Ontogenetic Development and Fault Tolerance in the POEtic Tissue

Gianluca Tempesti[1], Daniel Roggen[1], Eduardo Sanchez[1], Yann Thoma[1], Richard Canham[2], and Andy Tyrrell[2]

[1] Swiss Federal Institute of Technology in Lausanne, Switzerland
[2] University of York, England
E-mail: gianluca.tempesti@epfl.ch

Abstract. In this article, we introduce the approach to the realization of ontogenetic development and fault tolerance that will be implemented in the POEtic tissue, a novel reconfigurable digital circuit dedicated to the realization of bio-inspired systems.
The modelization in electronic hardware of the developmental process of multi-cellular biological organisms is an approach that could become extremely useful in the implementation of highly complex systems, where concepts such as self-organization and fault tolerance are key issues.
The concepts presented in this article represent an attempt at finding a useful set of mechanisms to allow the implementation in digital hardware of a bio-inspired developmental process with a reasonable overhead.

1 Introduction

The POEtic tissue is a self-contained digital integrated circuit aimed at the implementation of bio-inspired systems being developed in the framework of a new three-year research project, called "Reconfigurable POEtic Tissue" or "POEtic" for short (see the project's website at http://www.poetictissue.org for more details), recently started under the aegis of the Information Society Technologies (IST) program of the European Community, involving the Swiss Federal Institute of Technology in Lausanne (EPFL, Switzerland), the University of York (England), the Technical University of Catalunya (UPC, Spain), the University of Glasgow (Scotland), and the University of Lausanne (Switzerland).

POEtic tissues are designed to implement a vast range of bio-inspired systems, covering all the major axes of research in the domain (Phylogenesis, Ontogenesis, and Epigenesis) [7, 16, 17, 19, 20]. In this article, we will present the approaches we have selected to realize one of the three biological models that inspire circuit design: ontogenesis, that is, the development and growth of multicellular organisms, along with the fault tolerance that such structures imply.

In the next section, we will introduce the main features and the global architecture of POEtic systems as developed in our project, with particular emphasis on those aspects that are directly involved in the ontogenetic axis. We will then examine in some detail the two main areas of research currently under study for this part of the project, namely development and fault tolerance, and present our solutions for their implementation in silicon.

2 POEtic Hardware

As the general features of POEtic systems have been detailed elsewhere [19, 20], in this section we will concentrate on those aspects of the architecture most relevant to the topics of the article, i.e., ontogenetic development and fault tolerance.

POEtic systems draw inspiration from the multi-cellular structure of complex biological organisms. As in biology, at the heart of our systems is a *hierarchical* structure (Fig. 1) ranging from the molecular to the population levels. In particular, the subjects of this article relate mainly to the cellular and organismic levels, with a brief foray into the molecular level for fault tolerance.

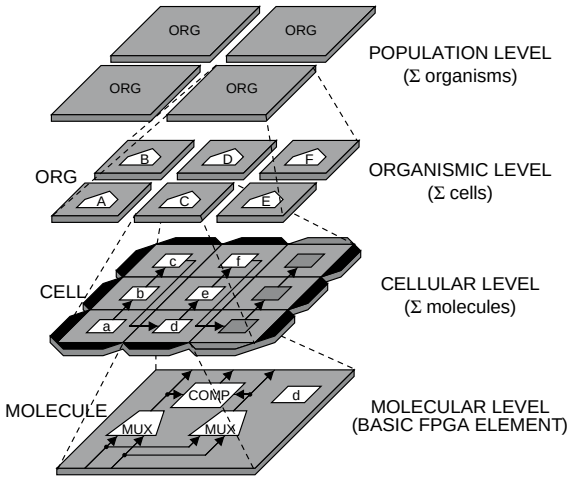

Fig. 1. The organizational layers of an electronic organism.

