A Multi-cellular Developmental Representation for Evolution of Adaptive Spiking Neural Microcircuits in an FPGA

Hooman Shayani
Department of Computer Science
UCL (University College London)
London, UK
h.shayani@cs.ucl.ac.uk

Peter J. Bentley
Department of Computer Science
UCL (University College London)
London, UK
p.bentley@cs.ucl.ac.uk

Andrew M. Tyrrell
Department of Electronics
University of York
York, UK
amt@ohm.york.ac.uk

***Abstract*—It has been shown that evolutionary and developmental processes can be used for emergence of scalability, robustness and fault-tolerance in hardware. However, designing a suitable representation for such processes is far from straightforward. Here, a bio-inspired developmental genotype-phenotype mapping for evolution of spiking neural microcircuits in an FPGA is introduced, based on a digital neuron model and cortex structure suggested and verified previously by the authors. The new developmental process is based on complex multi-cellular protein-protein and gene-protein interactions and signaling. Suitability of the representation for evolution of useful architectures and its adaptability is shown through statistical analysis and examples of scalability, modularity and fault-tolerance.**

***Keywords*-spiking neural networks; neurodevelopment; fault-tolerance; evolvable hardware; embryonic hardware; morphogenesis;**

I. Introduction

The idea of evolving a small adaptable, robust, fault-tolerant and intelligent brain in silicon suitable for a given class of problems has been around for more than a decade. Numerous different methods to evolve artificial neural networks [1] have been introduced, which were successful in creating intelligent systems to some extent. However, oversimplification of natural processes in these methods may have impaired the systems. Inclusion of some of the complexities of these natural processes has been shown to lead to higher performance and emergence of new capabilities. For instance, the higher computation power of spiking neural microcircuits compared to traditional neural networks [2] and the intrinsic ability of recurrent spiking neural networks to process temporal patterns [3] both indicate that including a spiking mechanism with all of its complexities is beneficial. The positive impact of introducing developmental processes on the emergence of robustness [4], scalability [5], regeneration, and fault-tolerance [6] in neural networks all implies that incorporating some of these complexities may improve the final performance of the system.

Inclusion of these complexities, on the other hand, requires a huge amount of computational power that can make an evolutionary approach intractable. The challenge may appear insurmountable given that nature has accomplished the equivalent to this through billions of years of evolution, employing huge numbers of processing elements, optimizing each system and process from scratch. However, there is a chance that by adoption of the right combination of natural processes, and imitation of this subset of nature at a sufficient level of detail, desired effects can be produced. Evolvable hardware [7] may enable us to exploit computational resources at a lower level, leading to fine-grained system interactions, low-level parallelism, and a biologically more plausible approach compared to traditional evolutionary computation. The present paper proposes a new bio-plausible developmental process for evolvable hardware as another step towards creating adaptable and bio-inspired spiking neural microcircuits in FPGAs using evolutionary, developmental and learning processes.

II. Background

Evolvable hardware has been previously used for evolving neural microcircuits. For example, Upegui et al. evolved a 3-layer recurrent spiking neural network on a Xilinx Spartan FPGA [8]. However, the number of neurons and synapses, general architecture of the network, and the neuron parameters were fixed during evolution. The seminal work of Thompson [9] with a cellular structure on Xilinx XC6264 not only revived the field of intrinsic evolvable hardware, but also showed the power of evolutionary cellular systems on FPGAs. Different multi-cellular developmental systems for FPGAs have been designed by Haddow and Tufte, Liu, Miller and Tyrrell, Gordon, and many others, cited in [5]. Cellular systems have been also used for development (and simulation) of neural networks. Roggen presented a comprehensive review of the developmental systems in hardware and introduced a new classification of developmental systems in [5], and noted that most of the advantages of developmental systems lie in the cellular online developmental systems implemented in hardware. He also introduced such a cellular development system for evolvable hardware and used it to evolve neural networks for pattern recognition and robot navigation [5]. However, the connectivity patterns of the neurons and the neuron model were limited to six fixed patterns and a simplistic leaky integrate and fire soma model. Recently, Upegui et al. introduced a dynamic routing algorithm to produce nature inspired activity-dependent synaptogenesis in the bespoke cellular reconfigurable chip Ubichip [10]. While this is a very fast routing algorithm of its kind, it has some scaling limitations to be resolved.

Evidence suggests that structural plasticity [11] and wiring delays [12] play major roles in the brain. The placement and wiring of the neurons are also optimized for the high interconnectivity in the brain [13]. In contrast, most of the existing evolvable hardware neural network models (e.g. [5], [8]) are not capable of flexible neurite growth in silicon. Typically, they either are restricted in terms of number of inputs per neuron or impose constraints on the patterns of connectivity and/or placement on the actual chip mostly due to implementation issues. They also do not allow heterogeneous networks with flexible parametric neurons and learning rules as important bio-plausible features. However, a flexible parametric digital neuron model [14] and a cellular structure [15] that allow diverse patterns of neurite growth in FPGAs has been introduced recently. Although this new model provides a flexible platform for construction and fast simulation of spiking neural microcircuits, evolving such networks for a given problem needs suitable developmental and evolutionary processes that can cope with the complexity and the raggedness of the fitness landscape. Designing such a genotype-phenotype mapping is far from straightforward.

