The POEtic Electronic Tissue and its Role in the Emulation of Large-Scale Biologically Inspired Spiking Neural Networks Models

Short title: POEtic tissue

J. Manuel Moreno[1], Yann Thoma[2], Eduardo Sanchez[2], Jan Eriksson[3], Javier Iglesias[3,4], Alessandro Villa[4]

[1]Technical University of Catalunya, Dept. of Electronic Engineering, Barcelona, Spain, moreno@eel.upc.edu
[2]Swiss Federal Institute of Technology Lausanne, Logic Systems Laboratory, Lausanne, Switzerland, {Yann.Thoma, Eduardo.Sanchez}@epfl.ch
[3]Laboratory of Neuroheuristics, Information Systems Department INFORGE, University of Lausanne, Lausanne, Switzerland, jan@lnh.unil.ch, Javier.Iglesias@unil.ch
[4]Laboratory of Neurobiophysics, INSERM U318, University Joseph-Fourier, Grenoble, France, Alessandro.Villa@ufj-grenoble.fr

Corresponding author:
Juan Manuel Moreno Arostegui
Technical University of Catalunya
Department of Electronic Engineering
Campus Nord, Building C4
c/Jordi Girona 1-3
08034 – Barcelona – Spain
E-mail: moreno@eel.upc.edu
Phone: +34 93 401 56 91
Fax: +34 93 401 67 56

Keywords: Artificial tissue, phylogenesis, ontogenesis, epigenesis, learning, spiking neural networks models, STDP rule, programmable hardware, POEtic.

Abstract:
One of the major obstacles found when trying to construct artefacts derived from principles observed in living beings is the lack of actual dynamic hardware with autonomous capabilities. Even if programmable devices offer the possibility of modifying the functionality implemented in the device, they rely on external hardware and software elements to provide its physical configuration. In this paper we shall present a new family of electronic devices, called POEtic, whose architecture has been derived from the basic properties that can be extracted from the three major organisation principles present in living beings: phylogenesis, ontogenesis and epigenesis. We shall demonstrate that the capabilities present in these new programmable devices make them an ideal candidate for the real-time emulation of large-scale biologically inspired spiking neural networks models.

1. Introduction

Even if there is a huge variability in the external features and functions associated with the living beings we can observe on the earth, their organisation is driven by principles that can be grouped around three main axes:

Phylogenesis: Also called evolution, it includes all the mechanisms that, driven by the pressure posed by nature, permit to determine the genetic information for a population of individuals that best fits to a given environment.
Ontogenesis: Ontogenetic mechanisms permit the development of a single individual driven by the information contained in its genome. Apart from developmental capabilities, self-replication and self-repair (what for most living beings means healing abilities) constitute clear examples of ontogenetic processes.
Epigenesis: It includes all the mechanisms that permit a single individual to efficiently interact with its direct environment. Epigenetic mechanisms include those plasticity-oriented processes that, driven by a sensor-actuator loop, permit an organism to modify its internal structure or its behaviour in order to adapt to the specific conditions present in a given environment at any time. Examples of biological subsystems showing epigenetic principles can be found in the central nervous system of mammals and in the immune system.

Taking inspiration from these organisation principles, the main goal of the POEtic project was the development of a flexible hardware substrate showing the basic features that permit living beings to show evolutionary, developmental or learning capabilities. The hardware substrate, in the form of a new electronic device, should permit the construction of electronic tissues able to solve tasks where these bio-inspired features represent a clear advantage over classical techniques.

The paper is organised as follows: In the next section we shall present the overall organisation of the POEtic tissue, describing the details of its main constituent parts. Then we shall introduce the features of a new learning model for spiking neural networks models that, when used in large-scale networks, shows interesting feature extraction capabilities. Once physically implemented into the POEtic devices, it will be demonstrated that these provide an efficient prototyping instrument for neuroscience research. The paper will finish presenting the conclusions and our current work.

2. Overall organisation of the POEtic tissue

The POEtic tissue is organised as a homogeneous bi-dimensional array of POEtic chips, each

one of them being able to implement a given number of cells as required by the application to be handled. The organisation of a single POEtic chip is presented in figure 1.

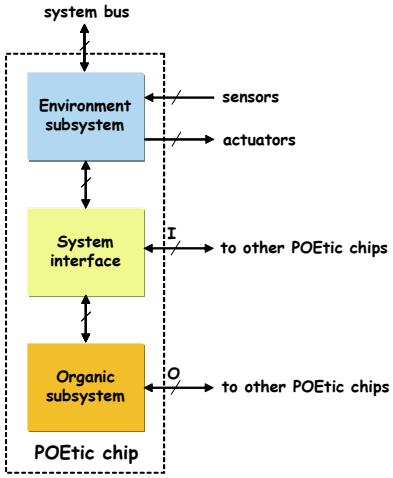

Figure 1. Organisation of a POEtic chip

From a structural point of view the organisation of a POEtic chip is divided in three main sections: the environment subsystem, the organic subsystem and the system interface. The environment subsystem is in charge of managing the interactions with the environment, and also of implementing the phylogenetic mechanisms of the tissue. The organic subsystem manages the physical realisation of the epigenetic and ontogenetic processes to be exhibited by the tissue. Finally, the system interface takes care of the efficient communication between these two subsystems. It also provides the mechanisms that permit the tissue to exhibit scalable properties.

The overall organisation of the resulting tissue is depicted in figure 2.

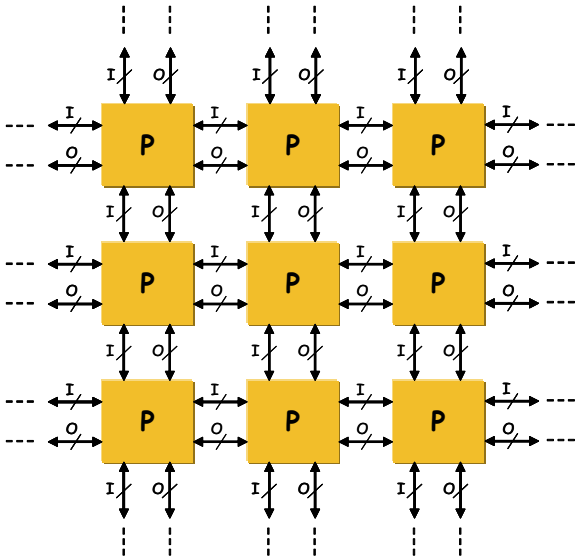

Figure 2. Overall organisation of the POEtic tissue

The squares in figure 2 represent POEtic chips, so that the sample tissue represented in the figure is constituted by 9 POEtic chips (the squares labelled as P) organised as a 3x3 matrix. As it can be deduced from the figure, the local communication between chips is separated in two different sections. The bidirectional lines labelled as I represent those connections associated with the system interface, while the bidirectional lines labelled as O indicate the connections established between the organic subsystems included in every chip. As it will be explained later the connections corresponding to the system interface provide the scalability features required

by the POEtic tissue, meaning that it can be constituted by as many chips as required by the actual application to be tackled. The connectivity between the organic subsystems is established at the routing plane level, and they allow for an effective communication mechanism between cells that are physically implemented in different chips.

Even if the POEtic tissue may be constructed from an arbitrary number of POEtic chips, each of them with their own functional subsystems, the system interface and the choice of the system bus makes it possible to handle the final tissue as a single POEtic chip. The only difference between a single chip and a tissue is the actual size of the organic subsystem, which in the later case is an aggregation of all the organic subsystems present in the tissue.

2.1. The environment subsystem

Figure 3 shows the internal organisation of the environment subsystem.