It is important to note, in this context, that the only hardware (and hence fixed) level of the POEtic systems is the molecular level, implemented as a novel digital FPGA. All other levels are *configurations* for this FPGA, and thus can be seen as *configware*, that is, entities that can be structurally programmed by the user. As a consequence, most of the mechanisms described in this article, and notably ontogenetic development and cellular fault tolerance, can be altered (or removed) to fit a particular application or a novel research approach.

The current (being configware, it could be modified in the future as new solutions are researched) architecture of the POEtic cells is described in Fig. 2. It consists of three logical layers (physically, of course, the three layers are "flattened" onto the molecular FPGA), each with a specific function within the domain of bio-inspired systems. In this article, we shall concentrate on the *mapping layer*, as, together with the *differentiation table*, it is the layer directly concerned by the ontogenetic development of our machines (fault tolerance, as we shall see, is involved in the system at all levels).

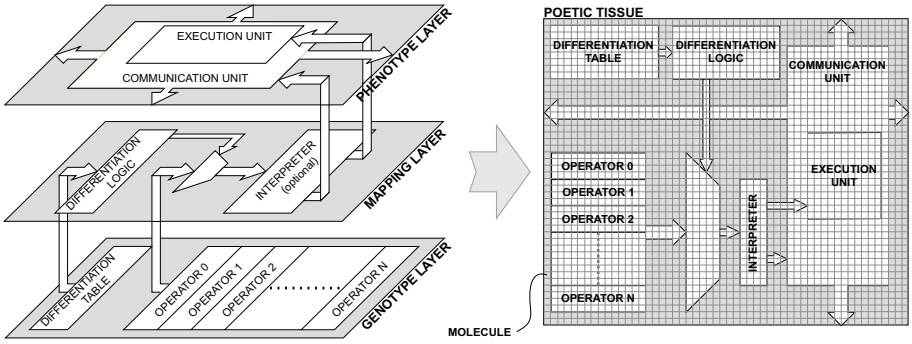

Fig. 2. The three logical layers of an electronic cell, mapped on the molecular tissue. The use of a programmable circuit allows the layers to be adapted to each application.

To illustrate with a practical example the structure of POEtic circuits, we shall use a very simple application: a *timer* (Fig. 3) capable of counting minutes and seconds. This architecture has the twin advantage of being conceptually very simple and of being easily broken down into a cellular structure.

The first step in the design of a POEtic application is in fact to identify a logical division into cells. For this example, the division is obvious: tens of minutes, units of minutes, tens of seconds, units of seconds, which incidentally also correspond to the four outputs of the system (while the only input is the 1Hz synchronization clock). In general, the difficulty of this step depends of course on the application, and is probably the main criterion to determine whether an application is amenable to being implemented in a POEtic circuit.

Next, it is necessary to identify, on the basis of the cellular structure of the application, the operators or functions required by the cells of the system. For the timer, a minimum of two operators is required: a counter by six (for the tens of seconds and minutes) and a counter by ten (for the units of seconds and minutes). In addition, the communication pattern of the cells must also be fixed: for the timer of Fig. 3, the output of the counters is sent to the north, while the synchronization signals are sent out to the west.

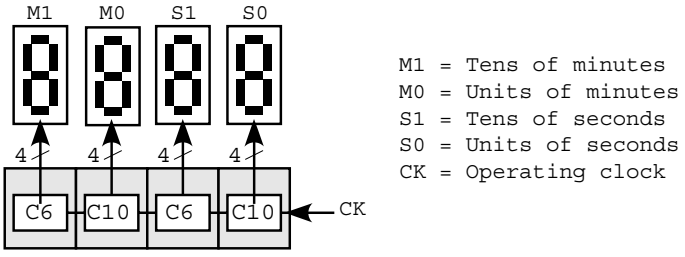

Fig. 3. Cellular implementation of a simple four-digit timer.