Following a nature-inspired approach, here we propose a bio-plausible developmental genotype-phenotype mapping for evolution of spiking neural microcircuits in an FPGA. It is based on the digital neuron model and cortex structure suggested and verified in [14] and [15], multi-cellular protein-protein, gene-protein interactions, and signaling.

III. Hardware platform

The neural microcircuit simulation process runs on an FPGA-based hardware platform. The proposed developmental process (currently implemented in software running on a PC) will be used to reconfigure the FPGA during the development and simulation. The neuron model and the cellular structure used in this system are explained in this section. While the details of the neuron model are not central to this paper, its general architecture and features provide some background for understanding of the developmental process proposed here.

A. Digital neuron model

The digital spiking neuron model used as the basis of this work is explained in detail in [14]. Here, we summarize its general design and advantages. In this model, each digital neuron consists of a set of synapse units and a soma unit connected in a ring architecture. The axonal input of each synapse is connected to the axon of a pre-synaptic neuron. This architecture creates a two-way communication channel in the dendrite and allows the development of different dendrite structures [14]. The dendritic lines and D flip-flops, which connect the units, form a loop (dendritic loop) that conveys data packets. The soma unit sends a packet containing the current membrane potential on its dendritic output. Synapse units process the packets. If a synapse unit receives a pre-synaptic action potential, it adds (or subtracts) its synaptic weight to the first arriving packet. Therefore, the soma unit receives the sum of membrane potential and post-synaptic currents in its dendritic input. After processing this packet, the soma unit sends another packet with the updated membrane potential. Serial arithmetic is used in all the units to create pipelined parallel processing inside each neuron.

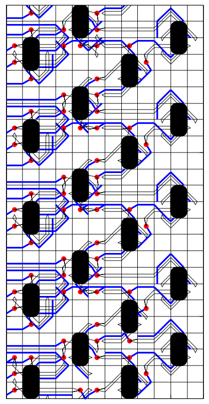
This model is suitable for evolutionary development of heterogeneous spiking neural networks on FPGAs in different ways. First, it uses a parametrically flexible and bio-plausible soma model, which has the potential to be upgraded to more plausible models if hardware budget permits. Secondly, it provides the means for adding a local (thus parallel) learning process such as STDP (Spike-Time-Dependent Plasticity) [16] in each synapse. Moreover, it is relatively fast (up to 4 million updates per second) and occupies acceptable area on the FPGA (less than two and four CLBs for synapse and soma units). More importantly, it allows us to develop adaptable dendrite and axon branches in a cellular cortex structure. The user is free to trim (add) dendrite branch-lets at any point simply by switching few multiplexers. This flexibility is vital for a developmental model that needs on-line growth and modification.

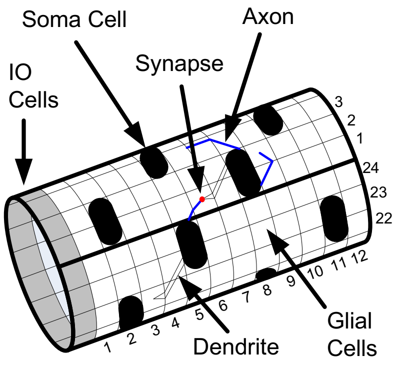
B. The cortex structure

In this work, the cellular platform for development of the neural microcircuits in the FPGA shall be called the cortex. Detailed design and implementation of this cortex is explained in [15]. Here we review its general design and features. The biological cortex is mainly composed of neurons and glial cells [17]. Biological glial cells provide support and nutrition for neurons and act as glue between them. Recently, they were suspected to be also involved in the synapse formation as well as axon and dendrite development [17], [18]. In this study, we use the words glial cells referring to non-neuron cells that provide the means for routing dendrites and axons, and formation of synapses at their intersections.

The cortex consists of a 2D grid of glial cells with neuron soma cells embedded in the middle of them wrapped around like a cylinder so that the top and bottom rows of cells are neighbors. A column (ring) of IO cells is also connected to the left side of the cortex that provides the interfacing with the environment. Soma cells are two times larger than glial cells and fit into two vertically adjacent grid cells. Fig. 1 shows a 12x24 cortex with twenty neurons.