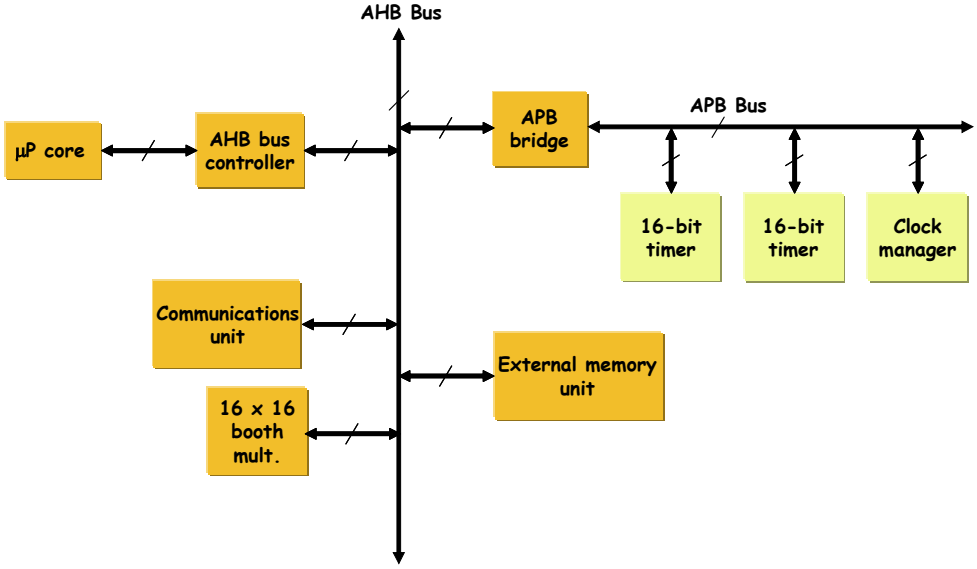

Figure 3. Internal organisation of the environment subsystem

As it can be deduced from figure 3 the architecture of the environment subsystem is structured around a specific microprocessor core. It is a 32-bit custom RISC processor, with dedicated instructions for developing evolutionary algorithms. A pseudo-random number generator is included in the ALU of the processor. The organisation of the environment subsystem is organised around the AHB (Advanced High-performance Bus) bus corresponding to the AMBA specification [1]. Simple peripherals are placed in a separate bus section, called APB (Advanced Peripheral Bus) that interfaces with the AHB bus through a bridge.

All the subsystems included in the POEtic tissue can be managed by the environment subsystem through a careful design of its memory map, whose structure is presented in table 1. The numbers provided in table 1 are specified in hexadecimal format. Even if the organic subsystem of the POEtic tissue is mapped in only one memory section, in fact this section maps the organic subsystems of all the chips that are present in the tissue for a given application.

The first 25 words of the program data section are reserved for the interrupt vectors of the microprocessor. Table 2 summarises the organisation of this interrupt vector table. The content of each of these memory positions is a JUMP instruction that points to the start address of the corresponding interrupt service routine.

The priority of the interrupt sources is directly related to the value of its associated interrupt vector, being thus the internal interrupt 0 the interrupt source with the highest priority.

Section	Start address	End address
Program	0x0000_0000	0x3FFF_FFFF
Data	0x4000_0000	0x7FFF_FFFF
Multiplier	0xC000_0000	0xC000_0003
Communications unit	0xD000_0000	0xD000_0150
Timers	0xE000_0000	0xE000_0007
Clock manager	0xE000_0008	0xE000_000F
Organic subsystem	0xF000_0000	0xFFFF_FFFF

Table 1. Memory map organisation of the POEtic tissue

Interrupt source	Interrupt vector
Main program	0x0000_0000
Timer 0	0x0000_0001
Timer 1	0x0000_0002
Multiplier	0x0000_0003
Clock manager	0x0000_0004
UART 0 TX	0x0000_0008
UART 0 RX	0x0000_0009
UART 1 TX	0x0000_000A
UART 1 RX	0x0000_000B
I2C	0x0000_000B
SPI	0x0000_000C
Parallel port	0x0000_000D
External interrupt 1	0x0000_0010
External interrupt 0	0x0000_0018

Table 2. Organisation of the interrupt vector table of the microprocessor

The communications unit included in the environment subsystem permits to implement an 8-bit bi-directional port, two UARTs, one SPI interface and one I2C interface. The functionality of these interfaces can be programmed by the user to match the requirements of a given application.

The clock manager unit has been added to the environment subsystem of the POEtic tissue in order to facilitate the hardware debugging procedures for the functionality implemented in the organic subsystem. This unit permits to generate a clock signal for the organic subsystem whose frequency is divided with respect to that associated to the system clock. Furthermore, if desired, this unit permits also to stop the clock signal provided to the organic subsystem after a specified number of clock cycles (from 1 to 65535). This feature allows for advancing the state of the organic subsystem edge-by-edge and then observing it (note that the environment subsystem has access through the system interface to the configuration and state of the organic subsystem).

From an architectural point of view the organisation of the external memory unit of the environment subsystem is divided in three main parts: the boot ROM, the program ROM and the data RAM.

The presence of a boot ROM section permits the user to load upon a power up sequence a program that may be transferred to the microprocessor using any one of the peripherals included in the communications unit. This means that the physical architecture of the memory unit of the microprocessor has two possible configurations, as depicted in figure 4.

The organisation depicted in figure 4(a) corresponds to a situation where the program to be executed by the microprocessor is fixed and already stored in a ROM. In this case after the power up sequence the microprocessor starts executing directly from this memory section. Figure 4(b) shows an organisation corresponding to a case where there is just a boot loader program stored in a boot ROM that takes care of capturing through one of the peripherals included in the communications unit the actual program to be executed by the microprocessor. This program is stored in the program ROM section that is physically implemented by means of a Flash or a SRAM unit. In order to permit the microprocessor to physically write the program Rom section during this boot sequence the memory map is slightly changed, so that the program ROM section is mapped in the memory area starting at address 0x6000_0000.

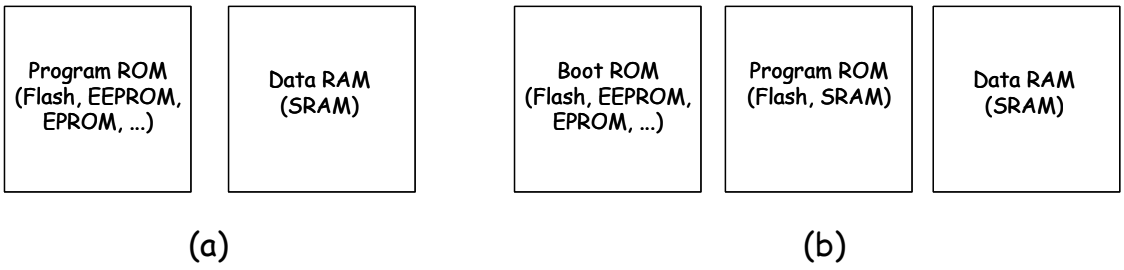

Figure 4. Physical architecture of the external memory unit of the environment subsystem

2.2. The organic subsystem

The organic subsystem is made up of 2 layers, as depicted in figure 5: a two-dimensional array of basic elements, called molecules, and a two-dimensional array of routing units. Each molecule is connected to its four neighbours in a regular structure. Mainly containing a 16-bit look-up table (LUT) and a flip-flop (DFF), it has the capability of accessing the routing layer that is used for inter-cellular communication. This second layer implements a dynamic routing algorithm allowing the creation of data paths between cells at runtime.