At this stage, the architecture of the cells can be defined. This step implies essentially a design choice to decide where to place the complexity of the system. In fact, in a typical example of hardware/software co-design, the complexity can reside in the genome and in its interpreter, Embryonics-style [11], or in the phenotype. For example, in the case of the timer, the first approach would require a genome implemented by a structured program, used to control a simple 4-bit register in the phenotype. Or, in the second approach, the genome could well be reduced to a single number, used as a parameter for a programmable counter in the phenotype layer. Probably a more efficient solution in this case, this second approach would practically eliminate the requirement for an interpreter of the genome, and shift all the (in this case limited) complexity to the phenotype layer.

Of course, the set of operators represents only part of the genetic information of the organism, the rest being made up by the differentiation table used to assign operators to the cells in the system. The definition of a mechanism to implement differentiation in a POEtic system is the subject of the next section.

3 Development in the POEtic Tissue

The modelization in electronic hardware of the developmental process of multi-cellular biological organisms is a challenging task, for reasons that should be obvious: on one hand, the sheer complexity of even the simplest multi-cellular biological organisms is orders of magnitude beyond the current capability of electronic circuits, and on the other hand the development of such organisms implies, if not the creation, at least the transformation of matter in ways that cannot yet be realized in digital or analog hardware.

And yet these same considerations make such a process extremely interesting for hardware design: a developmental approach could well provide a solution, via the concept of self-organization, to the design of electronic circuits of a complexity beyond current layout techniques, and introduce into electronics at least some of the astounding fault tolerance of multi-cellular organisms, while an approximation of the growth process of biological entities could be an invaluable tool to introduce adaptability and versatility to the world of electronics.

Setting aside, for the moment, the concept of fault tolerance, in this section we will describe two different approaches to the twin mechanisms of *cellular division* and *cellular differentiation*, mechanisms at the basis of biological development.

This section will introduce two solutions to the implementation of the second mechanism: cellular differentiation, that is, the mechanism that allows a cell to determine its function within the organism (in other words, the architecture of the mapping layer of our cell). As far as cellular division is concerned, we are currently working on the development of some algorithms allowing our circuit to implement a rough approximation of this mechanism (see the "Implementation Issues" subsection for more details).

3.1 Coordinate-Based Development

Probably the simplest approach to cellular differentiation in an architecture such as that of the POEtic circuits is to assign to each cell in the system a unique coordinate (in fact, for two-dimensional structures, a set of [X,Y] coordinates). The cell can then determine its function by selecting an operator depending on its (unique) coordinates. This is the approach used in the Embryonics project, and has been described in much detail elsewhere [11, 18].

Applied to the timer example (Fig. 4), this approach would require three different kinds of functional cell (count by 6, count by 10, and pass-through), and the timer can be realized by four cells (out of the 15 that could fit in the circuit in this example). The computation of the coordinates is quite simple, implemented by incrementing the coordinate in each cell. It is then trivial to define a differentiation table to assign an operator to each set of coordinates.

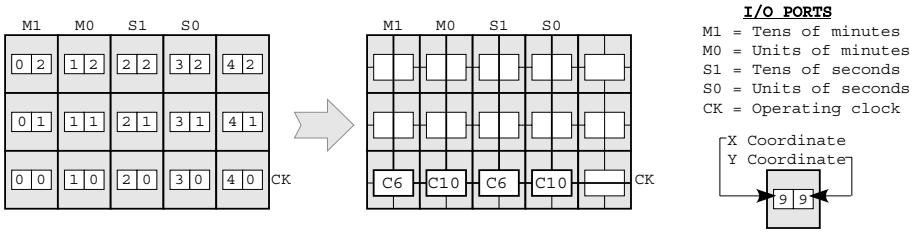

Fig. 4. A possible layout for the timer in a coordinate-based development system. In this system, the position of each cell in the array is determined at design time.

It might be worthwhile to mention that in the Embryonics approach the complexity of the system resides mostly in the genome, implemented as an executable program. As a result of this design choice, the interpreter for the genome is the most complex part of the cell, while the phenotype layer is limited to a single register plus some rudimentary connection network. The POEtic approach, insofar as a coordinate-based development is concerned, will investigate a wide range of tradeoffs between the complexity of the genome and the complexity of the phenotype, as described above for the timer example.