Each glial cell contains a synapse unit. A glial cell receives an axonal and a dendritic input signal from each side and has an axonal and a dendritic output on each side. Soma cells have six of those signals as they are in contact with six neighboring glial cells. Each cell has a number of multiplexers for routing dendrite and axon signals. Unlike axons, dendrites are two-way signals, consisting of two lines forming a closed


(a)



(b)

Fig. 1. (a) A sample 12x24 cortex with 20 neurons. (b) The 2D cylindrical structure of the cortex.

dendritic loop. Each soma cell can sprout up to six axons and six dendrites into its neighboring glial cells directly from the cell body before any division into branch-lets. It is possible to route up to four axons and four dendrites through a glial cell. There are also other multiplexers inside glial cells that can connect the synapse unit to any routed axon and dendrite in the glial cell. The only significant limitation is that there is only one synapse unit available in each glial cell.

C. The Cortex reconfiguration

The cortex structure is designed for Virtex-5 architecture. A Cortex of size 12x120 is implemented in a XC5VLX50T chip. The cortex can be reconfigured to simulate any developing neural microcircuit. The reconfiguration process is comprehensively explained in [15]. However, in current commercial FPGAs (including Virtex-5), relocatable reconfiguration of soma cells is not a straightforward process. In a relocatable design, all the routed internal signals of each cell need to be routed only through routing resources inside that cell area and all the cell input/output signals should use the same signals regardless of cell type (soma, glial or IO cell). For simplicity, the location of soma and IO cells are predefined and fixed during the evolutionary process. Based on those locations, a primary configuration bit-stream will be generated using traditional FPGA design process and tools. Then during the developmental process, the reconfigurable multiplexers (implemented in LUTs) on the edges of the soma and glial cells are reconfigured in every development cycle using the results from developmental process. This way the network simulation can keep running on the chip during development without any interruption. This allows the neurodevelopment process to be provided with the activity from the network simulation for addition of activity dependent synaptogenesis in future.

IV. NEURODEVELOPMENTAL PROCESS

The development process proposed in this paper is similar to the fractal proteins evolutionary gene regulatory network [19] in terms of using the same general bio-plausible genome structure, and basic protein-protein and gene-protein interactions. This is mainly motivated by the evolvability of the fractal proteins in different successful applications [19], [20]. Adoption of these features is based on an analytical study of fractal proteins in [20]. The fractal protein system uses a fractal protein translation and folding mapping into 2D shapes. Using 2D fractal protein shapes in a multi-cellular system slows down the protein folding and development processes significantly. Therefore, a faster and simpler protein folding mapping into 1D protein shapes is introduced in this study. However, the protein folding process is a separate module and can be replaced with any other mapping in order to compare the evolvability and performance of different methods. The neurodevelopmental process proposed here, fully exploits the intercellular signal proteins using novel behavioral protein-protein interactions.

A. Definition of proteins

Proteins are defined here as strings of real numbers in [0,1] of a certain length (L). The values of the real numbers collectively define the shape of a protein. Fig. 2 shows two samples of protein shapes of length $L = 10$. These values are calculated by the protein-folding mapping explained in section IV-C.

B. Genome structure

The genome consists of a single chromosome of variable number of genes. Each gene consist of fifteen fields:

pa, pb, pr, px, ps	T_A	T_C	a, b, r, x, s	C_s	C_d	$Type$

The first five values (pa, pb, pr, px, ps) specify the shape of the promoter using the protein-folding mapping. These values along with T_A (affinity threshold) and T_C (concentration threshold) form the promoter region of a gene. The next five values (a, b, r, x, s) specify the shape of the protein synthesized by this gene using protein folding. These values along with C_s (stability coefficient - specifying the decay rate of the protein), C_d (diffusion coefficient of the protein) and $Type$ (protein type) form the coding region of the gene. All values are real except for $Type$, which is a bit-string that can specify any combination of the protein types. In this system, proteins are of eight different types (written in *italics*). They can be classified into two major groups: transcription factors and structural proteins. Transcription factors, which regulate the expression of the genes, include maternal factors (*soma cell maternal protein, glial cell maternal protein, IO cell maternal protein*) and *regulatory proteins*. Structural proteins, which are virtually part of the cell structure and influence the behavior of the cell, include behavioral proteins (*axon growth protein, dendrite growth protein, synapse formation protein*) and *cell receptor proteins*. The role of each protein type is explained later in this section.

C. Protein folding

Protein folding is the process that translates a set of a, b, r, x, s values (or pa, pb, pr, px, ps values in case of a gene promoter) into a protein (or promoter) shape which is a string of length L of real values. This is performed using the logistic map [21]. The logistic map is a very simple dynamical system of form:

$$x_{k+1} = \mu x_k (1 - x_k) \tag{1}$$

that can create very complex time series with periodic, or chaotic behavior. In this equation x_k is the value of the time series in step k, and

$$\mu = 3 + tanh(5|r|) \tag{2}$$

is the logistic map parameter. The x field in the gene specifies the initial value of x_k in this equation. The r field is also used to calculate μ that controls the behavior of the logistic map dynamical system. This equation will be first iterated for $n = |\lfloor L{\cdot}s \rfloor|$ times. Then the x_k values in subsequent iterations of the equation are scaled and offset by a and b using equation:

$$V_{k-n} = (2a - 1)x_k + 2b - 1 \tag{3}$$

to calculate all the protein shape values V_i for $i = 1..L$. This way, s controls the number of skipped iterations before using the x_k values. The pa, pb, pr, px, ps fields are used instead in case of promoter translation. All the protein and promoter shapes are calculated using this mapping and stored before starting the developmental process in the cells. Fig. 2 shows two sample protein shapes. These shapes can be shifted horizontally by changing the value of s. Protein shapes can be scaled and shifted vertically by changing a and b respectively. The r value in the gene specifies the behavior of the dynamical system, and thus the shape of the protein. The x value in the gene can significantly affect the shape of the protein, particularly when the dynamical system has a chaotic behavior and $|s| \gg 0$. This is because of the sensitivity of a chaotic system to initial conditions. But this sensitivity can be smoothly controlled by evolution using both s and r values.

D. Protein diffusion

For each protein described in the genome, a concentration value in range of $[0, 1]$ is defined at each cortex cell (thus two concentration values for two half cells of a soma cell). Before any protein-protein or gene-protein interaction takes place, the amount of proteins diffused into neighboring cells should be calculated. Here, a simple weighted average of concentration values of the cell and its neighboring cells of form:

$$c_0^{t+1} = C_s\big((1 - C_d)c_0^t + \frac{1}{4}\sum_{i=1}^{4} C_d \cdot c_i^t\big) - 0.002 \tag{4}$$

is used where c_0^t is the concentration value in the centre cell at development step t, and $c_i^t, i = 1, 2, 3, 4$ are the concentration values in neighboring cells. C_s is the stability coefficient of the protein, which is a real number in $[0, 1]$, with 1.0 meaning no decay. C_d is the diffusion coefficient, again a real number in $[0, 1]$, with 0 meaning no diffusion. Zero diffusion coefficients are useful for those internal proteins that cannot cross the cell membrane. The -0.002 offset makes sure that concentration can actually drop to zero instead of converging to zero [19]. Concentrations out of the $[0, 1]$ range are clipped to 0 and 1.

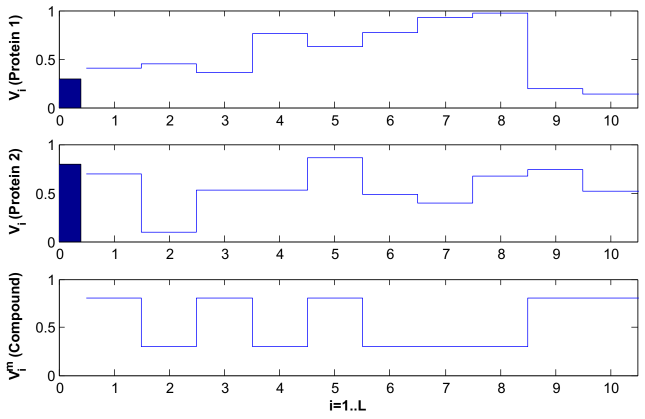


Fig. 2. Example of two protein shapes ($V_i, i = 1..L$) of length $L = 10$. Their concentrations are shown as bars on the left. Merging these proteins results in the protein compound shape (V_i^m) at the bottom.

E. Protein-protein interactions

There are different types of protein-protein interactions depending on the protein types. Proteins can merge together to create a protein compound. The protein compound is also a string of real values of length L. Each value in the protein compound string is equal to the concentration of the protein with the highest value in that position of the string [19]. Fig. 2 shows how two different sample protein shapes of length $L = 10$ are merged to result in a protein compound of the same length. Note that protein compounds do not have separate values for shapes and their concentrations. Protein compound values are actually concentration of shapes. Only proteins with nonzero concentration (existing proteins in a cell) can contribute to the shape of the protein compound. This is to create a very diverse and dynamic set of protein compounds shaped by the protein concentrations in different regions of the cortex. All the proteins in the genome, which are tagged as *cell receptor protein* in the $Type$ field of the genes, are merged in each cortex cell to create a compound cell receptor shape. This string of real values is then used as a mask to filter the shape of those proteins that are tagged as *intercellular signal protein*, meaning that only those shape values with a corresponding nonzero value in the mask are used [19]. The filtered shapes of the intercellular signal proteins are then merged with all the proteins that are tagged as a transcription factor (*regulatory protein* or any type of maternal factors) to create a protein compound.

