The Perplexus bio-inspired reconfigurable circuit

Andres Upegui, Yann Thoma,
Eduardo Sanchez, Andres Perez-Uribe
HEIG-VD, Yverdon-les-Bains, Switzerland
(andres.upegui, yann.thoma, eduardo.sanchez,
andres.perez-uribe)@heig-vd.ch

Juan Manuel Moreno, Jordi Madrenas
UPC, Barcelona, Spain
(moreno, madrenas)@eel.upc.edu

Abstract

This paper introduces the ubichip, a custom reconfigurable electronic device capable of implementing bio-inspired circuits featuring growth, learning, and evolution. The ubichip is developed in the framework of Perplexus, a European project that aims to develop a scalable hardware platform made of bio-inspired custom reconfigurable devices for simulating large-scale complex systems. In this paper, we describe the configurability and architectural mechanisms that will allow the implementation of evolvable and developmental cellular and neural systems in an efficient way. These mechanisms are dynamic routing, self-reconfiguration, and a neural-friendly logic cell's architecture.

1. Introduction

The Perplexus project [12] aims to develop a scalable hardware platform made of custom reconfigurable devices endowed with bio-inspired capabilities. This platform will enable the simulation of large-scale complex systems and the study of emergent complex behaviors in a virtually unbounded wireless network of computing modules.

The Perplexus platform will consist thus in a scalable network of *ubiquitous computing modules* (*ubidules*) equipped with wireless communication capabilities and rich sensory elements. The platform will be modular for allowing the application developer to customize his platform setup. In this way the application developer can easily build his system setup by selecting what to plug to the *ubidule* from a set of peripherals. These peripherals can be different communication interfaces (wifi, bluetooth), sensors, actuators, cameras, or flash memories. This modularity is guaranteed by the use of standard interfaces such as USB.

At the heart of these *ubidules*, we will use a custom reconfigurable electronic device capable of implementing bio-inspired mechanisms such as growth, learning, and evolution. These bio-inspired mechanisms will be possible thanks to reconfigurability mechanisms like dynamic routing, distributed self-reconfiguration, and a simplified connectivity. Such an infrastructure will provide several advantages compared to classical software simulations: speed-up, an inherent real-time interaction with the environment, self-organization capabilities, simulation in the presence of uncertainty, and distributed multi-scale simulations. Our modeling framework will be tested on three application domains: biologically-plausible developing neural networks modeling, culture dissemination modeling, and cooperative collective robotics [12].

Recent work in this field is the POEtic tissue [16], a reconfigurable hardware platform for rapidly prototyping bio-inspired systems that employ POE principles [13], which has been developed in the framework of the European project POEtic. The POEtic chip has been specifically designed to ease the development of bio-inspired applications.

The limitations exhibited by the POEtic tissue suggest several architectural and configurability features to be improved. These improvements may lead us to a reconfigurable platform better suited for supporting the bio-inspired principles that we want our devices to mimic.

Before discussing the hardware mechanisms that will be considered for the design of our *ubichip*, in section 2 we will introduce some concepts and issues concerning bio-inspired hardware, also known as POE hardware [13], and we will also present the desired bio-inspired features to be supported by the *ubichip*. Then, in section 3, we will describe the hardware mechanisms that will allow the implementation of such bio-inspired systems.

2. POE Hardware

Living beings, unlike engineered systems, exhibit a high level of adaptability and robustness thanks to several biological mechanisms: reproduction, learning, self-repair, growth, and evolution. It would be thus desirable to include such mechanisms in human-designed systems in or-

Second NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007)
0-7695-2866-X/07 $25.00 © 2007 IEEE

der to increase their lifetime and to improve their adaptability to the environment and to the user. If one considers life, as we know it, the following three levels of organization can be distinguished [13]: (1) Phylogeny, concerning the temporal evolution of a certain genetic material in individuals and species, (2) Ontogeny, concerning the developmental process of multicellular organisms, and (3) Epigenesis, concerning the learning process during an individual's lifetime. Analogous to nature, the space of bio-inspired hardware systems can be partitioned along these three axes, to which we refer as the POE model.