3.2 Gradient-Based Development

The coordinate-based approach described above has a number of advantages, and notably its simplicity on the one hand, and the fact that it has been extensively used and tested in the Embryonics project over many years on the other. In a way, as long as the systems to be implemented are completely pre-determined (i.e., as long as the entire development of the organism is fully specified in the genome, as is the case, for example, of the *caenorhabditis elegans* [15]), it is also the most efficient and complete, and in fact any other approach can be reduced to a coordinate-based approach during development.

However, the growth and development of complex multi-cellular organisms in nature is influenced by the environment from the earliest stages. While a problem from the point of view of an electronic implementation, this feature can lead to extremely interesting systems capable of adapting to environments unknown at design time.

In order to illustrate how such adaptation could be applied to digital circuits, we shall exploit, for clarity's sake, the same example used above for the coordinate-based approach, that is, the timer. It should be noted, however, that such a simplistic example is far from the ideal candidate for an approach designed to address the issue of adaptation: a much more interesting field of application are neural networks, systems that are inherently capable of exploiting rich interactions with the environment.

To come back to the simple example of the 4-digit timer, let us then assume that the circuit has to operate in an *a priori* unknown environment. The environment of a timer is, of course, extremely limited, and could be seen, in the most complex case, as consisting of one input port for the operating frequency of the timer, and of four output ports for the four digits. An unknown environment could then be an environment where the exact position of these input and output ports is not known at design time (once again, this is an extremely unlikely scenario in the design of a timer, but makes much more sense from the point of view of, for example, the control of a robot). A coordinate-based system is not capable of handling such a changing environment, since coordinates are defined as a function of the position of the cells *within* the circuit, independently of the external conditions. A more versatile approach is then required, capable of assigning a function to a cell depending not only on the internal configuration of the circuit, but also of the environment in which the circuit operates. Again drawing inspiration from biology, our choice has fallen on an approach based on the protein gradients that direct the development of organisms in nature.

The concept of protein gradients has been examined in detail in many publications, for biological [4, 14] as well as electronic [5, 6, 9] organisms. Essentially, a gradient-based system consists of a set of *diffusers* that release a given protein into the system. The concentration of the protein in the system is highest at the diffuser, and decreases with the distance. Cells are assigned a functionality depending on the proteins' concentration, and hence on the distance from the proteins' diffusers, implementing a cellular differentiation based on their position *relative* to the diffusers rather than on their coordinates within the system.

For the timer example, we can consider that each of the five I/O ports mentioned above is a diffuser for a different kind of protein (again, we chose this solution for the sake of clarity, as in reality there is a tradeoff between the number of different proteins and the complexity of the mapping circuit). It is then relatively simple to design a cell that selects its function depending on the concentration of the appropriate protein. For example (Fig. 5), a simple algorithm could place the timer so that the rightmost cell is contiguous to the input port for the operating frequency and use the other cells in the circuit to create paths from the four cells to the four output ports, following the corresponding gradients.