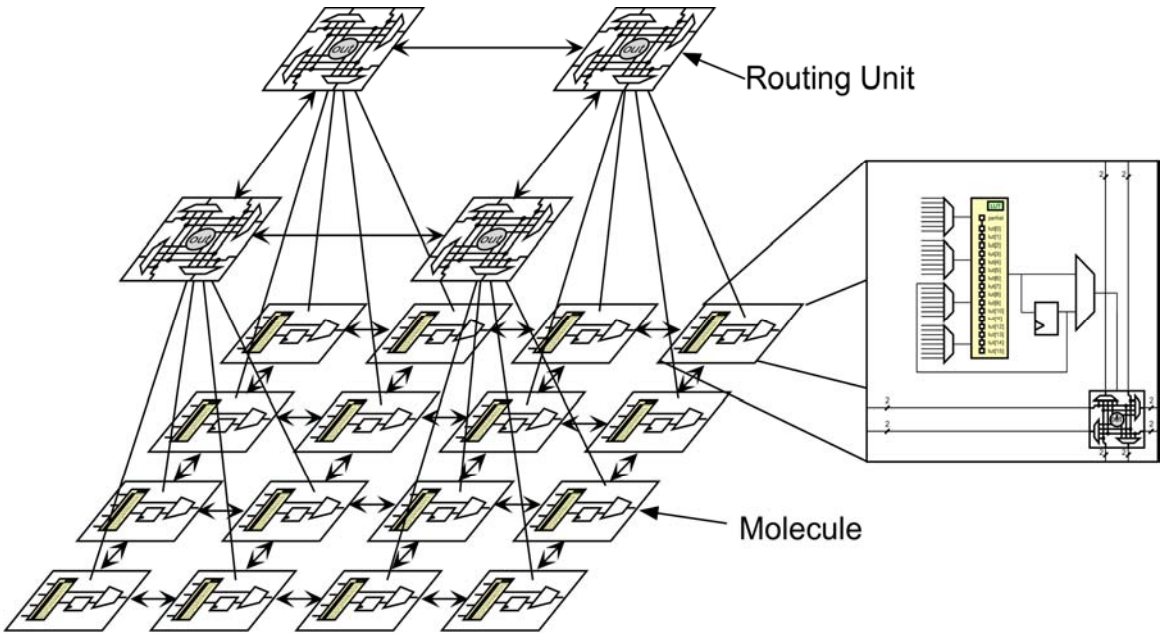

Figure 5. Organisation of the organic subsystem

A molecule is the smallest programmable element of the POEtic tissue. It is mainly composed

of a flip-flop (DFF), and a 16-bit look-up table (LUT) (figure 5). Eight modes of operation are supplied to ease the development of applications that need cellular systems and/or growth and self-repair. The LUT is composed of a 16-bit shift register that can be split in two, used as a shift register, or as a normal look-up table.

A molecule has eight different operational modes, to speed up some operations, and to use the routing plane. The functional modes provided for the molecules are the following:

- In **4-LUT** mode, the 16-bit LUT supplies an output, depending on its four inputs.
- In **3-LUT** mode, the LUT is split into two 8-bit LUTs, both supplying a result depending on three inputs. The first result can go through the flip-flop, and is the first output. The second one can be used as a second output, and is directly sent to the south neighbor (can serve as a carry in parallel operations).
- In **Comm** mode, the LUT is split into one 8-bit LUT, and one 8-bit shift register. This mode could be used to compare a serial input data with a data stored in the 8-bit shift register.
- In **Shift Memory** mode, the 16 bits are used as a shift register, in order to store data, for example a genome. One input controls the shift, and another one is the input of the shift memory.
- In **Input** mode, the molecule is a cellular input, connected to the inter-cellular routing plane. One input is used to enable the communication. When inactive, the molecule can accept a new connection, but won't initiate a connection. When active, a routing process will be launched at least until this input connects to its source. A second input selects the routing mode of the entire POEtic tissue.
- In **Output** mode, the molecule is a cellular output, connected to the inter-cellular routing plane. One input is used to enable the communication. As in Input mode, when inactive the molecule can accept a new connection, but won't initiate a connection. When active, a routing process will be launched at least until this output connects to one target. Another input supplies the value sent to the routing plane, as so to another cell.
- In **Trigger** mode, the 16-bit shift register should contain "000...01" for a 16-bit address system. It is used by the routing plane to synchronize the address decoding during the routing process. One input is a circuit enable, that can disable every DFF in the tissue, and another one can reset the routing plane, and so start a new routing.
- In **Configure** mode, the molecule can partially configure its neighborhood. One input is the configuration control signal, and another one is the configuration shifting to the neighbors.

Long distance inter-molecular communication is possible by the way of switch boxes. Each switch box consists of eight input lines (two from each cardinal direction) and eight corresponding output lines, and is implemented with eight inputs multiplexers. Two outputs are sent into each of the four neighbors of the molecule, as shown in figure 6.

Each output line can be connected to one of the six input lines from the other cardinal directions (no u-turns allowed) or to one of two possible outputs of the molecules (the output or the inverted output).

A molecule is defined by 75 configuration bits. They are configured by loading them in parallel, from the micro-controller. A partial reconfiguration is also possible, a molecule being able to shift configuration bits of its neighbourhood. Actually, when shifting, 76 bits are used, as the value of the flip-flop has to be in the configuration chain, in order to be able to retrieve its value.

The configuration system of the molecules can be seen as a shift register of 76 bits split into 5 blocks: the LUT, the selection of the LUT's input, the switch box, the mode of operation, and an extra block for all other configuration bits. Each block contains, as shown in figure 7, together

with its configuration, one bit indicating, in case of a reconfiguration coming from a neighbour, if the block has to be bypassed. This bit can only be loaded from the micro-processor, and remains stable during the entire lifetime of the organism.

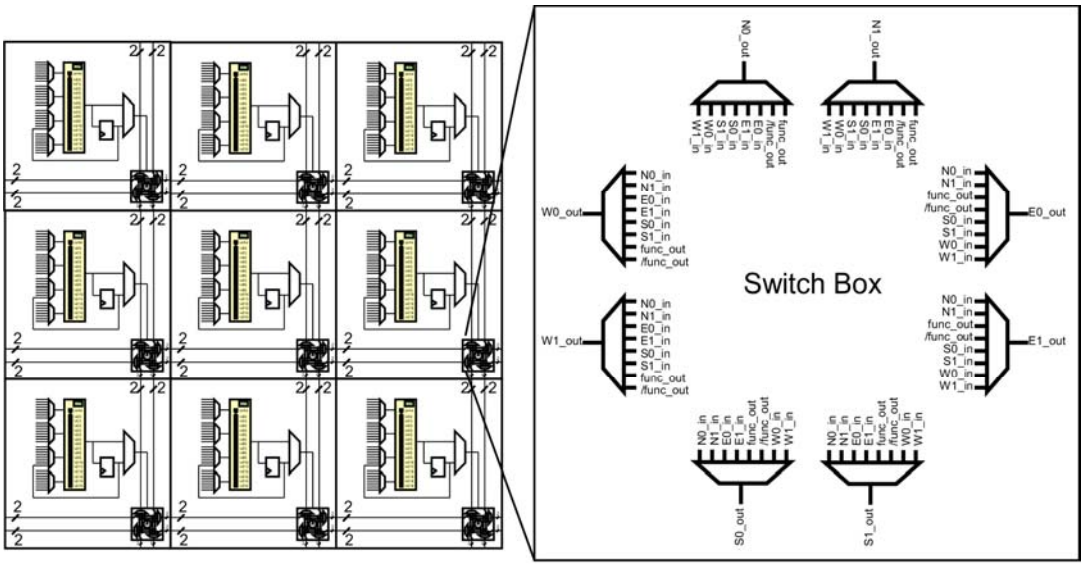

Figure 6. Nine molecules, connected through their switchboxes, and detailed view of a switchbox

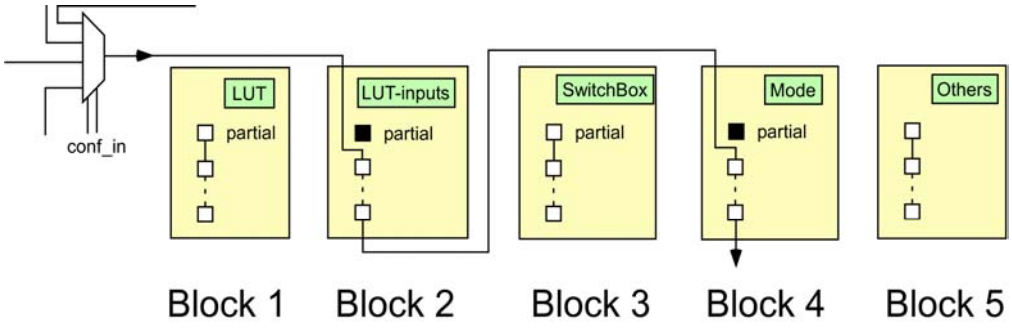

Figure 7. Organisation of the configuration bits for partial reconfiguration

The special configure mode allows a molecule to partially reconfigure its neighbourhood. It sends bits coming from another molecule to the configuration of one of its neighbours. By chaining the configurations of neighbouring molecules, it is possible to modify multiple molecules at the same time, allowing, for example, the synaptic weights in a neuron to be changed.