F. Gene expression (gene-protein interaction)

In each cell, the protein compound interacts with the shape of the promoter in a gene [19], resulting a difference value δ, defined as:

$$\delta = \frac{\sum_{i=1, V_i^p \neq 0}^{L} |V_i^m - V_i^p|}{N} \tag{5}$$

where V_i^m and V_i^p are the ith values in the protein compound string and in the promoter shape string of the gene, and N is number of nonzero values of V_i^p for $i = 1..L$. The probability of the gene expression is then defined as [19]:

$$P(E|\delta, T_A) = \begin{cases} \frac{1+tanh\left(30(2T_A-1+\delta)\right)}{2} & \text{if } T_A < \frac{1}{2} \\ \frac{1+tanh\left(30(2T_A-1-\delta)\right)}{2} & \text{if } T_A \geq \frac{1}{2} \end{cases} \tag{6}$$

where T_A is the affinity threshold of the gene promoter. If a gene is expressed, the concentration of the protein coded in the gene will be increased (or decreased) by [19]:

$$\sigma = c_p \cdot tanh(c_p + T_C) \tag{7}$$

where T_C is the concentration threshold of the gene promoter, and c_p (total concentration seen by promoter of the gene) is calculated using [19]:

$$c_p = \frac{\sum_{i=1, V_i^p \neq 0}^{L} V_i^m}{N}. \tag{8}$$

G. Neurite growth and synapse formation

Behavioral proteins control the growth and guidance of the neurite in this system. Currently each soma sprouts six axonal and six dendritic growth cones at the beginning of the developmental process. However, the probability of generation of a growth cone can also be controlled by separate behavioral proteins (to be added to the protein types). At each development step, the likelihood of growth of growth cone j toward side d of the glial cell (where routing resources are available) is calculated using:

$$\Lambda(G_{jd}) = \frac{\sum_{i=1}^{L} V_i^{mg_j} \cdot V_i^{m\Delta_d}}{L} \tag{9}$$

where $V_i^{mg_j}$ is the ith value in the growth protein compound (merging all growth proteins tagged as *axon growth protein* or *dendrite growth protein*) in the mother cell of growth cone j, and $V_i^{m\Delta_d}$ is the ith value in the gradient compound of all proteins across side d. This gradient compound is calculated in the same way that protein compounds are calculated, except that the gradient (difference) of the protein concentrations across side d of the glial cell is used instead of the local protein concentrations. For each side of a glial cell (processed in a clockwise order), the growth cone with the highest positive likelihood will be routed towards that side. Clearly, the likelihood of growth into soma cells and out of the right edge of the cortex is zero. Moreover, dendrites cannot grow into IO cells. Each IO cell has an axonal growth cone in the neighboring glial cell. Axons of other soma and IO cells can also grow and connect to IO cells. Currently, no neurite branching is allowed and when a growth cone grows into a neighboring cell it moves to that cell and does not duplicate. However, the digital neuron model, cortex cellular structure, and the neural-development algorithm allow the addition of that functionality by adding more behavioral proteins to the system for generation of growth cones, branching, or just by setting a constant threshold for growth likelihood to detect branching.

Currently, the *synapse formation protein* type is not used and whenever an axon and a dendrite of two different cells are present in a glial cell with an available synapse unit, a synapse will form. However, a similar but three-way interaction of synapse formation proteins of two mother cells with the local protein compound in the glial cell can be used to calculate the likelihood of synapse formation between two neurites. Every time that a neurite grows into another cell or a synapse is formed, the configurations of the associated multiplexers are updated to reflect the latest changes.

H. General algorithm

The general neural development algorithm repeats the same procedure for all cells in all development cycles as follows:

```
Initialize the cortex and arrange the soma cells
Calculate and store all protein and promoter shapes
for all development steps do
  for all cortex cells do
    for all proteins do
      Diffuse protein
    end for
    for all genes in the genome do
      Express the gene with prob. $P(E)$ and increase (or
      decrease) the associated concentration
    end for
    if cell type = glial then
      process glial cell
    end if
    if cell type = soma then
      process soma cell
    end if
    Update multiplexer configurations
  end for
  Reconfigure the hardware platform accordingly
end for
```

Processing a glial cell includes synapse formation and neurite growth. Synapse formation involves checking if a free synapse unit, at least one axon and one dendrite exist in the cell and then forming a synapse between two neurites with the highest likelihood of synapse formation. Neurite growth involves calculating the growth likelihood of all growth cones in the cell towards each side and then growing the ones with the highest nonzero likelihood. At the end, the configuration of the corresponding multiplexers involved in the synapse formation and neurite growth are updated accordingly. The neural development algorithm is currently implemented in a synchronous and sequential manner on a CPU. However, with some inter-thread coherence precautions, it is possible to have parallel threads for protein diffusion, gene expression, and neurite growth processes in each cortex cell. Therefore, this

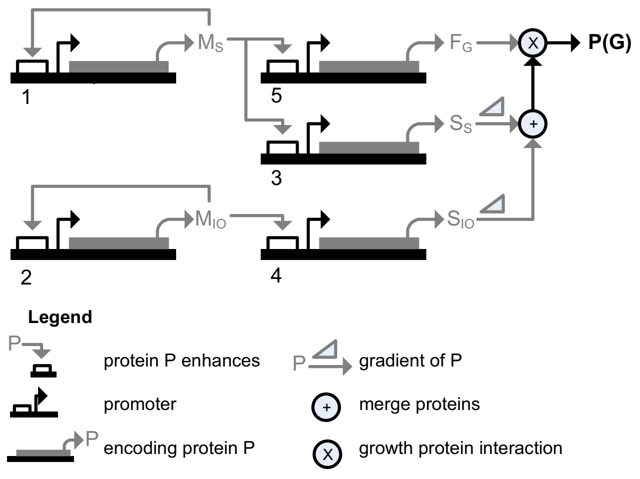


Fig. 3. Gene regulatory network of the designed genome showing the gene-protein and protein-protein interactions.

algorithm lends itself to massively-parallel architectures like FPGAs and GPUs (Graphics Processing Units).