When we refer to the *phylogenetic* axis of bio-inspired hardware systems, we are talking about "Evolvable Hardware" (EHW). EHW makes use of evolutionary algorithms in order to define a description of a hardware system. From a desired behavior specification of a circuit, an evolutionary algorithm will search for a circuit configuration describing a satisfactory solution to the specification. If one examines the work carried out to date under the heading EHW, one can identify four taxonomic subdivisions according to the level of bio-inspiration: extrinsic, intrinsic, complete, and open-ended evolution [13]. The *ubichip* must provide thus the capabilities for performing each of the aforementioned levels of evolution. The capability of evolving at these four levels will allow our *ubidules* to evolve, in a completely autonomous way, under a real-time interaction with the environment and under the presence of uncertainty.

The *ontogenetic* axis comprises several mechanisms of high interest for inclusion in human-designed systems. Self-replication and self-repair are two key characteristics of living beings that are still far from being exploited by engineered systems with an efficiency comparable to nature. However, some key factors from multicellular beings have been identified for use in the design of ontogenetic machines: the dependence of cell's functionality upon its relative position, the relevance of the physical neighborhood for chemical interactions between cells, the importance of time scales during cellular reproduction, and the fundamental role played by protein's regulation and cell's differentiation, which is driven by regulatory and differentiation genes. Research projects as *Embryonics* [6] (embryonic electronics) and *POEtic* [1, 10, 11] have studied the issues related to hardware implementations of such mechanisms.

The *epigenetic* axis of bio-inspired hardware systems mainly refers to hardware implementations of artificial neural networks, also known as "neural hardware". Most neural models are conceived for being implemented in software platforms, making them unsuitable for hardware implementations. These models don't take care about data resolution, floating point operations overhead, or multiplications, since their overall overhead in software is negligible or nonexistent. These aspects turn out to be very expensive when one considers their implementation as a hardware architecture. Some previous works have focused on optimizing the implementation of such types of models, and other works have focused on proposing original models that exploit better the hardware specificity of the implementation [9, 18].

The implementation of such bio-inspired features on a hardware substrate requires some special hardware mechanisms to be provided by the underlying reconfigurable architecture. These mechanisms must allow an efficient use of hardware resources when designing POE circuits.

3. Ubichip mechanisms

The system architecture envisioned for the *ubichip* is represented in figure 1, and it is composed of four main parts. (1) The encoder/decoder is in charge of managing the shared address bus that implements the inter-chip communication with an address event representation (AER) scheme (to be further explained in subsection 3.4). (2) The memory controller takes care of handling the data RAM needed to store system parameters as well as the CAM (Content Addressable Memory) that will be used to implement the AER communication scheme between neurons. (3) The system manager handles the overall configuration of the *ubichip* and its interface with the main controller of the *ubidule*. For the neurobiological modeling application, it is envisioned to implement a SIMD-like solution with a centralized sequencer and a set of reconfigurable neural units. The sequencer included in this subsystem interprets the code corresponding to the execution of the neurons to be implemented in the configurable section of the *ubichip*. A small instruction set has been specifically designed for this sequencer. This instruction set is general enough to permit the implementation of any parallel system. Furthermore, it includes conditional store instructions so as to permit a linear execution of the code, thus improving the concurrence of the system. Finally, (4) the configurable array consists in a bi-dimensional regular array of elementary reconfigurable cells.

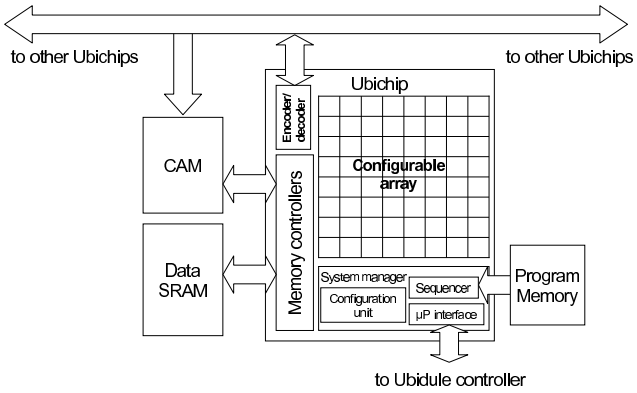

Figure 1. System Architecture of the ubichip.

