POEtic: A Prototyping Platform for Bio-inspired Hardware

J. Manuel Moreno[1], Yann Thoma[2], and Eduardo Sanchez[2]

[1] Technical University of Catalunya, Dept. of Electronic Engineering, Campus Nord, Building C4, c/Jordi Girona 1-3, 08034-Barcelona, Spain
moreno@eel.upc.edu

[2] Swiss Federal Institute of Technology Lausanne, Logic Systems Laboratory, IN-Ecublens, CH-1015, Switzerland
{Yann.Thoma, Eduardo.Sanchez}@epfl.ch

Abstract. This paper will present the final hardware realization of a new family of programmable devices that has specifically being conceived in order to address the prototyping of bio-inspired principles. The devices are organized around a custom 32-bit RISC microprocessor and a custom FPGA. The internal architecture devised for the devices is scalable, so that it is possible to construct a physical hardware platform whose size matches the requirements of the application to be handled. To facilitate the development of applications for this hardware platform a complete set of design tools has been developed.

1 Introduction

During the last years standard programmable devices have been extensively used to provide physical implementations for bio-inspired principles, either as a direct substrate [1] or as a supporting platform for extended architectures [2], [3], [4]. Even some custom programmable architectures [5], [6] have been developed in order to provide an efficient substrate for the realization of these principles. However, there is still a lack of an integrated system able to offer at the same time the basic features required to implement actual autonomous bio-inspired hardware:

- Partial dynamic reconfiguration, i.e., the ability to modify the functionality of a section of the design while it is in normal operation and with a delay comparable to the execution delay of the system. Even if this capability is being offered by the programmable devices offered by Xilinx [7] and Atmel [8] it is usually limited by the lack of information about the physical configuration string or by the granularity of the reconfiguration area.
- Self-configuration, i.e., the capability of the programmable device of modifying its functionality using its own resources. This feature is already present in the Cell Matrix devices [5].
- Dynamic routing, i.e., the possibility of changing in real time the connectivity between the elementary programmable cells included in the system without the need for an external compiler.

The main goal of the POEtic project [9] was the development of a flexible hardware substrate able to provide capabilities similar to those present in living beings, like evolution, development, self-replication, self-repair and learning. One of the major outcomes of the project was an integrated programmable electronic system, the POEtic chip, that provides in a single device the three features mentioned above: partial dynamic reconfiguration, self-configuration and dynamic routing. We shall demonstrate that the combination of these capabilities in a single hardware substrate provides an efficient platform for the prototyping and development of artifacts based on bio-inspired principles.

The rest of the paper is organized as follows: in the next section we shall present the major features included in the architecture conceived for the POEtic devices. Then we shall review the actual physical implementation of the device and the development environment that has been built around it. Finally, the conclusions and our current work will be outlined.

2 The POEtic Architecture

The structural organization of a POEtic chip is represented in Fig. 1.

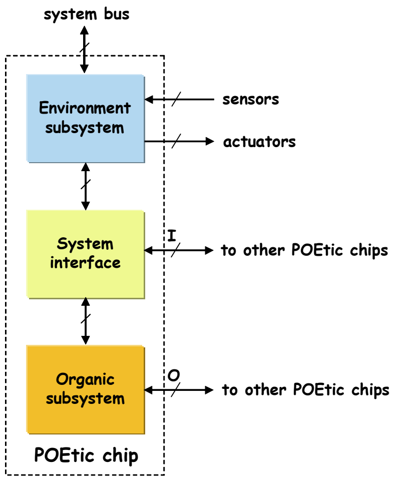

Fig. 1. Organization of the POEtic chip

As it can be deduced from this figure, a POEtic chip is constituted by three main building blocks:

- **Environment subsystem:** This is the component of the tissue that is in charge of managing the interaction with the environment. This interaction can be considered at two different time scales: on-line interaction and evolution (phylogenesis). The on-line interaction refers to the continuous process by means of which a given individual implemented in the tissue is sensitive to the input stimuli that arrive from the external environment. These stimuli may take the form of any physical magnitude (light, pressure, temperature, …), and after a conditioning and conversion processes are translated into internal signals that may be used by the individual either to extract some knowledge from the environment or to produce an output as a result of some internal processing. These output signals may be later translated, by means of a set of proper actuators, into physical magnitudes that are reverted as output actions to the environment. This on-line interaction constitutes the basic sensor-actuator loop that permits a given individual to adapt its behaviour to the specific characteristics of the environment where it is placed. The second kind of interaction with the environment acts at a population level and exceeds the life time of an individual. In this case the sensor-actuator loop is used to define the basic substrate (the genome) of the individuals that are capable of adapting its behaviour to the environment in the most efficient way according to a given fitness measure.