Three configuration bits are used to define the possible origin of a partial reconfiguration: two bits for selecting the origin, and one bit that enables the partial configuration. In case of a neighbor tries to partially reconfigure the molecule, if this config_partial_enable bit is set to ’1’, then the molecule is partially reconfigured, and it tries to partially reconfigure its neighbors, by chaining the output of the configuration stream. If the config_partial_enable bit is set to ’0’, then no partial reconfiguration is executed, and no signal is sent to the neighbors.

This partial reconfiguration allows for instance to use the configuration bits of a molecule to store information. A maximum of 54 bits can be stored in only one molecule, allowing for efficiently implementing genome storage. By modifying the LUT content, a cell can also modify its behaviour, that is a useful feature for evolvable hardware.

The second plane of the organism subsystem implements a dynamic routing algorithm to allow the circuit to create paths between different parts of the molecular array. The possibility of having a pseudo-static routing has also been added, to ease the development of applications that only need local connections between cells.

The dynamic routing system is designed to automatically connect the cells' inputs and outputs. Each output of a cell has a unique identifier, at the organism level. For each of its inputs, the cell stores the identifier of the source from which it needs information. A non-connected input (target) or output (source) can initiate the creation of a path by broadcasting its identifier, in case of an output, or the identifier of its source, in case of an input. The path is then created using a parallel implementation of the breadth-first search algorithm. When all paths have been created, the organism can start operation, and execute its task, until a new routing is launched, for example after a cell addition or a cellular self-repair.

Our approach has many advantages, compared to a static routing process. First of all, a software implementation of a shortest path algorithm, such as Dijkstra's [2], is very time-consuming for a processor, while our parallel implementation requires a very small number of clock cycles to finalize a path. Secondly, when a new cell is created it can start a routing process, without the need of recalculating all paths already created. Thirdly, a cell has the possibility of restarting the routing process of the entire organism, if needed (for instance after a self-repair). Finally, our approach is totally distributed, without any global control over the routing process, so that the algorithm can work without the need of the central micro-processor.

Every routing unit is composed of a switch box and a finite state machine. The switch box contains five multiplexers that can select the value sent to each of the four neighbors, and to the molecules underneath. The state machine is responsible for correctly configuring the multiplexers, and implements the distributed routing algorithm, by communicating with the other routing units.

The routing algorithm is executed in four phases:

Phase 1: Finding a Master
In this phase, every target or source that wants to and is not connected to its correspondent partner tries to become master of the routing process. A simple priority mechanism chooses the most bottom-left routing unit to be the master, as shown in figure 8. Note that there is no global control for this priority, every routing unit knowing whether or not it is the master. This phase is over in one clock cycle, as the propagation of signals is combinational.

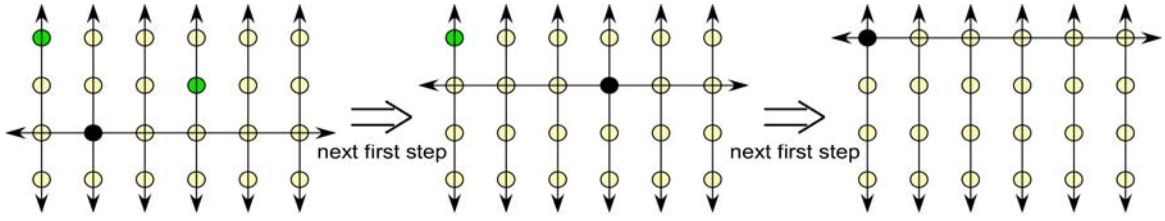

Figure 8. Three consecutive steps of the routing algorithm. The black routing unit will be the master, and therefore will perform its routing

Phase 2: Broadcasting the Address
Once a master has been selected, it sends its address in case of a source, or the address of the needed source in case of a target. It is sent serially, in `n` clock cycles, where `n` is the size of the address. The same path as in the first phase is used to broadcast the address, as shown in figure 9.

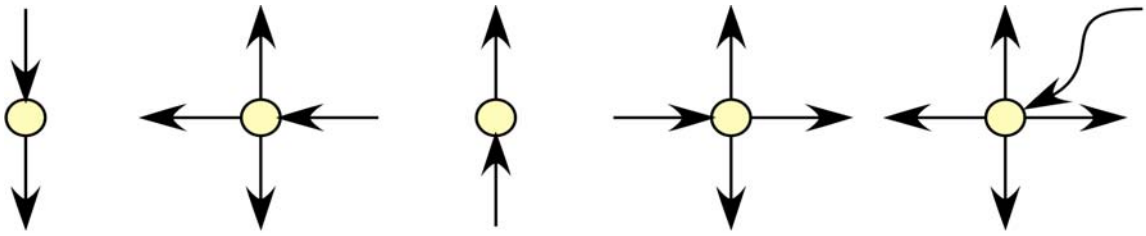

Figure 9. The propagation direction of the address: north → south | east → south, west, and north | south → north | west → north, east, and south | routing unit → north, east, south, and west

Every routing unit, except the one that sends the address, compares the incoming value with its own address (stored in the molecule underneath). At the end of this phase, that is, after `n` clock cycles, each routing unit knows if it is involved in this path. In practice, there has to be one and only one source, and at least one target.

Phase 3: Eliminating sources and targets

In some situations, a source should start a routing process, for instance, in a developmental process. In such a process, it would be useful to have many sources and targets with the same ID. So at this stage, it is possible there is more than one source involved in the routing process. In order to avoid multiple sources, in this phase that lasts only one clock cycle, if a source is at the origin of the routing process, it sends a signal to every other routing unit, to let them know a source is at the origin. Then every other source with the same ID disabled its participation in the current process, and during the next phase, the source will connect to the nearest target.

The same disable is performed in case a target launched the routing process. Every target that is not the master disables its participation to the current process, to ensure that the target that started the process will be the only one connected to a source. In this case, the nearest source will be connected to this target.

Phase 4: Building the Shortest Path

The last phase, largely inspired by [3], creates a shortest path between the selected source and the selected targets. An example involving 8 sources and 8 targets is shown in figure 10, for a densely connected network.

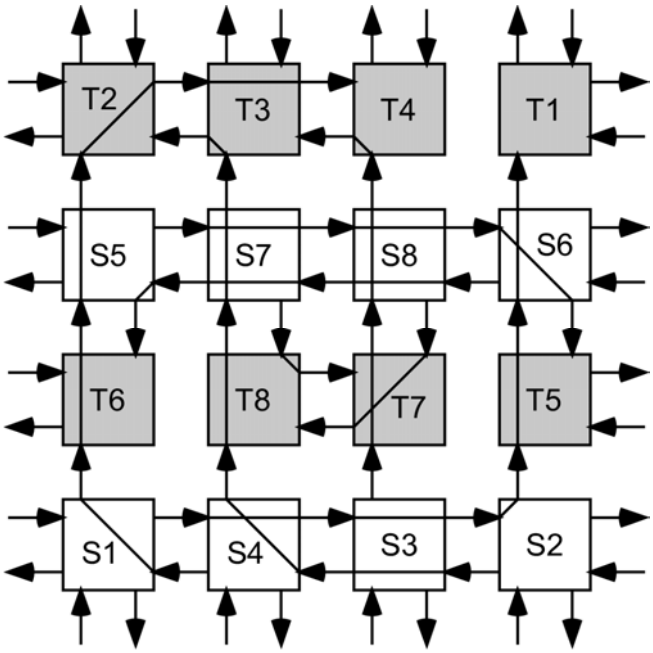

Figure 10. Test case with a densely connected network

A parallel implementation of the breadth-first search algorithm allows the routing units to find the shortest path between a source and many targets. Starting from the source, an expansion process tries to find targets. When one is reached, the path is fixed, and all the routing resources used for the path will not be available for the next successive iterations of the algorithm.