V. Experiments

Before performing any evolvability studies, it is always necessary to check if the new developmental process is able to produce the desired phenotypes at all. A set of experiments was carried out to examine the suitability of this developmental representation for generation of useful neural microcircuits through statistical analysis. The possibility of the emergence of scalability, modularity and fault-tolerance using this system was also checked using designed genomes. The protein size (L) and max development cycles were set to 10 and 200 respectively in these experiments.

A. Network characteristics

The aim of this experiment is to check the possibility of growing useful networks using the new developmental process. Brain networks and animal nervous systems show the properties of small-world networks, that is higher clustering coefficients and shorter characteristic path lengths (average shortest path between any two nodes) compared to random networks [22]. Three set of 1000 networks were developed using 3 different arrangements of 120 neurons in the cortex and randomly generated genomes of length 16. The real values in the genes were set to random numbers in the range [0,1] and the protein types were set to random binary strings. The characteristic path length and clustering coefficient of the resulting networks were recorded for three different neuron arrangements in the cortex.

B. Modularity and scalability

A very simple genome of 5 genes was designed to show how this genotype-phenotype mapping lends itself to emergence of scalability and modularity. Fig. 3 shows a schematic of the gene regulatory network of the designed genome. Soma and IO cells each have a maternal factor of their own ($\mathrm{M_S}$ and $\mathrm{M_{IO}}$), which is sustained at saturation level by positive feedback loops of gene 1 and 2 respectively. These maternal factors have a diffusion coefficient of zero, meaning that they are internal proteins and cannot cross the cell membrane. Each of those maternal factors exactly matches and enhances the promoters of the gene 3 and 4, which leads to synthesis and diffusion of intercellular signal proteins ($\mathrm{S_S}$ and $\mathrm{S_{IO}}$ - with diffusion coefficients of 0.5 and 0.99). The soma cell maternal factor also exactly matches the promoter of the gene 5, which synthesis another internal protein in soma cells ($\mathrm{F_G}$ - with diffusion coefficient of zero) that works both as axon growth factor and dendrite growth factor. The shape of this growth factor can interact with the merged gradient of intercellular signal proteins diffused form both soma and IO cells resulting in growth probably P(G). This genome was used to develop networks using two different cortex sizes (12x12 and 12x24 with 9 and 18 neurons).

C. Fault-tolerance

Fault-tolerant neural microcircuits can be very useful when developed in a huge cortex with large numbers of neurons and glial cells. Fabrication of such a huge cortex in a very large VLSI chip involves low yield factors or high number of faulty cells in the cortex. SEUs (single-event upsets) and unit failures are more frequent in such large systems. A fault-tolerant and robust cortex can resolve these problems. Although evolutionary process will be able to evolve networks that are robust to loss of nodes and links, a bio-inspired developmental system like this can also contribute to fault-tolerance and robustness of the system. This can be achieved either by regeneration or by growth cones avoiding the faulty cells in the first place. Errors and faulty cells might be detected by rather traditional methods such as post-fabrication test, POST (power-on self-test), TMR (triple modular redundancy) or by more innovative and bio-plausible methods such as artificial immune systems [23].

The aim of this experiment is to demonstrate the basic fault-tolerance capability of the developmental system. For this experiment, another simple genome was designed that simply grows an axon from one neuron to another neuron. The IO cell signal protein was used to initiate the differentiation of two neuron types. The network was developed and the routing path of the axon was recorded. In the second step, a glial cell in the axon routing path were tagged “faulty” in order to simulate the effect of a fabrication fault. It was assumed that “faulty” cells are detected either by an error detection mechanism or using a post-fabrication test before starting the developmental process. In this experiment a “faulty” cell simply does not involve in the protein diffusion process (setting all protein concentrations in the cell to zero). Therefore, the likelihood of neurite growth into that cell will be always non-positive. It was anticipated that the axon should deviate from its original path and bypass the “faulty” glial cell.

