively. The above result suggested that the chip had performance comparable to either BP or SVM (Table 1).

4.3. Noise Tolerance. This experiment was to test the capability of the ANM design after substantial learning in tolerating

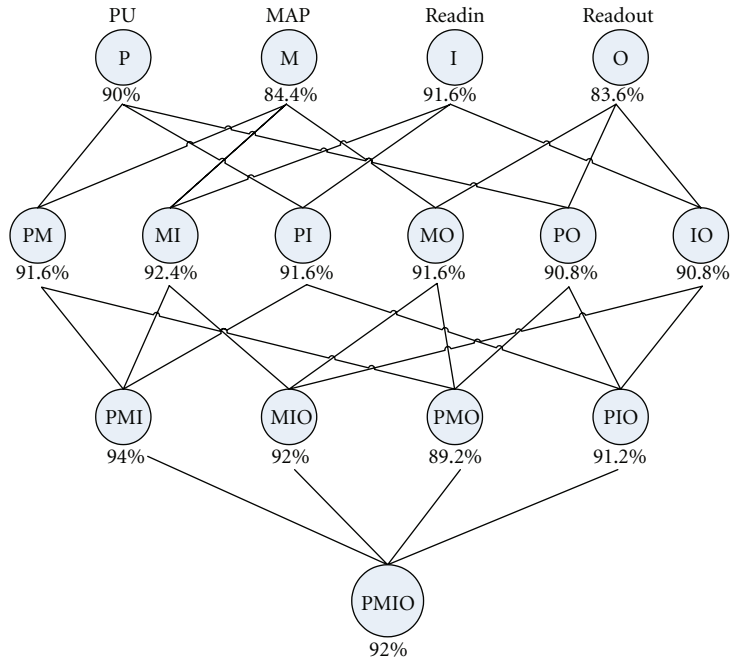


Figure 9: Learning performance of each learning mode.

Table 1: Average differentiation rates of different models with the Iris dataset.

Model	Training	Testing
BP neural network	96.7%	91.7%
Support vector machine	95.6%	91.9%
ANM	94.8%	94.0%

Table 2: Effect of increasing the degree of noise on the rate of the TCD value growth.

Level of variations imposed	5.0%	10.0%
Rate of the TCD value increased	5.7%	11.1%

noise. Through gradually increasing the degree of noise, we observed the input/output relationship of the ANM design. For each test, we kept the system's structure unchanged but varied the pattern of signals sent to cytoskeletal neurons. If the system's outputs changed gradually with the extent of the increase in pattern variations, this in part supported that the system's structure embraced some degree of noise tolerance capability. The ANM design trained for 1,200 cycles was used.

In the following, we first tested the system with spatial noise imposed on the training patterns. To generate a test set, we made a copy of the training set, but altered some bits during the copy process (changing a bit into "1" if it was "0" and into "0" if it was "1"). Five levels of variations were imposed during the copy process: 5%, 10%, 15%, 20%, and 25%. For example, at the 5% level of variations, we mean that each bit has a 5% possibility of being altered. For each level of variations, ten test sets were generated. The total clock difference (TCD) value is the measure pointer indicates the difference of firing time in the circuit. The TCD value increased slightly as we imposed a 5% level of variations on the patterns. Even when we increased the variation level to 25%, the system still demonstrated acceptable results. The TCD value was increased from 38 to 310 at the 5% level of variation (i.e., increased 272), to 569 at the 10% level of variation (i.e., increased 531), to 633 at the 15% level of variation (i.e., increased 625), to 626 at the 20% level of variation (i.e., increased 588), and to 1030 at the 25% level of variation (i.e., increased 992). If we divided the increments by the TCD value before learning (i.e., 4781), the rate of the TCD values increased gradually as we augmented the noise levels (Table 2). This implied that the system had good noise tolerance capability in dealing with spatial noise.