Figure 11 shows the development of the algorithm, building a path between a source placed in column 1, row 2 and a target cell placed in column 3, row 3. After 3 clock cycles of expansion, the target is reached, and the path is fixed, prohibiting the use of the same path for a successive routing.

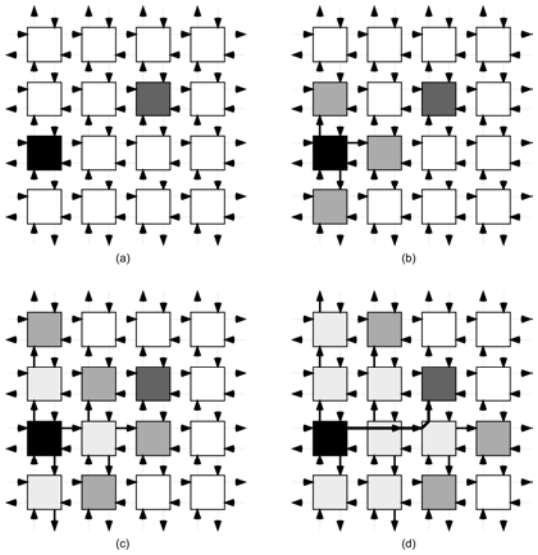

Figure 11. Step (a) one, (b) two, (c) three and (d) four of the path construction process between the source placed in column 1, row 2 and target cell placed in column 3, row 3

Based on addresses, the dynamic routing presented above is very flexible. However, for some applications, this flexibility can become a disadvantage, for example if we only need local communications between cells like a 4-neighborhood.

A second mode of routing has been added for this purpose. A flip-flop in the tissue can be configured by the molecules to choose the mode to use for a specific application. The pseudo-static mode uses the fact that every switch boxes are pass-through after a hardware reset. When in pseudo-static mode, the routing units that are connected to input or output molecules only shift the content of the molecule LUT into the configuration of the switchbox. By this way, in 16 clock cycles, the inter-cellular routing is completed, and the circuit can start its task. The only limitation is that a path between two cells can only be a vertical or a horizontal one, without more complex possibilities (figure 12).

2.3. The system interface

As it has been mentioned previously, the system interface of the Poetic tissue plays a major role in allowing for its scalability features. This means that the physical size of the tissue can be accommodated to the actual needs of a given application without posing specific constraints neither on the system architecture nor in the connectivity pattern among the POEtic chips that constitute the tissue.

The POEtic tissue, as it was presented in figure 1, can be constructed as a bidimensional array constituted by POEtic chips. The connectivity between these chips, as depicted in this figure, is based on two different buses, named organic (O) and interface (I) buses. The signals that constitute the organic bus allow the organic subsystems present in every POEtic chip to

communicate (at a cellular level).

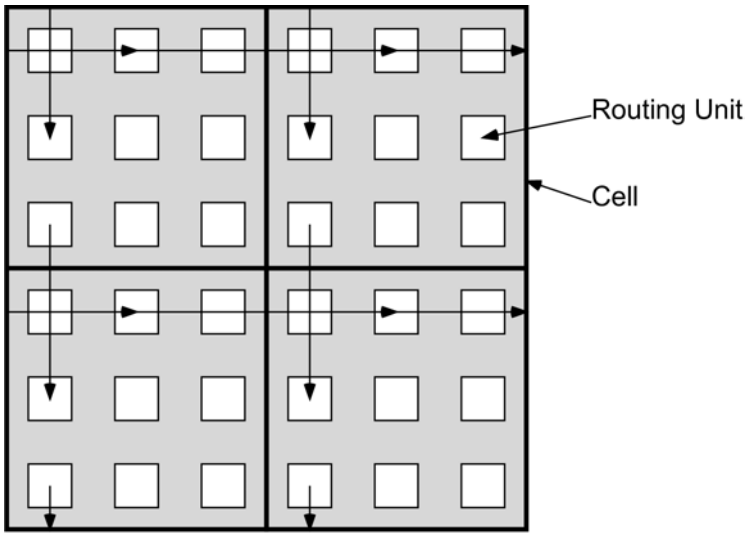

Figure 12. A pseudo-static communication scheme between four cells

The interface bus carries those signals that permit to handle the collection of POEtic chips as a single tissue, so that from a user point of view the tissue has only one environment subsystem and one organic subsystem. This is represented in figure 13.

Regarding the scalability of the environment subsystem, even if every POEtic chip contains a single environment subsystem, only one of them will be active in the tissue. This is accomplished by a specific signal present in every POEtic chip, called master, that indicates (when set to a value '0') that the environment subsystem of a specific chip will be managing the complete tissue.

The 68 signals (32 data lines, 32 address lines, sahbi_hsel, sahbi_hready, sahbi_hwrite and sahbo_hready) that constitute the AHB bus used for the POEtic tissue are connected to all the POEtic chips. This means that the chip identified as a master of the system can access the resources present in any other chip. A specific chip is identified within the array using Cartesian coordinates that correspond to the physical position of the chip in the array. This means that a chip with coordinates (X,Y) is placed in column X and row Y within the array.

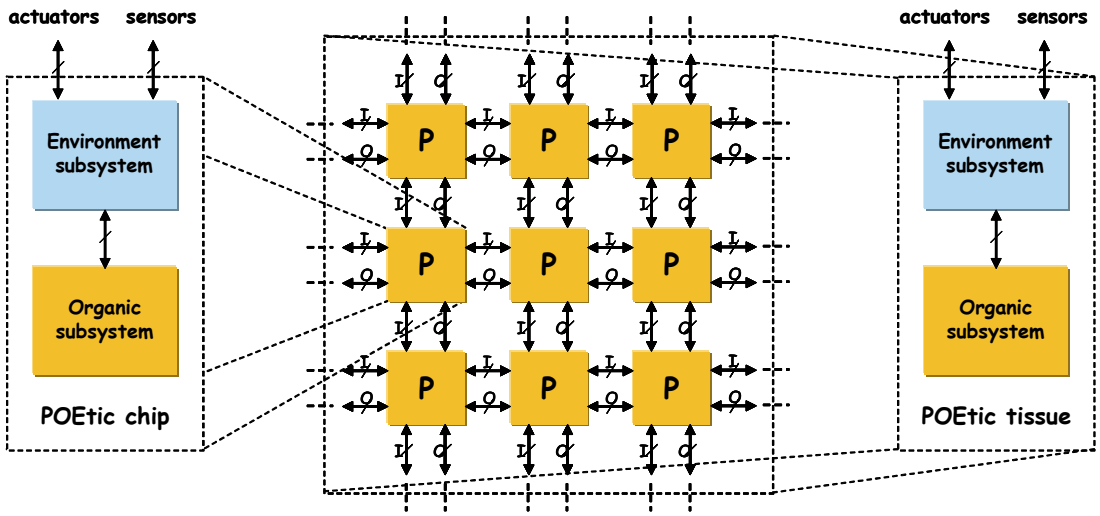

Figure 13. Scalability properties of the POEtic tissue

The coordinates of a given chip are not pre-programmed, but are calculated for a given array configuration during a coordinate propagation phase that should be performed before the tissue is operational. For this purpose every POEtic chip has two inputs, named Xin and Yin, and two outputs, Xout and Yout. The Xin input of a given chip is connected to the Xout output of the chip placed in the same row and in the previous column within the array. The Yin input of a given chip is connected to the Yout output of the chip placed in the same column and in the previous row within the array.

Every POEtic chip receives in serial mode its X coordinate through its Xin input and its Y coordinate through its Yin input. The coordinates are received in serial mode, so that by default the Xin and Yin inputs are in idle state (i.e., with a value '0'), and after one of these input is set to value '1' the POEtic chip should recognise that during the next 4 (in the current version of the POEtic chip the X and Y coordinates are 4-bit wide, but this can be easily extended to any desired size) cycles its X or Y coordinate will be received through the corresponding input. Once a given chip has received its X and Y coordinates it calculates and sends the coordinates for its direct neighbours. The coordinate propagation process is started by the chip whose environment subsystem has been identified as a master. The coordinate propagation process is started when the microprocessor included in the environment subsystem of the master chip performs a write cycle on the address 0xF000_0004 (as it was indicated in table 1, the organic subsystem is mapped in the memory space ranging from 0xF000_0000 to 0xFFFF_FFFF).