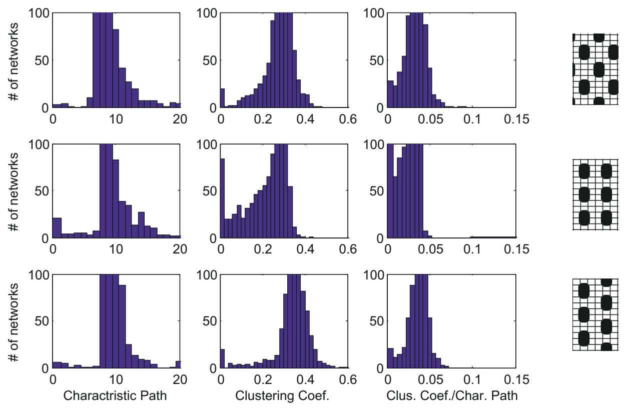


Fig. 4. Distribution of the characteristic path length, clustering coefficient, and the ratio of clustering coefficient to characteristic path length for 1000 developed networks using randomly generated genomes with 3 different neuron arrangement patterns.

VI. Results

A. Network characteristics

Fig. 4 shows the distributions of characteristic path length and clustering coefficient, along with the distribution of their ratio of the developed networks with three different neuron arrangements. All the distribution histograms are cropped at the top to show details, as peak values of the histograms are not of interest here. All three arrangements showed almost the same distribution of characteristic path length with a fat tail on the left side, meaning that developing networks with short characteristic paths is possible using this system. The clustering coefficient was more influenced by neuron arrangement and some networks with clustering coefficients of greater than 0.5 was recorded in case of the third arrangement. Development of networks with both a high clustering coefficient and a short characteristic path length at the same time is captured in the distribution of the ratio of these two, in the third column of Fig. 4. The right tail of the distribution of this ratio shows that generation of such networks with this developmental process is not impossible. The characteristic path length and clustering coefficient of some of the generated networks were similar to statistics of the brain networks reported in [22], namely those of *C.elegans*.

B. Modularity and scalability

Fig. 5 shows the results of the second experiment for two different cortex sizes of 12x24 (Fig. 5(a)) and 12x12 (Fig. 5(b)). Fig. 5(c) shows the diffusion patterns of the five proteins in the cortex at the end of the development process. The same connectivity motif was repeated vertically for all cortex sizes showing the possibility of creating a modular structure using the neurodevelopmental system.

C. Fault-tolerance

Fig. 6(a) shows the single axon grown from one neuron to the other along with the diffusion pattern of six proteins in the cortex after normal development. The glial cell, which is

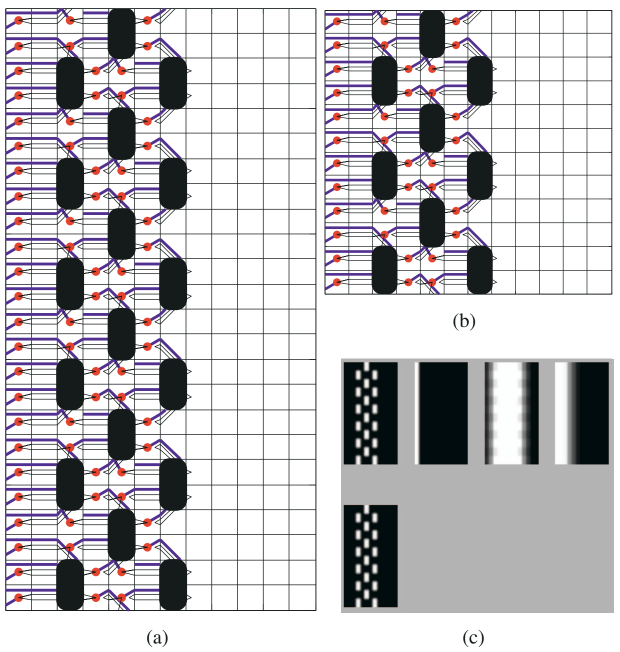


Fig. 5. (a) The developed network using the designed genome in a 12x24 cortex. (b) The developed network using the same genome in a 12x12 cortex. (c) The protein diffusion patterns of the 12x24 cortex.

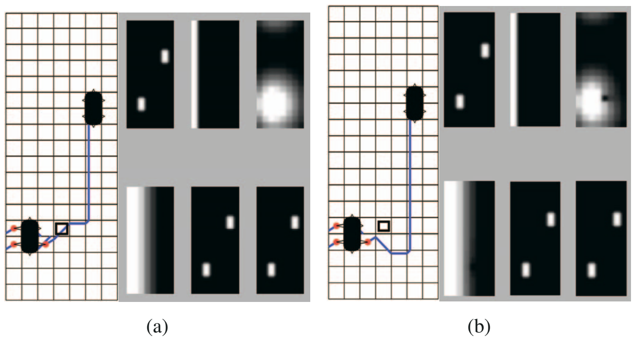


Fig. 6. (a) The single axon routed from one neuron to the other along with the diffusion pattern of six proteins in the cortex. (b) The axon diverted to bypass the faulty glial cell (marked with a gray square) along with the affected protein concentration pattern.

selected to be the "faulty" cell in the second step is labeled with a square in Fig. 6. In the second step (Fig. 6(b)), the glial cell was actually tagged as "faulty" and development process was rerun. Concentration levels of all proteins in the "faulty" cell were zero. The effect of the "faulty" cell in the protein diffusion pattern is notable in the third protein diffusion (Fig. 6(b) top-right). Consequently, the axon avoided the "faulty" glial cell, and connected to the target neuron through another path.