Second NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007)
0-7695-2866-X/07 $25.00 © 2007 IEEE

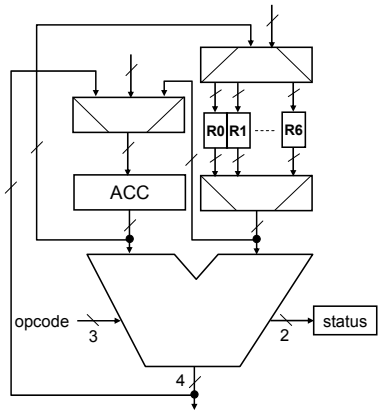

Figure 2. Configurable cell in 4-bit ALU mode.

Bioinspired circuits require a number of reconfigurability and architectural features in order to be implemented in an efficient way. In this section we describe the hardware mechanisms that will allow the *ubichip* to implement circuits able to adapt by means of evolution, development, and learning.

3.1 Neural-friendly Architecture

Among the applications being targeted in Perplexus, the neurobiological modeling application imposes the more critical computing power requirements. In the specific neural application considered within the framework of Perplexus [4, 5], the hardware platform should be able to simulate the functionality of a spiking neural network constituted by 10000 neurons, and each neuron establishes on average 300 synaptic connections with other neurons.

A careful analysis has been carried out in order to define the internal organization of the reconfigurable cells, in order to allow an efficient implementation of such neural network. Taking into account the trade-off to be achieved between the integration density and the granularity of the basic cells, it has been decided to define their structure around four 4-LUT units that can be configured as four independent 4-input functions or as a 4-bit ALU. This 4-bit ALU will allow the implementation of the neural processing units that will be controlled by the sequencer. The memory elements required to implement the LUT units will be designed so that they can also be used as an 8 x 4-bit register file for the ALU, so as to optimize the implementation of the arithmetic and logic instructions. The architecture of a basic cell in 4-bit ALU mode is depicted in figure 2.

The neural-friendly architecture is provided thus in the form of a reconfigurable unit array that can be configured as an efficient neural SIMD multiprocessor. A single 4-LUT logic unit can act as a 4-bit SIMD processing element (PE), or it can be assembled to neighbor units to form an n-bit PE (being n a multiple of 4). A neural-oriented instruction set, specifically defined for these PEs, guarantees an optimal implementation of spiking neural systems, allowing a single processing element to compute a plurality of neurons and synapses.

3.2 Self-reconfiguration

Ontogenetic features correspond to the way an organism develops from a single cell to an entire organism (self-replication), as well as to the capability of self-repair. Both processes of self-replication and self-repair require cellular replication and differentiation. Although differentiation can act at system level to simply express a particular functionality depending on some factors, self-replication requires specific hardware mechanisms.

Our concept of self-replication considers the case of a given circuit, lets call it a *cell*, configured on a reconfigurable array of logic units, lets call them *molecules*. This cell is able to self-replicate, by creating a complete and exact copy of itself somewhere else on the reconfigurable array. In our case, self-replication is performed by means of *self-inspection* [7]. The self-inspection concept performs the replication by directly self-inspecting the content of the cell in order to replicate it somewhere else. In the case of a reconfigurable circuit, this content corresponds to the configuration bits of the reprogrammable elements. The advantage of this approach is that there is no need to duplicate information, because the cell content is directly scanned. This gain in terms of data storage leads to a more complex hardware, capable of supporting this inspection.

In our self-replication process, we split the cell into three organelles, each performing a different function in the cell during self-replication, as shown in figure 3: a functional unit (FU), and two replication units (RU0 and RU1) responsible for the self-replication process.

The self-replication algorithm is decomposed into three steps: (1) RU0 creates a copy of RU1 somewhere on the reconfigurable array, (2) RU1 creates a copy of RU0, and (3) RU0 creates a copy of FU, by connecting to the newly created RU0.

This algorithm, while being quite simple, requires special hardware support (for instance, in the POEtic chip, it was not possible to implement it): connections have to be created at runtime, molecules have to allow for a self-inspection process to retrieve configuration bits, and molecules have to allow also the creation of the configuration path in order to determine the cell's morphology.

This process of self-replication requires specialized configurability mechanisms on the replicator and replica side. The replicator cell needs to inspect itself to retrieve its configuration bits, while the replica needs to build itself.