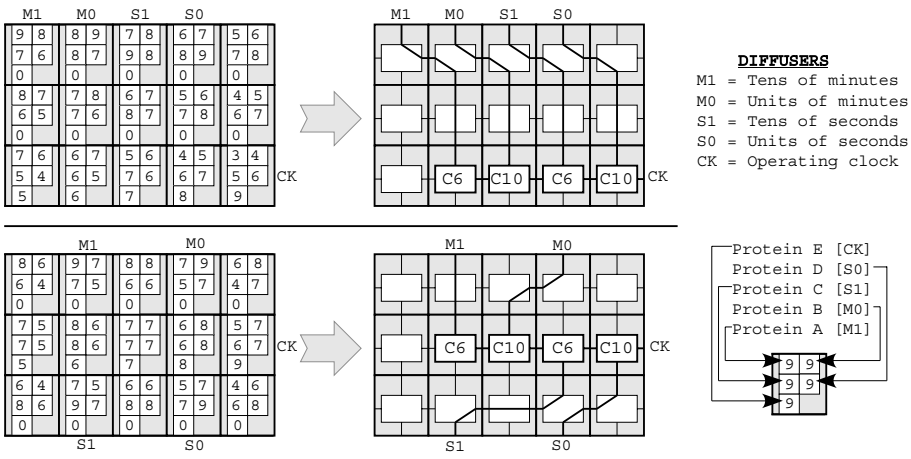

Fig. 5. Two possible layouts for the timer in a gradient-based development system. The position of each cell in the array is determined by the system during configuration.

Even in such a simplistic example, the versatility of a gradient-based system should be obvious. This kind of approach can also exploit the other axes of bio-inspiration to increase the adaptability of multi-cellular systems (aside from the obvious interest for adaptive systems such as neural networks, evolution can be used, for example, to define the position within the system of some emitters, allowing the genome to partially direct the development of the organism).

3.3 Implementation Issues

We are actively pursuing research on both of the developmental approaches described above. As we mentioned, for systems that can be completely defined in the genome at design time, a coordinate-based system would be more efficient. On the other hand, for adaptive systems where the environments influences development, a gradient-based approach provides a much more versatile tool.

In parallel, we are also investigating more complex, richer mappings between the genotype and the phenotype. The Embryonics approach, for example, requires a rigid one-to-one mapping from coordinates to function. A much more interesting approach exploits the possibility of using partially-defined differentiation tables that are accessed not through a one-to-one mapping, but rather through more complex algorithms (e.g., by computing the Hamming distance between the coordinates or the protein concentrations and the table entries).

The use of partially-defined differentiation tables, coupled with the possibility of environment-directed development (e.g., through protein diffusers), also opens the possibility of mechanisms that approximate the growth process in biological organisms. If the organism's structure is only partially defined in the genome, and if the environment is used to define the cells' function, then an informational (rather than physical, as in nature) growth process can be realized.

4 Fault Tolerance in the POEtic Tissue

Even if the complexity of digital hardware remains orders of magnitude behind that of the more developed biological organisms, some of the issues that nature had to confront during the millions of years required for the evolution of such organisms are gaining interest for electronics. For example, as we mentioned earlier, silicon might well be replaced, in a not-so-distant future,by molecular-size components with densities beyond what could be handled through current layout techniques. Among the many challenges represented by such novel hardware, one in particular stands out: these incredibly small components will never be "perfect". In other words, circuits will inevitably contain faulty elements.

Nature, faced with this problem, has developed extremely efficient solutions. Fault tolerance is definitely one domain where nature is vastly more advanced that electronics, and one of our goals is to draw inspiration from biological healing and repair mechanisms to achieve stronger fault tolerance in digital hardware.

From a "practical" standpoint, it is important to note that fault tolerance implies in fact two different processes: self-test, to detect that a fault has occurred within the circuit, and self-repair, to somehow allow the circuit to keep operating in the presence of one or more faults. The two processes are in fact very distinct, both conceptually and in practice. In particular, self-test implies that the circuit be capable of detecting incorrect operation and, since the fault will eventually have to be neutralized, the exact site where the incorrect operation occurs. Self-repair, on the other hand, implies that all parts of the circuit must be replaceable, physically or at least logically, via some sort of reconfiguration.

There exist, for both processes, many "conventional" approaches [1, 10, 12], but no really efficient off-the-shelf solution. Drawing once again inspiration from nature, where "fault tolerance" is achieved through not one, but a set of mechanisms ranging from molecular repair to complex organism-level immune systems, we are developing a *hierarchical* approach to fault tolerance, spanning all the levels (Fig. 1) of our POEtic systems.

4.1 Molecular-Level Fault Tolerance

Molecular-level fault tolerance is probably the most sensitive area of our hierarchical approach, as the molecules (and hence their test and repair mechanisms) represent the hardware of our system, and thus will necessarily have to be fixed at design time (rather than being modifiable by configware as the other layers).