Once all the chips have got their actual coordinates within the Poetic tissue it is quite simple for the environment subsystem to access to the organic subsystem present in any chip. In order to access (either in read or write mode) the configuration of a specific molecule present in a POEtic chip placed at coordinates (X,Y) the environment subsystem should perform a read or write access to the memory position 0xF00X_YABC, where:

- **X:** Row where the POEtic chip is placed
- **Y:** Column where the POEtic chip is placed
- **A(3:0)B(3:0)C(3:2):** These 10 bits indicate the address of the molecule within the chip. One POEtic chip contains 144 molecules, and their mapping ranges from 0x002 to 0x091.
- **C(1:0):** These 2 bits indicate which one of the 3 configuration words of the molecule are to be read or written. A value "01" implies the activation of the cs1 signal, a value "10" implies the activation of the cs2 signal, while a value "11" implies the activation of the cs3 signal.

Bearing this in mind, the final organisation of the system interface included in every POEtic chip is that depicted in figure 14.

The wen signal depicted in this figure indicates if the access to the configuration of a given molecule is in read or write mode. The bidirectional configuration data bus is in fact constituted by two independent 32-bit buses, one for read access and the other for write access.

2.4. Physical implementation

The POEtic chip has been implemented and fabricated as an ASIC of 54 mm^2 using a 0.35 μm CMOS process. The chip, whose layout is depicted in figure 15, contains 144 molecules organised as an 8x18 array and the complete environment subsystem explained in previous sections.

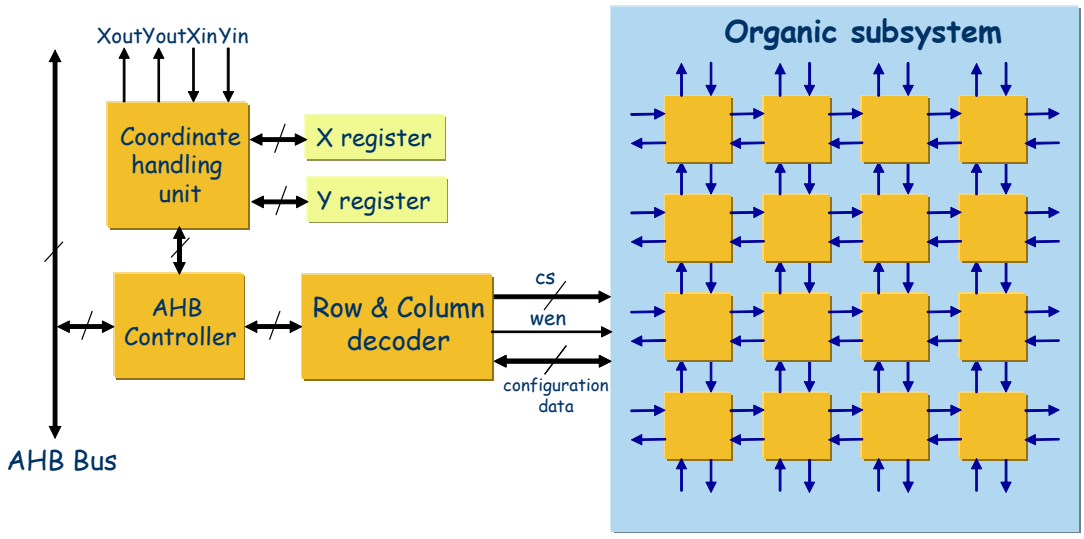

Figure 14. Internal organisation of the system interface

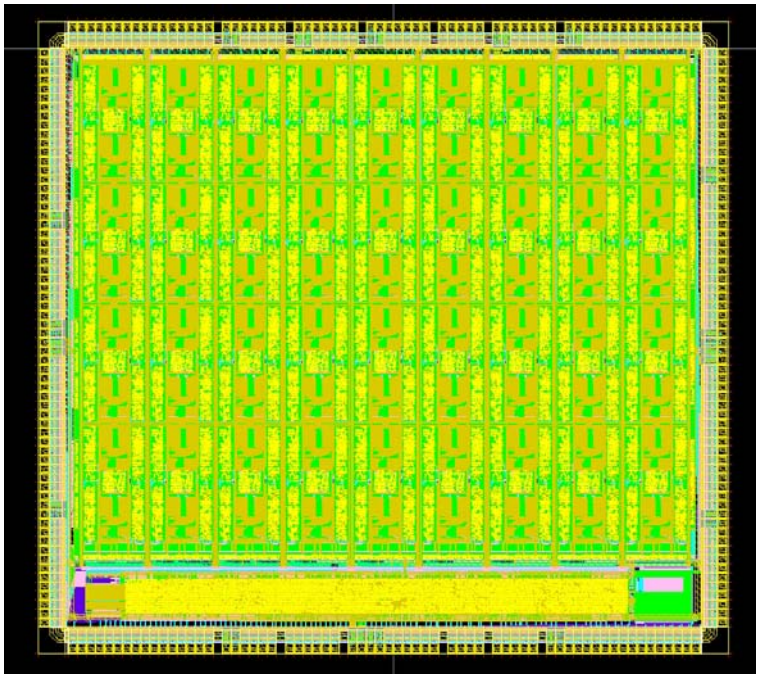

Figure 15. Layout of the POEtic chip

3. Emulation of large-scale spiking neural networks models

The spiking neural network model considered in our approach is that presented in [4]. This model outperforms previous approaches for implementing Spike Time Dependent Plasticity (STDP)-like learning methods when dealing with dynamic input stimuli.

Basically, this model consists in a leaky Integrate-And-Fire scheme, in which synapses can change their weights depending on the time difference between spikes. The outputs of the synapses are added until their result $V_i(t)$ overcomes a certain threshold θ. Then a spike is produced, and the membrane value is reset.
The simplified equation of the membrane value is:

$$V_i(t+1) = \begin{cases} 0 & when\ S_i(t) = 1 \\ k_{mem} \cdot V_i(t) + \sum J_{ij}(t) & when\ S_i(t) = 0 \end{cases} \quad (1)$$

Where $k_{mem}=\exp(-\Delta t/\tau_{mem})$, Vi(t) is the value of the membrane, J_{ij} is the output of each synapse and $S_i(t)$ is the variable which represents when there is a spike.

The goal of the synapse is to convert the spikes received from other neurons in proper inputs for the membrane. When there is a spike in the pre-synaptic neuron, the actual value of the output J_{ij} is added to the weight of the synapse multiplied by its activation variable. But if there is no pre-synaptic spike then the output J_{ij} is decremented by the factor k_{syn}. The output J of the synapse i-j is ruled by:

$$J_{ij}(t+1) = \begin{cases} J_{ij}(t) + (w_{RiRj} \cdot A_{RiRj}(t)) & when\ S_j(t) = 1 \\ k_{syn} \cdot J_{ij}(t) & when\ S_j(t) = 0 \end{cases} \quad (2)$$

Where j is the projecting neuron and i is the actual neuron. R is the type of the neuron: excitatory or inhibitory, A is the activation variable which controls the strength of the synapse, and k_{syn} is the kinetic reduction factor of the synapse. If the actual neuron is inhibitory, this synaptic kinetic factor will reset the output of the synapse after a time step, but if the actual neuron is excitatory, it will depend on the projecting neuron. If the projecting neuron is excitatory the synaptic time constant will be higher than if it is inhibitory. The weight of each synapse also depends on the type of neuron it connects. If the synapse connects two inhibitory neurons, the weight will always be null, so an inhibitory cell can not influence another inhibitory cell. If a synapse is connecting two excitatory neurons, it is assigned a small weight value. This value is higher for synapses connecting an excitatory neuron to an inhibitory one, and it takes its maximum value when an inhibitory synapse is connected to an excitatory cell.
In order to strengthen or weaken the excitatory-excitatory synapses, the variable A will change depending on an internal variable called L_{ij} which is ruled by:

$$L_{ij}(t+1)=k_{act}*L_{ij}(t) + (YD_j(t)*S_i(t)) - (YD_i(t)*S_j(t)) \quad (3)$$