- **Organic subsystem:** It is in charge of implementing the behaviour of an individual, following the principles described by the innate information that has resulted from the evolutionary process. Therefore, it is the goal of this system to permit the development (ontogenesis) of a given functionality from the information stored in a genome, and also to permit the adaptation (epigenesis) of this functionality according to the stimuli received from the environment.
- **System interface:** This element will allow for an efficient communication between the environment and the organic subsystem of the tissue. It also constitutes the substrate that will provide the basic mechanisms that will permit the scalability of the tissue.

2.1 The Environment Subsystem and System Interface

The environment subsystem of the POEtic tissue has been built around a custom 32-bit microprocessor with an efficient and flexible system bus, based on the AMBA specification [10], and several custom peripherals. The reason for using a centralised system to carry out evolutionary processes is motivated by the fact that, even if evolution acts on a population of individuals, at the end there must be a global unit that should evaluate the fitness of the individuals and determine those from which the next population has to be constructed. Therefore, the functionality of the individuals will be implemented in the organic subsystem, but it is the microprocessor that constitutes the core of the environment subsystem that will drive the basic steps of the evolutionary process, as well as the interaction of the individuals with the environment. Additionally, the use of a programmable unit to implement the phylogenetic mechanisms of the tissue will permit to test and develop different evolutionary strategies, since this will imply just an update of the software executed by the microprocessor. Finally, this alternative will largely simplify the management of the acquisition/conversion units that are required to handle the sensor-actuator loop needed to complete the epigenetic and phylogenetic processes to be implemented by the tissue.

The system interface of the Poetic device plays a major role in allowing for its scalability features. This means that it is possible to connect several POEtic chips in order to construct an electronic tissue whose size 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.

In this way, a POEtic tissue can be constructed as a bidimensional array constituted by POEtic chips. The connectivity between these chips is based on two different buses, named organic (O) and interface (I) buses. The signals that constitute the organic bus permit to communicate (at a cellular level) the organic subsystems present in every POEtic chip. 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 Fig. 2.

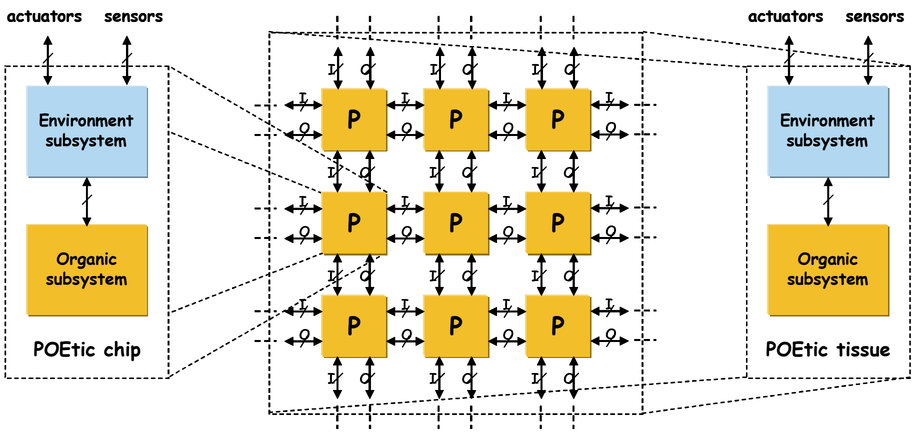

Fig. 2. Scalability properties of the POEtic tissue

2.2 The Organic Subsystem

The organic subsystem of the POEtic device is made up of 2 layers, as depicted in Fig. 3: 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.

A molecule has eight different operational modes, to speed up some operations, and to use the routing plane.

- 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).