However, the hardware layer of the POEtic tissue has not yet been fully designed (it represents the main research effort under way at the moment within the project) and, by their nature, logic level test and repair mechanisms require a completed layout to be efficiently implemented.

In general, the fault tolerance mechanisms implemented for the POEtic tissue will bear a loose resemblance to those used for Embryonics designs [11, 18], at least in that they will rely on a hybrid approach mixing mechanisms such as duplication (a mechanism found in nature in the DNA's double helix and in its error-correcting processes) and memory testing [10].

4.2 Cellular-Level Fault Tolerance

The coarser grain of cells with respect to molecules can be exploited to introduce more complex test and repair patterns. Fault tolerance at the processor level (cells can be viewed, for testing purposes, as small processing units) is usually implemented through techniques radically different from the logic level, and more complex reconfiguration patterns can be realized by exploiting the development approaches implemented in the POEtic circuits.

For example, testing at the processor level can take advantage of the presence of multi-bit busses to test the correct operation of the machine through techniques such as parity-checking. As far as self-repair is concerned, reconfiguration mechanisms can exploit the features of the POEtic architecture to simplify the rearrangement of tasks within the array: the presence of *the full genome in each cell*, coupled with the developmental mechanism that assigns the function to the cells depending on local conditions, can for example allow a relatively simple implementation of on-line self-repair via reconfiguration (Fig. 6).

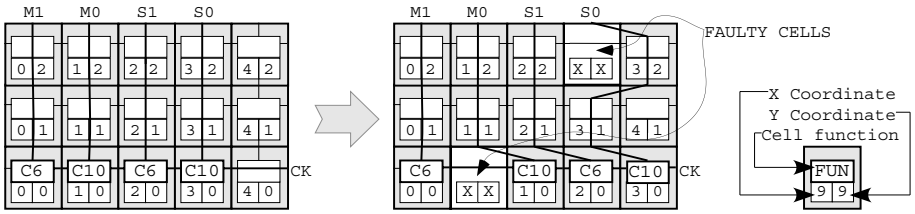

Fig. 6. Self-repair through reconfiguration in a coordinate-based system. As each cell is totipotent, reconfiguration consists simply of a re-computation of the cells' coordinates.

Of course, it should also be noted that many of the systems that POEtic architectures are meant for, and notably neural networks, are *inherently* fault tolerant at the cellular level: the death of a neuron within the system will force the learning algorithms to automatically avoid the faulty neuron and to find a solution that can perform the desired computational task in a reduced network.

4.3 Organismic-Level Fault Tolerance

A simplistic vision of organismic-level fault tolerance consists of exploiting the developmental approach of POEtic systems to create multiple redundant copies of an organism during configuration. This approach, used in the Embryonics systems [11], introduces improved fault tolerance through a simple comparison of the outputs of multiple identical circuits (the biological analogy would reside in the survival of a population even if individuals die).

This form of fault-tolerance, while relatively simple to realize (through cycles in the coordinates, as in Fig. 7, or by using multiple diffusers of the same kind

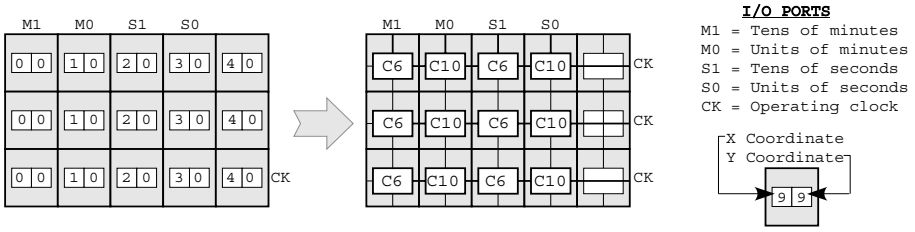

Fig. 7. Multiple redundant copies of an organism through coordinate cycling.

in a gradient-based system), requires material *outside* of the POEtic circuits to identify (and possibly kill) faulty organisms.