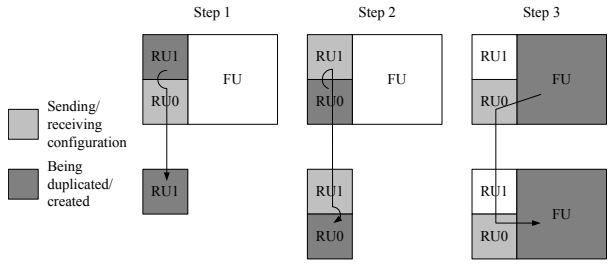

Figure 3. Self-replication principle.

For this purpose, we propose the THESEUS mechanism (standing for *Theseus-inspired Embedded SElf-replication Using Self-reconfiguration*) in order to build the cell [17]. In THESEUS, an organelle is defined by a *genome* that is composed of 2 parts: (1) a set of special flags for defining the cell's morphology and (2) the configuration for each of the molecules forming the cell. We have borrowed an idea from the Tom Thumb algorithm [7] in order to manage the morphogenetic development of cell's organelles by creating a configuration path. The idea we borrowed is the way a flag is loaded into a molecule and serves then to indicate where to continue the cell construction.

The creation of an exact copy of the organelle requires to be able to recover the genome in the exact order that it was sent. So the result of pulling the configuration string must be the obtention of the same genome, that has been previously introduced. Our solution to this problem is to virtually create a shift register following the path used for the cell shape creation, and to traverse it in both directions.

For a more detailed description of the THESEUS mechanism, including cell's construction, self-inspection, replication, and hardware implementation, please refer to [17].

3.3 Dynamic Routing

Ontogenetic processes, such as self-replicating mechanisms described in the previous section, require the ability of creating paths at runtime, in order to connect newly created cells. Epigenetic systems such as growing neural networks would also need to build connectivity during the lifetime of the artificial network. Therefore the *ubichip* has to propose a hardware mechanism to handle this kind of dynamic routing.

Considering the typical high silicon overhead due to routing matrices, specially high for dynamic routing, we chose a solution requiring the less amount of logic as possible, while being flexible enough to deal with the changing topology of the network. One of the simplest physical realization consists in the wormhole routing concept [8]. However the hardware overhead of this kind of algorithms is not suitable for the granularity of the reconfigurable array. Therefore, we will implement a dynamic routing algorithm, by improving the routing implemented in the POEtic chip [15]. The risk of congestion will be reduced by means of two features. First, the new algorithm will better exploit the existing paths, and second an 8-neighborhood will allow a dramatic reduction of congestion risk compared to the amount of logic required. And finally, while in POEtic the circuit execution was frozen during a routing process, in the *ubichip* the creation of a new path will let the system run without interruption.

The basic idea of the algorithm is to construct paths between sources and targets by dynamically configuring multiplexers, and by letting the data follow the same path for each pair of source and target. A phase of path creation executes a breadth-first search distributed algorithm, looking for the shortest path. Sources and targets can decide to connect to their corresponding unit at any time by launching a routing process.

Special routing units will be implemented in hardware. They will be composed of the multiplexers needed to route the data, the corresponding registers required to store the multiplexers configuration, and a finite state machine to handle the routing processes. The routing units will be connected to the logic units in order to allow the cells (neurons for instance) to manage the creation of new connections.

A routing process, dealing the construction of a new path, is decomposed in 5 main phases: (1) When a source or a target wants to initiate a connection, it activates a global signal. If different logic units start such a process at the same time, priority is given to the most bottom-left unit. (2) After a master is identified, it sends serially its ID. This ID will be stored in a LUT of the logic unit, exploiting the configuration bits shift register. (3)At the end of the address broadcasting, all sources and targets have compared the data with their own address and know if they are involved in the current process. (4)A breadth-first search algorithm then searches for the shortest path. (5) When the target is found, a backward signal allows for the configuration of the multiplexers present on the path, and the process ends up with a new path, allowing the newly connected logic unit to share information.

3.4 Scalability Issues

One of the most salient features of complex systems is the dense interaction scheme established between their constituent components. This implies that special attention has to be paid to the scalability properties of any hardware platform envisioned for the efficient implementation of complex systems. That is, the main figures of merit of the platform should be kept irrespective of the number of physical units (chips in the case of the Perplexus platform) that constitute it and also irrespective of the partitioning done (i.e., the number of components that are simulated on a single

physical unit).