VII. Conclusions

The work described here provides another step towards the evolution of a small adaptable, fault-tolerant, and intelligent brain in silicon. A bio-inspired multi-cellular developmental process for growing spiking neural microcircuits in an FPGA, based on a flexible and bio-plausible digital neuron model and cortex structure was introduced. The developmental process was implemented in software and the network simulation in hardware. However, the proposed developmental algorithm also lends itself to parallel architectures such as FPGAs and GPUs. The statistical analysis of the networks developed from randomly generated genomes showed that even with short chromosomes and proteins, useful small-world networks (with short characteristic paths and high clustering coefficients) could be produced using this genotype-phenotype mapping. It was also demonstrated that basic requirements for the emergence of scalability, modularity and fault-tolerance have been met in this system.

Future works include a fast implementation of the algorithm on a GPU, evolving useful neural microcircuits to tackle specific problems along with an evolvability study, comparison with other methods, and demonstrating adaptability of this system by evolving scalable, modular, robust and fault-tolerant neural microcircuits in an FPGA.

References

[1] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[2] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9. The MIT Press, 1997, p. 211.

[3] Maass, Natschlager, and Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *NEURCOMP: Neural Computation*, vol. 14, pp. 2531–2560, 2002.

[4] A. N. Hampton and C. Adami, "Evolution of robust developmental neural networks," in *Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, 2004.

[5] D. Roggen, D. Federici, and D. Floreano, "Evolutionary morphogenesis for multi-cellular systems," *Genetic Programming and Evolvable Machines*, vol. 8, no. 1, pp. 61–96, 2007.

[6] D. Federici, "A regenerating spiking neural network," *Neural Networks*, vol. 18, no. 5-6, pp. 746–754, 2005.

[7] T. Gordon and P. Bentley, "Evolving hardware," in *Handbook of Nature-Inspired And Innovative Computing*. Springer, 2006, pp. 387–432.

[8] A. Upegui, C. A. Pena-Reyes, and E. Sanchez, "An FPGA platform for on-line topology exploration of spiking neural networks," *Microprocessors and Microsystems*, vol. 29, no. 5, pp. 211–223, Jun. 2005.

[9] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, T. Higuchi, M. Iwata, and L. Weixin, Eds. Berlin: Springer-Verlag, 1997, pp. 390–405.

[10] A. Upegui, A. Perez-Uribe, Y. Thoma, and E. Sanchez, "Neural development on the ubichip by means of dynamic routing mechanisms," *Evolvable Systems: From Biology to Hardware: 8th International Conference Proceedings, ICES 2008, LNCS 5216*, vol. 5216, p. 392401, 2008.

[11] D. B. Chklovskii, B. W. Mel, and K. Svoboda, "Cortical rewiring and information storage," *Nature*, vol. 431, pp. 782–788, 2004.

[12] D. B. Chklovskii, T. Schikorski, and C. F. Stevens, "Wiring optimization in cortical circuits," *Neuron*, vol. 34, pp. 341–347, 2002.

[13] D. B. Chklovskii, "Synaptic connectivity and neuronal morphology: Two sides of the same coin," *Neuron*, vol. 43, pp. 609–617, 2004.

[14] H. Shayani, P. J. Bentley, and A. M. Tyrrell, "Hardware implementation of a bio-plausible neuron model for evolution and growth of spiking neural networks on FPGA," in *AHS '08: Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 236–243.

[15] H. Shayani, P. Bentley, and A. Tyrrell, "A cellular structure for online routing of digital spiking neuron axons and dendrites on FPGAs," in *Proceedings of the 8th international conference on Evolvable Systems: From Biology to Hardware*. Springer, 2008, pp. 273–284.

[16] S. Song, K. Miller, and L. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[17] P. Laming and E. Syková, *Glial Cells: Their Role in Behaviour*. Cambridge University Press, 1998.

[18] F. W. Pfrieger, "Role of glia in synapse development," *Current Opinion in Neurobiology*, vol. 12, pp. 486–490, 2002.

[19] P. J. Bentley, "Controlling robots with fractal gene regulatory networks," in *Recent Developments in Biologically Inspired Computing*, L. de Castro and F. von Zuben, Eds. Idea Group Inc, 2005, ch. 13, p. 320.

[20] J. Krohn, P. J. Bentley, and H. Shayani, "The challenge of irrationality: Fractal protein recipes for PI," 2009, proceedings of the Genetic and Evolutionary Conmputation Conference (GECCO 2009), in press.

[21] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, pp. 459–467, 1976.

[22] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[23] J. Timmis, M. Amos, W. Banzhaf, and A. Tyrrell, ""Going back to our roots": second generation biocomputing," *Journal of Unconventional Computing*, vol. 2, pp. 349 – 378, 2006.