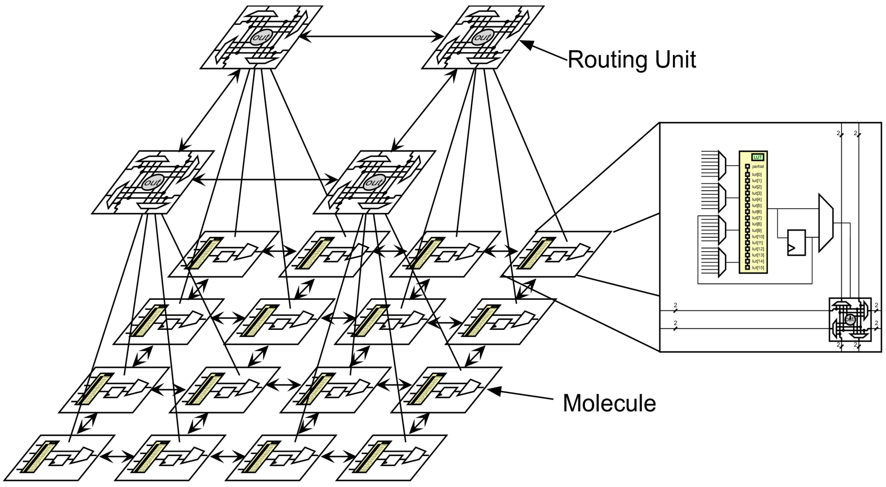

Fig. 3. Organization of the organic subsystem

- 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.
- In **Output** mode, the molecule is a cellular output, connected to the inter-cellular routing plane.
- In **Trigger** mode, the 16-bit shift register should contain "000...01" for a 16-bit identifier system. It is used by the routing plane to synchronize the identifier decoding during the routing process.
- 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.

The configuration of the device can be made in a parallel manner, through a 32-bit bus. The 76 configuration bits of a molecule are split into three 32-bit words. Additionally, 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 Fig. 4, 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 microprocessor.

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

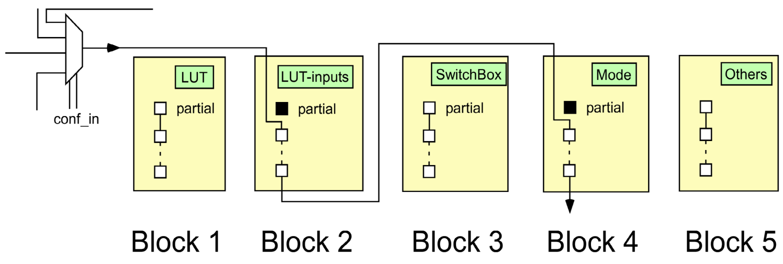

Fig. 4. Organization of the configuration bits for partial reconfiguration

possible to modify multiple molecules at the same time, allowing, for example, the synaptic weights in a neuron to be changed. Moreover, this mechanism permits to use up to 54 of the configuration bits to store information, that can be accessed serially.

2.3 Dynamic Routing

The dynamic routing system is designed to automatically connect the cells' inputs and outputs. Each output of a cell has a unique identifier. 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 Lee's [11], 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.

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 Fig. 5. 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.

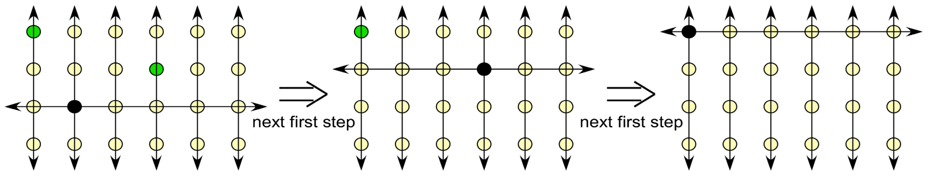

Fig. 5. Three consecutive steps of the routing algorithm. The black routing unit will be the master, and therefore will perform its routing.

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 Fig. 6.

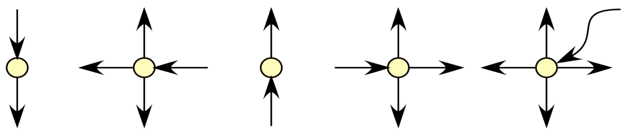

Fig. 6. The propagation direction of the address: north → south | east → south, west, and north | south → north | west → north, east, and south | routing unit → north, east, south, and west

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.

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.

Phase 4: Building the Shortest Path
The last phase, largely inspired by [12], creates a shortest path between the selected source and the selected targets. An example involving 8 sources and 8 targets is shown in Fig. 7, for a densely connected network.