Where k_{act} is the kinetic activity factor, which is the same for all the synapses.
YD is the learning variable that measures, with its decay, the time separation between a pre-synaptic spike and a post-synaptic spike. When there is a spike, YD will have its maximum value in the next time step, but when there is not, its value will be decremented by the kinetic factor $k_{learn,}$ which is the same for all synapses.
When a pre-synaptic spike occurs just before a post-synaptic spike, then the variable L_{ij} increases and the synapse strengthens. This means it reinforces the effect of a pre-synaptic spike in the soma. But when a pre-synaptic spike occurs just after a post-synaptic spike, the variable L_{ij} decreases, the synapse weakens and the effect of a pre-synaptic spike in the soma will descend. For other kind of synapses, the activation variable is always equal to 1.
Regarding the network configuration, 80% of the neurons are excitatory, while the remaining 20% are inhibitory. Each cell makes connections with other neurons within a 5x5 neighbourhood, i.e. 24 neurons. Figure 16 represents this connectivity pattern.

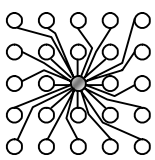

Figure 16. Connectivity of a single neuron.

The parameters that govern the functionality of the neuron block are:

- The membrane path has a resolution of 12 bits, with a range [-2048, 2047], and the threshold is kept fixed to +640.
- The membrane decay function has a time constant value of τ=20.
- The refractory time is set to 1.

The decay block will be used both in the learning and synapse blocks. Its goal is to have a logarithmic decay of the input; it is obtained with a subtraction and controlling the time when it is done depending on the input value. Taking into account that this block is used in many parts of the design, the variable decayed has been called x.
The block diagram is represented in figure 17. First of all, a new value of x should be obtained. It will be the input of a shift register which is controlled by the most significant bit of x and the external parameter *mpar*.
The output of this shift register will be subtracted from the original value of x. This operation will be done when the time control indicates it. The time control is done with the value of a counter that is compared with the result of choosing between the external value *step* or the multiplication of (MSB – *mpar*) by *step*. The decay variable τ depends on the input parameters *mpar* and *step*.

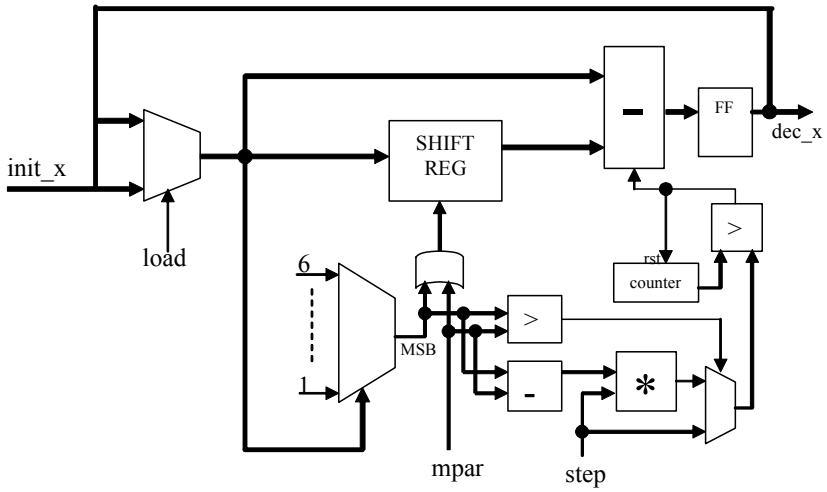

Figure 17. Block diagram of the decay block.

The learning block "measures" the time difference between a spike in the projecting neuron (j) and the actual neuron (i). Depending on these time differences and the types of the neurons, the synapse will be more or less active.

When a spike is produced in the projecting neuron, the variable YD loads its maximum value and starts to decay slowly. Then, if the actual neuron spikes, the value of YD_j is added to the decayed value of the L variable. On the other hand, if a spike is produced first in the actual neuron and after in the projecting neuron, the value of YD_i is subtracted to the decayed value of the L variable.

When the L variable overcomes a certain threshold (L_th), positive or negative, the activation variable (A) increases or decreases respectively, unless it is already in its maximum or minimum. If A is increased, L is reset to the value L-2*L_th, but if it is decreased, then L is reset to L+2*L_th. Figure 18 presents the organisation of this learning block.

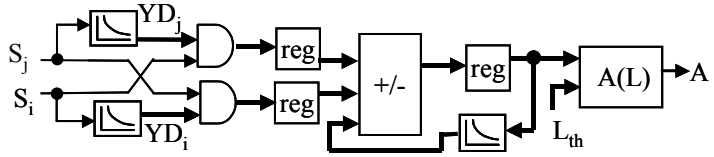

Figure 18. Organisation of the learning block.

The parameters that govern the functionality of the learning block are:

- The YD variable has a resolution of 6 bits and the learning variable (L) of 8 bits. The activation variable (A) can have four states.
- The time constant for the variable YD is τ=20.
- L_th= [-128,127]

To improve the sensitivity of the block for long time differences spikes, the time constant for the variable L is 4000, but it can change depending on the size of the network where the neuron works.

When there are spikes in the actual neuron after the spikes in the projecting neuron, the value of L increases, and the value of A also increases, so the synapse becomes more active.

The goal of the synapse block is to set the value of J (the input value added to the membrane) and it depends on four factors: the synapse activation level (A), the spikes of the projecting neuron (s_j) and the type of the actual neuron and the projecting neuron (r_i and r_j).
For each synapse a certain weight is set. This weight is multiplied by the activation variable (A). For this purpose, a shift register is used, so when A=0, the weight becomes 0, when A=1 the weight rests the same, when A=2 the weight is multiplied by 2 and when A=3 it is multiplied by 4.
This output weight is added to the decayed value of the output J. But the decay curve depends on the type of the actual and the projecting neurons (r_i and r_j).
There are two possible types for each neuron, excitatory and inhibitory, so we should obtain four possible values for the time constant which will decrease the addition. But, when both neurons are inhibitory, the weight of the synapse is always 0, so the J value is also always 0 and therefore it is nonsense to decrease it. Due to this reason, there are only three possible decay time constants.
The three time constants are multiplexed, and the multiplexer is controlled by the types of neurons (r_i,r_j). The multiplexer output controls the decay block, and finally we obtain the J value at the output of this decay block. Figure 19 shows the organisation of the synapse block.

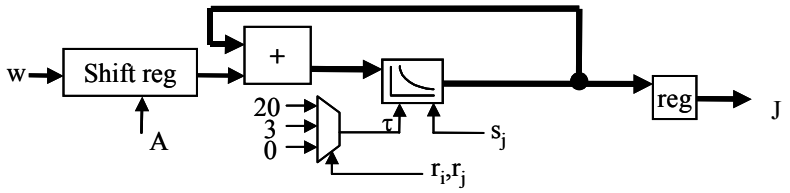

Figure 19. Organisation of the synapse block.

The parameters that govern the functionality of the synapse block are:

- The internal resolution of the block is 10 bits. But the output resolution is of 8 bits, due to the internal value of J is divided by 4 to keep the correct scaling.
- The time constants used by this block are presented in Table 3.

Time Constant (τ)	Projecting Neuron Type (r_j)	Actual Neuron Type (r_i)
20	0	0
0	0	1
3	1	0
0	1	1

Table 3. Time constants for different synapse types.

In this table r=0 means an excitatory neuron, while r=1 indicates an inhibitory neuron.

The high resolution needed for the variables, as well as the number of operations to be performed may pose a serious limitation for the final implementation. Therefore, the first step in the physical realisation of the model has consisted in an evaluation of the minimum resolution to be used in the neuron data path.

In a first attempt the resolution of the parameters has been reduced by two bits and some values and time constants have been changed to keep the correct scaling. Table 4 shows the new values of the internal parameters after this optimisation process. The final organisation resulting from this optimisation process is depicted in figure 20.