If one considers the neural application described in subsection 3.1, it means that if 100 neurons could be physically mapped in a single chip (a number that still has to be verified), the number of I/O pins required per chip would exceed 20000, far more than can be attained with any foreseeable packaging technology. One of the major hardware issues to be faced by the Perplexus project is related to the scalability of the basic building blocks (*ubichips*) that will constitute the Perplexus platform. Among the different approaches that may provide a feasible solution for the I/O scalability problem it is foreseen to analyse and adapt the principles involved in the Address Event Representation (AER) scheme, initially proposed in [14] and developed later in [2]. This communication mechanism was developed in order to overcome the bottlenecks that appear when information has to be exchanged within a system composed of massively interconnected components. The principle of the AER scheme consists in converting an ordered sequence of events (spikes in the case of a spiking neural network) into a sequence of addresses that encode the source of the event and that are broadcasted to the rest of the system. In the receiver side, the sequence of addresses are converted again into a sequence of events that are transferred to the corresponding destinations.

The AER communication scheme can be easily adapted to the computational needs of a distributed system such as that constituted by the Perplexus platform, where a *ubidule* can contain more than one *ubichip*. A global bus containing the address of the source components that generate events at a given time is shared between all the *ubichips*. Every *ubichip* contains an encoder unit that converts the events generated by the components contained in it into addresses to be placed in the shared bus, and also a decoder unit that translates the addresses present in the shared bus into events for its implemented components. The arbitration for the access to the bus can be established in a sequential way between all the *ubichips* present in the system. In this way, every *ubichip* will indicate to the next one by means of a specific signal, *start_frame*, that it is accessing the shared bus and broadcasting the addresses corresponding to the events generated by its components. Another signal, *end_frame*, would indicate that its access to the bus has finished and that the next *ubichip* is granted to access the bus. When the last *ubichip* generates the *end_frame* signal, the first one will activate a global signal, called *frame_update*, so that all the components included in all the *ubichips* may update their outputs from the inputs received in the current simulation frame. Figure 4 represents the system organisation for the implementation of this communication scheme. The boxes labeled as *C* in the figure are the components that correspond to the PEs introduced in the subsection 3.1, which are implemented in the different *ubichips*.

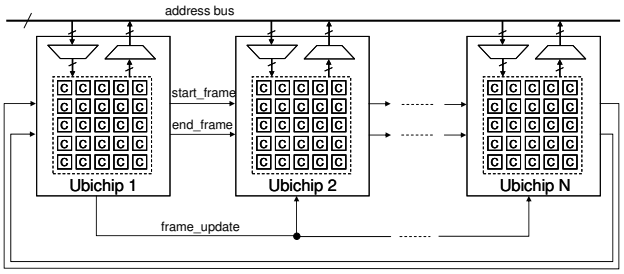

Figure 4. AER implementation.

It is worth noting that, because it basically consists on a multiplexed broadcasting of information, this communication scheme is valid either for a single-*ubichip* per *ubidule* scenario or for a multi-*ubichip* per *ubidule* scenario. Furthermore, this communication scheme may provide enough bandwidth for the communication needs of the applications considered in the Perplexus project, even in the single-*ubichip* per *ubidule* scenario and a wireless physical link between *ubidules*. If we consider the neural application (the most restrictive one in terms of capacity and bandwidth), and assuming 100 neurons are implemented in each *ubichip* and a 54 Mbits/second wireless link, this would permit a neuron firing rate of around 300 spikes/second, something that is in line with the simulation experiments already performed for the application [3]. In the case of a multi-*ubichip* per *ubidule* scenario with a shared bus running at 10 MHz (a quite conservative approach) this would imply a firing rate of around 1000 spikes/second, far exceeding the application needs.

Finally, it is also worth noting that this communication scheme permits a local synchronous implementation of the target functionality and an asynchronous information exchange, something that fits well with the scalability features to be attained by the Perplexus platform. Additionally, the proposed communication scheme permits to synchronize the overall emulation of the target system in the platform, a strict requirement in some applications like the spiking neural network that is being considered within the framework of the project.

4 Summary

In this paper we have introduced the *ubichip*, a custom reconfigurable electronic device capable of implementing bio-inspired hardware systems featuring growth, learning, and evolution. We have also presented the architectural and reconfigurability mechanisms that will allow an efficient implementation of such systems. These mechanisms are dynamic routing, distributed self-reconfiguration, and a neural-friendly logic cell architecture, keeping in mind

the scalability issues that rise in the implementation of such type of complex systems.