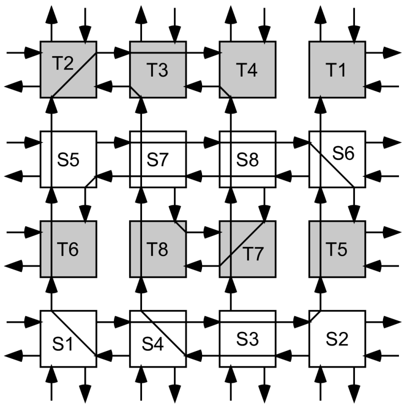

Fig. 7. Test case with a densely connected network

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.

3 Physical Realization

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 Fig. 8, contains 144 molecules organized as an 8x18 array and the complete environment subsystem explained previously. Even if implemented using a standard technology the ASIC implementation of the POEtic tissue demonstrates its superior integration capabilities when compared with those offered by standard prototyping platforms (the prototyping experiments performed within the framework of the project show that an FPGA with 3 million system gates capacity is able to implement the functionality of just 80 POEtic molecules).

Specific development boards have been constructed in order to test the POEtic devices and to implement practical applications on them. These are depicted in Fig. 9. Fig. 9(a) represents the master board, containing one POEtic chip, Flash and SDRAM memory blocks and a USB communication unit that permits to create an interface with an external host. Fig. 9(b) depicts the slave board, containing a 2 x 2 array of POEtic chips. The slave board can be attached to the master board, and it is also possible to connect several slave boards between them in order to create an electronic tissue with the required size for the application to be handled.

Since the complete POEtic tissue has been specified and developed using a standard hardware description language (VHDL) it can be implemented in a standard prototyping platform (though with limited functionality due to the capacity restrictions of current programmable devices). Therefore, in order to facilitate the

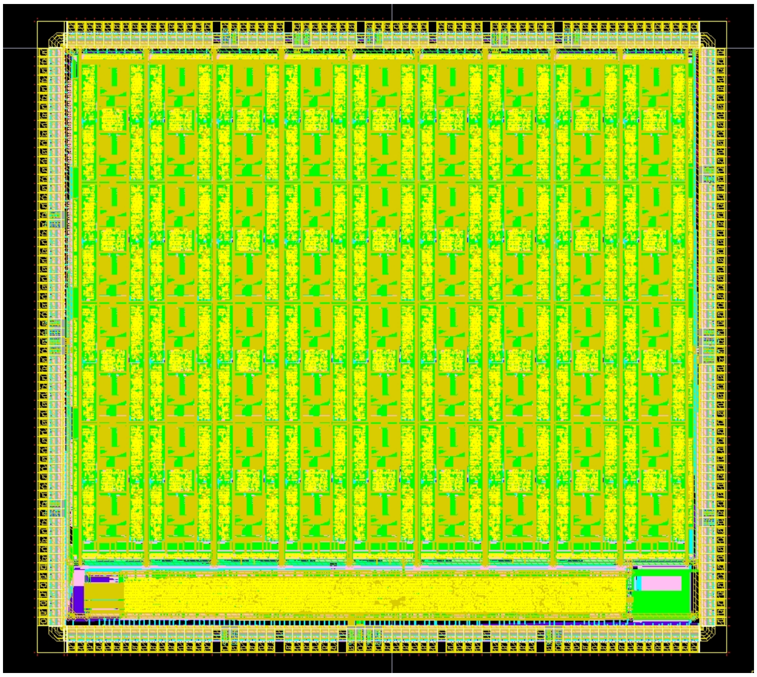

Fig. 8. Layout of the POEtic chip

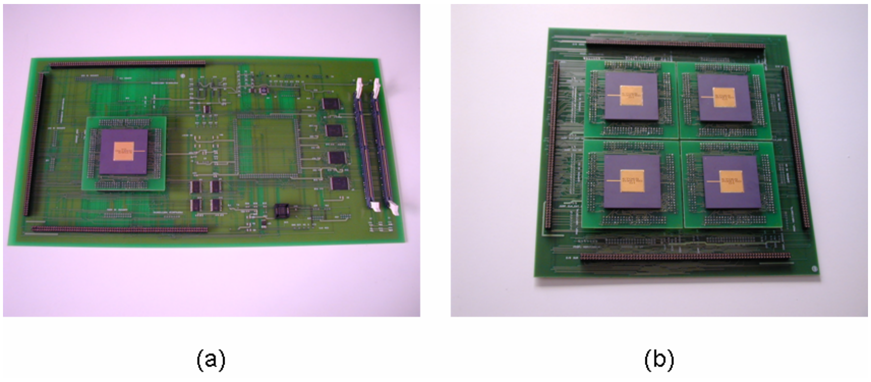

Fig. 9. Details of (a) the master and (b) slave boards developed for the POEtic devices

use of the tissue by external users a complete set of tools have been developed within the framework of the project. This set includes a schematic editor and synthesizer, a molecule-level design entry and simulation tool for the organic subsystem, a C compiler and an assembler integrated in a graphical user interface with language-sensitive editing capabilities, a graphical user interface for the simulation of programs developed for the microprocessor and a system debugger.