Due to the complexity of the design, the simplification of the model is very important to avoid redundancy or to use just the necessary components. For this reason, a further simplification of all the building blocks that constitute the model has been performed [5].

Parameter	New value
Membrane resolution	10
Threshold	+160
Input (J) resolution	6
Weights	[0:8],[64:128],[128:256],[0:0]
YD resolution	4
L resolution	6
Membrane decay time constant	20
YD decay time constant	20
L decay time constant	4000
J decay time constants (00,01,10,11)	20,0,3,0 (keep the same values)

Table 4. Resolution of the parameters for an optimised implementation.

Once the model has been optimised it has been physically translated into the molecules that constitute the basic building blocks of the organic subsystem of the POEtic tissue. Figure 21 shows this physical realisation.

The molecule organisation shown in figure 21 corresponds to the actual structure of the organic subsystem present in the POEtic tissue, which is arranged as an 8x18 array of molecules.

After designing the neuron model the VHDL models developed for the POEtic tissue have been configured and simulated to validate its functionality.

After this validation stage the strategy for the simulation of large-sale SNN models has been considered. Since in its actual implementation the POEtic chip only allows for the implementation of a single neuron and the current number of POEtic chips is far less than 10000 it will be necessary to use a smaller array of POEtic chips whose functionality should be time multiplexed in order to emulate the whole network.

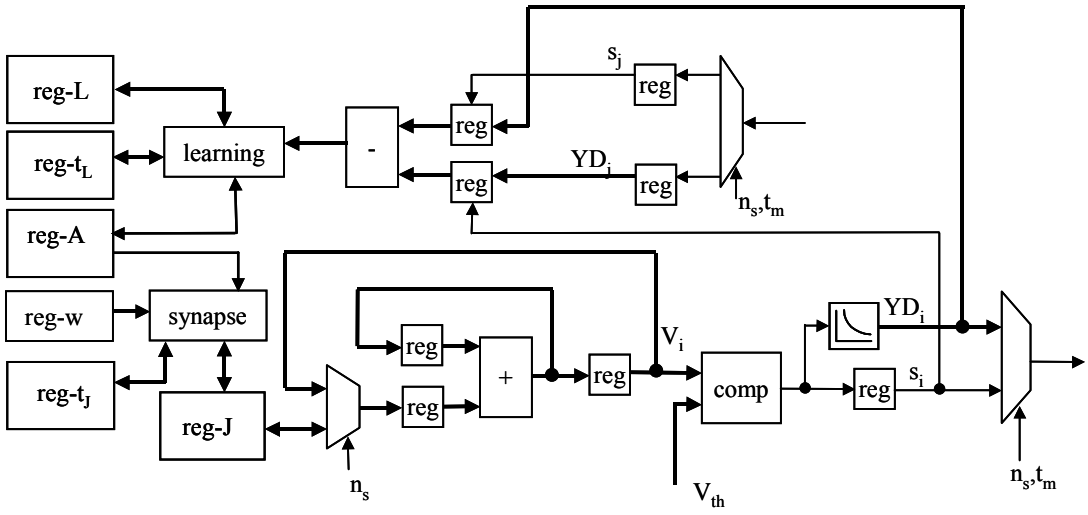

Figure 20. Block diagram for the serial implementation of the neuron model.

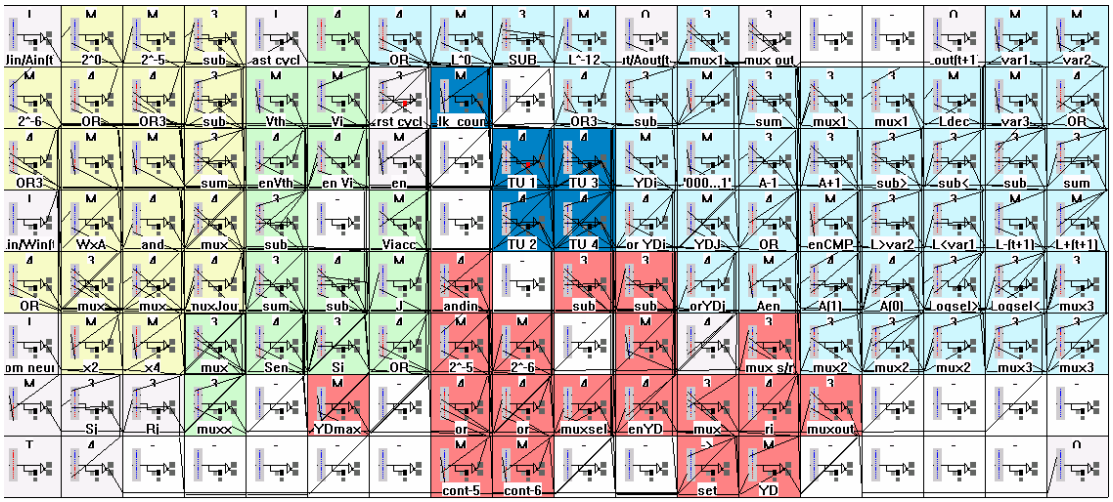

Figure 21. Molecule-level implementation of the neuron model.

This means that every POEtic chip should be able to manage a local memory in charge of storing the weights and learning variables corresponding to the different neurons it is emulating in time.

A 16-neuron network organised as a 4x4 array has been constructed using this principle. This would permit the emulation of a 10000-neuron network in 625 multiplexing cycles. Bearing in mind that each neuron is able to complete a single cycle in 150 clock cycles, this means that the minimum clock frequency required to handle input stimuli in real time (i.e., to process visual input stimuli at 50 frames/second) is around 5 MHz, far less than the actual clock frequency achievable by the organic subsystem of the POEtic tissue.

The visual stimuli will come from an OmniVision OV5017 monochrome 384x288 CMOS digital camera. Specific VHDL and C code have been developed in order to manage the digital images coming from the camera. To test the application, artificial image sequences have been generated on a display and then captured by the camera for its processing by the network.

4. Conclusions

In this paper we have presented a new family of programmable integrated electronic systems, called POEtic, that include features derived from some of the properties present in living beings, like evolution, development, self-repair, self-replication and learning.

The combination of partial and total dynamic reconfiguration, as well as the self-configuration and dynamic routing capabilities make these devices an ideal candidate for the efficient implementation of bio-inspired artefacts.

After describing in detail the different building blocks that constitute the tissue, an implementation approach for the emulation of large-scale spiking neural network models has been presented. The results derived from this implementation demonstrate that an electronic tissue built around these devices will permit the real-time emulation of this kind of models, thus serving as an excellent development and experimentation instrument for neuroscientists.

After receiving the first POEtic chips specific development boards have been constructed to develop applications to be solved using the bio-inspired features offered by the tissue.

Acknowledgements

The work presented in this paper has been funded by the grant IST-2000-28027 (POEtic) of the European Union (FET Proactive initiative on Neuroinformatics for living artefacts) and by grant OFES 00.0529-2 of the Swiss government. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] ARM. Amba specification, rev 2.0. advanced risc machines ltd (arm). http://www.arm.com/armtech/amba_spec, 1999.

[2] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] J. M. Moreno Arostegui, E. Sanchez, and J. Cabestany. An in-system routing strategy for evolvable hardware programmable platforms. In *Proc. 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 157–166. IEEE Computer Society Press, 2001.

[4] J. Eriksson, O. Torres, A. Mitchell, G. Tucker, K. Lindsay, D. Halliday, J. Rosenberg, J.M. Moreno, A.E.P. Villa, "Spiking Neural Networks for Reconfigurable POEtic Tissue", Evolvable Systems: From Biology to Hardware, A.M. Tyrrell, P.C. Haddow, J. Torresen (eds.), pp. 165-173, Springer-Verlag, 2003.

[5] O. Torres, J. Eriksson, J.M. Moreno, A. Villa, "Hardware optimization and serial implementation of a novel spiking neuron model for the POEtic tissue", BioSystems, Vol. 76, No. 1-3, pp 201-208, August_October 2004.