This *ubichip* will constitute the core of the *ubidule*, the supporting platform of the Perplexus project [12], which goal is to develop a scalable hardware platform made of custom bio-inspired reconfigurable devices for simulating large-scale complex systems. This *ubichip* will be associated to sensory elements, actuators, and wireless communication capabilities.

Acknowledgements

This project is funded by the Future and Emerging Technologies programme IST-STREP of the European Community, under grant IST-034632 (PERPLEXUS). 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] W. Barker, D. M. Halliday, Y. Thoma, E. Sanchez, G. Tempesti, and A. M. Tyrrell. Fault tolerance using dynamic reconfiguration on the poetic tissue. *IEEE Transactions in Evolutionary Computation*.

[2] K. Boahen. Point to point connetivity between neuromorphic chips using address events. *IEEE Trans. on Circuits and Systems II*, 47(5):416–434, 2000.

[3] J. Iglesias. *Emergence of Oriented Circuits driven by Synaptic Pruning associated with Spike-Timing-Dependent Plasticity (STDP)*. Phd thesis, University Grenoble I Joseph Fourier, University of Lausanne, 2005.

[4] J. Iglesias, J. Eriksson, F. Grize, M. Tomassini, and A. Villa. Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems*, 79(1-3):11–20, 2005.

[5] J. Iglesias, J. Eriksson, B. Pardo, M. Tomassini, and A. Villa. Emergence of oriented cell assemblies with spike-timing-dependent plasticity. In W. Duch et al., editor, *Artificial Neural Networks: Biological Inspirations, LNCS*, volume 3693, pages 11–20. Springer-Verlag, 2005.

[6] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–540, April 2000.

[7] D. Mange, A. Stauffer, E. Petraglio, and G. Tempesti. Self-replicating loop with universal construction. *Physica D*, 191(1-2):178–192, apr 2004.

[8] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[9] D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano. Hardware spiking neural network with run-time reconfigurable connectivity. In *5th NASA / DoD Workshop on Evolvable Hardware (EH 2003)*, pages 199–208. IEEE Computer Society, 2003.

[10] D. Roggen, Y. Thoma, and E. Sanchez. An evolving and developing cellular electronic circuit. In J. Pollack et al., editors, *Proc. Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, pages 33–38, Cambridge, Massachusetts, USA, 2004. The MIT Press.

[11] J. Rossier, Y. Thoma, P.-A. Mudry, and G. Tempesti. MOVE processors that self-replicate and differentiate. In A. Ijspeert et al., editors, *Proc. Biologically Inspired Approaches to Advanced Information Technology (BioADIT 2006)*, number 3853 in LNCS, pages 160–175, Berlin Heidelberg, 2006. Springer-Verlag.

[12] E. Sanchez, A. Perez-Uribe, A. Upegui, Y. Thoma, J. Moreno, A. Villa, H. Volken, A. Napieralski, G. Sassatelli, and E. Lavarec. PERPLEXUS: Pervasive computing framework for modeling complex virtually-unbounded systems. In *AHS 2007 - Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems*, 2007.

[13] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, 1997.

[14] M. Sivilotti. *Wiring Considerations in Analog VLSI Systems With Applications to Field Programmable Networks*. Phd thesis, California Institute of Technology, Passadena, 1991.

[15] Y. Thoma, E. Sanchez, J. M. M. Arostegui, and G. Tempesti. A dynamic routing algorithm for a bio-inspired reconfigurable circuit. In *Field-Programmable Logic and Applications, LNCS*, volume 2778, pages 681–690. Springer-Verlag, 2003.

[16] Y. Thoma, G. Tempesti, E. Sanchez, and J. M. M. Arostegui. POEtic: an electronic tissue for bio-inspired cellular applications. *Biosystems*, 76(1-3):191–200, 2004.

[17] Y. Thoma, A. Upegui, A. Perez-Uribe, and E. Sanchez. Self-replication mechanism by means of self-reconfiguration. In *Workshop Procedings of the International Conference on Architecture of Computing Systems 2007 (ARCS'07)*. VDE Verlag, Berlin, 2007.

[18] A. Upegui, C. A. Peña Reyes, and E. Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29(5):211–223, 2005.