4 Conclusions

The POEtic project has produced at its end the first programmable integrated system with capabilities inspired in the organization principles present in living beings: evolution, development/growth, self-replication, self-repair and learning. The resulting electronic device permits to construct a multi-cellular tissue whose size can be adapted to the specific requirements of the application to be handled. The internal architecture of the device includes features, like dynamic partial reconfiguration, self-configuration or in-hardware dynamic routing, that were never combined (if not present at all) in any past electronic device.

In this paper we have presented the architecture that has been conceived for the POEtic devices, as well as the internal organization of its main constituent elements. Then the physical implementation details of the integrated systems and the development boards constructed to create applications based on these devices. The whole system has been described and developed using a standard hardware description language (VHDL). This, together with the set of tools that have been developed for the devices, will permit to test the concepts developed within the project using standard prototyping platforms.

The availability of this brand new family of programmable devices thus opens long term opportunities for the implementation of electronic systems and applications able to take profit of these new features. Among them we could consider the following list:

- Autonomous adaptive systems for deep space exploration.
- Safety critical systems in the aeronautics and the automotive domains.
- Sensor integration for distributed, highly immersive sensor and actuator environments.
- Personalized, user-adaptable assistant systems.
- User-adaptable monitoring and early warning systems for handicapped or elderly people.

Our current work is concentrated in the prototyping of large-scale spiking neural networks models with bio-inspired learning mechanisms using the prototyping platform offered by the POEtic devices.

Acknowledgements

The work presented in this paper has been funded by the grant IST-2000-28027 (POEtic) of the European Community 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. Vinger, K.A., Torresen, J.: Implementing evolution of FIR-filters efficiently in an FPGA. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (2003) 26–29

2. Haddow, P.C., Tufte, G.: Bridging the Genotype-Phenotype Mapping for Digital FPGAs. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (2001) 109-115
3. Sekanina, L: Virtual reconfigurable Circuits for Real-World Applications of Evolvable Hardware. Evolvable Systems: From Biology to hardware. Lecture Notes in Computer Science, Vol. 2606. Springer-Verlag, Berlin Heidelberg New York (2003) 186-197
4. Sekanina, L., Friedl, S.: On Routine Implementation of Virtual Evolvable Devices Using COMBO6. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (2004) 63-70
5. Macias, N.J.: The PIG Paradigm: The design and Use of a Massively Parallel Fine Grained Self-Reconfigurable Infinitely Scalable Architecture. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (1999) 175-180
6. Macias, N.J, Durbeck, L.J.K.: Self-assembling Circuits with Autonomous Fault Handling. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (2002) 46-55
7. Ullmann, M., Hübner, M., Grim, B., Becker, J.: On-Demand FPGA Run-Time System for Dynamical Reconfiguration with Adaptive Priorities. Field Programmable Logic and Applications. Lecture Notes in Computer Science, Vol. 3203. Springer-Verlag, Berlin Heidelberg New York (2003) 454-463
8. Bartosinski, R., Danek, M., Honzik, P., Matousek, R.: Dynamic Reconfiguration in FPGA-based SoC designs. Proceedings of the 2005 ACM/SIGDA 13th international Symposium on Field-programmable gate arrays (2005) 274
9. Tyrrell, A.M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J.M., Rosenberg, J., Villa, A.E.P.: POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware. Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science, Vol. 2606. Springer-Verlag, Berlin Heidelberg New York (2003)129-140
10. ARM. Amba specification, rev 2.0. advanced risc machines ltd (arm). http://www.arm.com/armtech/amba_spec (1999)
11. Lee. C.Y.: An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, Vol EC-10:3 (1961) 346-365
12. Moreno, J.M., Sanchez, E., Cabestany, J: An in-system routing strategy for evolvable hardware programmable platforms. Proceedings of the NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society (1999) 157-166