5. Conclusions

When a system is running in the real world, it is inevitable to be confronted with noise generated either from the environment or the system itself. When noise is made only temporarily, structural changes of a system may not be necessary (i.e., a system may ignore this noise). By contrast, a system is required to alter its structure in responding to this noise if it leads to a permanent change in the environment. In such a case, a system has to learn in a moving landscape when environmental change occurs from time to time. Two main results are obtained in the noise tolerance experiment. One is that learning is more difficult in a noisy environment than in a noiseless environment, and that the system is able to learn continuously when noise is made in a temporary manner. The other result is that the system demonstrates

a close structure/function relation. We note that noise occurring at different levels of the cytoskeletal structure has different degree of influence on its outputs. As we gradually modify the structure, its outputs change accordingly. On the other hand, we examine the system's outputs by gradually imposing noise in space and time on its input patterns. The output changes gradually (i.e., proportionally) as the degree of noise is increased. An interesting result is that its system's output does not necessarily change accordingly as we increase the degree of noise generated in time. Note that delaying a signal may alter a neuron's firing activity. However, this may not be true when several signals are delayed simultaneously as these signals may integrate at a later time (undoubtedly, this will delay its firing timing). The above results demonstrate that this system has good noise tolerance capability in dealing with spatiotemporal changes in its inputs, implying that it possesses an adaptive surface that facilitates evolutionary learning. With this feature, the ANM design can be applied to various real-world problems. We note that the ability to separate patterns is clearly a prerequisite for pattern recognition. However, it is equally important to recognize a family of patterns that are slightly varied in space and time. If a system is trained on a particular training set, any ability that it has to respond correctly to noise induced variations in this set will be a form of generalization. The manner of generalization depends on its integrative dynamics (i.e., the flow of signals in the cytoskeleton). This is directly or indirectly influenced by a neuron's PU configuration. Generally speaking, the input patterns recognized by a neuron with internal dynamics will be generalized in a more selective way than simple threshold neurons.

Acknowledgment

This paper was supported in part by the R.O.C. National Science Council (Grant NSC 98-2221-E-224-018-MY3).

References

[1] M. Conrad, "Bootstrapping on the adaptive landscape," *BioSystems*, vol. 11, no. 2-3, pp. 167–182, 1979.

[2] M. Conrad, "The geometry of evolution," *BioSystems*, vol. 24, no. 1, pp. 61–81, 1990.

[3] M. Sipper and E. M. A. Ronald, "A new species of hardware," *IEEE Spectrum*, vol. 37, no. 3, pp. 59–64, 2000.

[4] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, "Toward robust integrated circuits: the embryonics approach," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 516–540, 2000.

[5] T. Higuchi, M. Iwata, D. Keymeulen et al., "Real-world applications of analog and digital evolvable hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 220–234, 1999.

[6] T. Higuchi and N. Kajihara, "Evolvable hardware chips for industrial applications," *Communications of the ACM*, vol. 42, no. 4, pp. 60–66, 1999.

[7] M. Murakawa, S. Yoshizawa, I. Kajitani et al., "The GRD chip: genetic reconfiguration of DSPs for neural network processing," *IEEE Transactions on Computers*, vol. 48, no. 6, pp. 628–639, 1999.

[8] H. de Garis, "An artificial brain ATR's CAM-Brain Project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell Cellular Automata Machine," *New Generation Computing*, vol. 12, no. 2, pp. 215–221, 1994.

[9] H. de Garis, "Review of proceedings of the first NASA/Dod workshop on evolvable hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 304–306, 1999.

[10] J. Torresen, "A divide-and-conquer approach to evolvable hardware," in *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware*, vol. 1478 of *Lecture Notes in Computer Science*, pp. 57–65, Lausanne, Switzerland, 1998.

[11] M. Conrad, R. R. Kampfner, K. G. Kirby et al., "Towards an artificial brain," *BioSystems*, vol. 23, no. 2-3, pp. 175–218, 1989.

[12] J. C. Chen and M. Conrad, "A multilevel neuromolecular architecture that uses the extradimensional bypass principle to facilitate evolutionary learning," *Physica D*, vol. 75, no. 1–3, pp. 417–437, 1994.

[13] S. Rasmussen, H. Karampurwala, R. Vaidyanath, K. S. Jensen, and S. Hameroff, "Computational connectionism within neurons: a model of cytoskeletal automata subserving neural networks," *Physica D*, vol. 42, no. 1–3, pp. 428–449, 1990.

[14] I. Baradavka and T. Kalganova, "Assembling strategies in extrinsic evolvable hardware with bidirectional incremental evolution," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2610, pp. 276–285, 2003.