A more interesting approach (and one of the research axes of the POEtic project) tries to exploit mechanisms inspired by biological immune systems [2, 3]. In higher organisms, these systems rely on a multi-layered, distributed approach that is robust and capable of identifying numerous pathogens and other causes of illness, and is undoubtedly an interesting source of inspiration for the development of reliability mechanisms in silicon.

The approach relies on a negative selection algorithm [8] coupled with a more "conventional" mechanism that could be based on roving self-test areas (STAR) [1]. In practice, the approach merges a fixed testing mechanism (the *innate* part of the immune system) with a *learning* negative selection mechanism (the *acquired* part), working at a system or subsystem level to monitor the state of the circuit and identify and kill cells that develop faults at runtime.

4.4 Implementation Issues

The final goal of the mechanisms we introduced is to illustrate how a hierarchical approach to fault tolerance, along the same general lines as the one present in complex biological organisms, is a very efficient solution from the point of view of assuring the correct operation of the circuit in the presence of faults. By adopting a hierarchy of mechanisms, we can exploit the best features of each level of our systems, and thus limit the overhead required to obtain an "acceptable" level of reliability for the system as a whole.

In reality, the only relevant overhead generated by introducing fault tolerance in our POEtic circuits resides in the molecular level: since all other levels are implemented in configware, reliability (and its associated overhead) becomes an option, and can be removed should fault tolerance not be a priority.

The important implementation issues for fault tolerance thus concern essentially the test and repair mechanisms at the molecular level. As we mentioned, however, the design of the molecule is currently under way, and all the important parameters required to introduce reliability to the molecular FPGA (connection network, degree of homogeneity, configuration mechanism, etc.) have yet to be fixed. A complete description of the implementation of fault tolerance in the POEtic tissue will thus be the subject of a future article.

5 Conclusions

The concepts presented in this article represent an attempt at finding a useful set of mechanisms to allow the implementation in digital hardware of a bio-inspired developmental process with a reasonable overhead. The modelization of multi-cellular organisms in silicon is an approach that could soon become extremely useful for the realization of highly complex systems, where concepts such as self-organization and fault tolerance are becoming key issues.

Much of our effort is currently focused on finding an adequate approach to implement development in the POEtic tissue. We are examining processes that are at the same time efficient from the point of view of the required hardware resources and scalable across one or even multiple chips. Notably, we hope to develop a mechanism versatile enough to allow on-line addition of new chips (that is, the user will be able to add new hardware to the system without disrupting its operation), thus achieving something closer to the *physical* growth of biological organisms.

The two developmental models described in this article fit remarkably well our requirements (scalability, robustness, efficiency, etc.), and complement each other remarkably well: the coordinate-based approach is most efficient for "conventional" systems where the layout of the circuit is known in advance, while the gradient-based approach provides an increased versatility invaluable for the realization of adaptive systems such as neural networks. Moreover, the architecture of our POEtic machines places the developmental mechanism where it can be modified by configware, allowing in the future the implementation of different approaches.

It is definitely too soon to draw any conclusions on the topic of fault tolerance in the POEtic tissue: until the hardware layer is fixed, the application of self-test and self-repair techniques is impossible. One aspect that is already apparent, however, is that our research is focusing on a *hierarchical* set of mechanisms meant to work together to achieve a level of reliability that would not be possible for a single mechanism.

Finally, we wish to reiterate the *reconfigurable* aspects of the architecture we described: the molecular tissue is meant to be a *universal* tissue for the implementation of bio-inspired systems. The architectures and mechanisms described in this article represent only some of the ideas that will be pursued in the POEtic project. The circuit, once developed, will be made publicly available and should prove a useful tool for research in the domain beyond the participants to the project.

Acknowledgments. This project is funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, under grant IST-2000-28027 (POETIC). 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. The Swiss participants to this project are supported under grant 00.0529-1 by the Swiss government.

References

1. Abramovici, M., Stroud, C.: Complete Testing and Diagnosis of FPGA Logic Blocks. IEEE Trans. on VLSI Systems **9:1** (2001).
2. Bradley, D.W., Tyrrell, A.: Multi-Layered Defence Mechanisms: Architecture, Implementation, and Demonstration of a Hardware Immune System. Proc. 4th Int. Conf. on Evolvable Systems, LNCS **2210** (2001), 140–150.
3. Canham, R.O., Tyrrell, A.M.: A Multilayered Immune System for Hardware Fault Tolerance within an Embryonic Array. Proc. 1st Int. Conf. on Artificial Immune Systems (ICARIS 2002), Canterbury, UK, September 2002.
4. Coen, E.: The Art of Genes. Oxford University Press (1999), New York.
5. Edwards, R.T.: Circuit Morphologies and Ontogenies. Proc. 2002 NASA/DoD Conf. on Evolvable Hardware, IEEE Computer Society Press (2002), 251–260.
6. Eggenberger, P.: Cell Interactions as a Control Tool of Developmental Processes for Evolutionary Robotics. From Animals to Animats 4: Proc. 4th Int. Conf. on Simulation of Adaptive Behavior, MIT Press - Bradford Books (1996), 440–448.
7. Eriksson, J., Torres, O., Mitchell, A., Tucker, G., Lindsay, K., Halliday, D., Rosenberg, J., Moreno, J.-M., Villa, A.E.P.: Spiking Neural Networks for Reconfigurable POEtic Tissue. Elsewhere in this volume.
8. Forrest, A., Perelson, A.S., Allen, L., Cherukuri, R.: Self-Nonself Discrimination in a Computer. Proc. 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1994).
9. Gordon, T.G.W., Bentley, P.J.: Towards Development in Evolvable Hardware. Proc. 2002 NASA/DoD Conf. on Evolvable Hardware, IEEE Computer Society Press (2002), 241–250.
10. Lala, P.K.: Digital Circuit Testing and Testability. Academic Press (1997).
11. Mange, D., Sipper, M., Stauffer, A., Tempesti, G.: Towards Robust Integrated Circuits: The Embryonics Approach. Proc. of the IEEE **88:4** (2000), 516–541.
12. Negrini, R., Sami, M.G., Stefanelli, R.: Fault Tolerance through Reconfiguration in VLSI and WSI Arrays. The MIT Press, Cambridge, MA (1989).
13. Ortega, C., Tyrrell, A.: MUXTREE revisited: Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays. LNCS **1478**, Springer-Verlag, Berlin (1998), 206–217.
14. Raven, P., Johnson, G.: Biology. McGraw-Hill (2001), 6th edition.
15. Riddle, D.L., Blumenthal, T., Meyer, B.J., Priess, J.R., eds.: C. Elegans II. Cold Spring Harbor Laboratory Press (1997).
16. Roggen, D., Floreano, D., Mattiussi, C.: A Morphogenetic System as the Phylogenetic Mechanism of the POEtic Tissue. Elsewhere in this volume.
17. Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Pérez-Uribe, A., and Stauffer, A.: A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems. IEEE Transactions on Evolutionary Computation, **1:1** (1997) 83–97.
18. Tempesti, G.: A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes. Ph.D. Thesis No. **1827** (1998), EPFL, Lausanne, Switzerland.
19. Tempesti, G., Roggen, D., Sanchez, E., Thoma, Y., Canham, R., Tyrrell, A., Moreno, J.-M.: A POEtic Architecture for Bio-Inspired Systems. Proc. 8th Int. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life VIII), Sydney, Australia, December 2002.
20. Tyrrell, A.M., Sanchez, E., Floreano, F., Tempesti, G., Mange, D., Moreno, J.-M., Rosenberg, J., Villa, A.E.P.: POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware. Elsewhere in this volume.