In Y. Liu, K. Tanaka, M. Iwata, T. Higuchi, M. Yasunaga (eds.)
Proc. 4th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES2001), Tokyo, Japan, pp. 100-111.
LNCS 2210, Springer-Verlag, Berlin, 2001.

Embryonics: Artificial Cells Driven by Artificial DNA

Lucian Prodan[1], Gianluca Tempesti[2], Daniel Mange[2], and André Stauffer[2]

[1] "Politehnica" University (UPT), Timisoara, Romania
E-mail: lprodan@cs.utt.ro, WWW:http://www.cs.utt.ro

[2] Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
E-mail: name.surname@epfl.ch, WWW:http://lslwww.epfl.ch

Abstract. Embryonics is a long-term research project attempting to draw inspiration from the biological process of ontogeny, to implement novel digital computing machines endowed with better fault-tolerant capabilities. For this purpose FPGAs are extremely useful. However, through this project we designed MuxTree, a new coarse-grained FPGA, to implement our embryonic machines. This article focuses on the issues posed by the memory storage and the advances made to achieve more robust memory structures.

Motto. "The nature of an identity lies in its essence." Aristotle

1 Introduction

Present days computing systems are designed from the very beginning to face some challenging problems. One of the main issues is testability, or how to be able to verify that a system functions up to its specs and another is fault tolerance, or how to make the system continue to function properly even while faults are occurring. Though there are techniques developed to satisfy both constraints, engineers have found another source of inspiration, and closer than they thought. The answer could lie in adopting mechanisms tested and refined by nature ever since life began on Earth: bio-inspiration.

A human being is made up of some 60 trillion ($60x10^{12}$) cells. Key for the survival of the organism is the relentless decoding of the *genome*, a ribbon of 2 billion characters, to produce the necessary proteins [14]. The parallel execution of 60 trillion genomes in as many cells occurs ceaselessly from the conception to the death of the individual. At any given moment, many protecting mechanisms keep an eye on the well operating of the whole organism. Eventual faults, though rare, are in the majority of cases, successfully spotted and repaired. The inspiration of the Embryonics (*embryonic electronics*) project [3, 4, 10, 11] is this astounding degree of parallelism present in nature. Embryonics tries to adapt some of the development processes of multicellular organisms to the purpose of designing novel, robust architectures for massive parallelism in silicon.

It is biology that made possible the miracle of contemplating the successfully operating human organism. It is a miracle indeed to have trillions of cells operating in

parallel, forming intricate structures (tissues and organs) only to perform a single goal, that is, the living organism. These are the astounding biological features that make engineers think about a new way of designing novel electronic systems. A bio-inspired computing system would – theoretically – be capable of online self-testing and self-repairing. The article is structured as follows: Section 2 presents the path of Embryonics from its very beginning, Section 3 and 4 introduce the reader into some of the more peculiar of its features, while the focus is on Section 5 where details of improving the fault tolerance of the memories employed by Embryonics are presented. Finally, Section 6 presents the conclusions and some general guidelines for the future of the project.

2 Overview

2.1 Toward Bio-Inspiration

The transition form carbon-based organisms to silicon-based electronic circuits is, of course, far from immediate. Living beings exploit intricate processes, many of which remain undiscovered or unexplained. Therefore, the Embryonics project focuses on two goals [14]:

- *Similarity*: where possible, to develop digital circuits exploit processes similar (but obviously not identical) to those used by living organisms
- *Effectiveness*: while inspired by biology, the systems we design must remain useful and efficient from an engineer's standpoint.

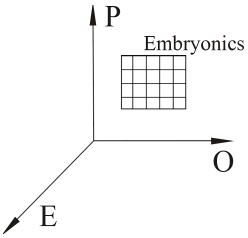

Fig. 1. The POE model and the current position of the Embryonics project.

Therefore the Embryonics project does not try to *imitate* life, but rather to extract some useful ideas from some the most fundamental mechanisms of living creatures.

2.2 The Embryonics Project

Bio-inspired computing systems are categorized by Sipper et al. [13] using their proposed *POE* model, which makes use of three orthogonal axes. Called *phylogenetic*

(P), *ontogenetic* (O) and *epigenetic* (E), they define the space inside which all bio-inspired systems, both software and hardware, are situated.

The phylogenetic axis represents the evolution of the genetic program, the reproduction of all living organisms being based on to. The phylogenetic processes exhibit a very low error rate at the individual level and are fundamentally nondeterministic, the source of diversity being mutation and sexual reproduction. Systems represented along the phylogenetic axis are known as *evolvable hardware* or *evolware*.

The epigenetic axis involves online learning through interaction of the systems with the surrounding environment. To the best of our knowledge, only three epigenetic systems exist in living beings: the immune, the nervous, and the endocrine systems. Represented along the epigenetic axis are the bio-inspired systems that are capable of learning, usually under the form of artificial neural networks.

The ontogenetic axis focuses on the development of the single individual from its very own genetic material. Environmentally induced behavior is not considered, thus the main process off the ontogenetic axis being the *growth* of the organism. Characteristics such as replication (self-replication), which can be seen as a special case of growth, and regeneration (self-repair), or recovery after wounds or illnesses, are part of the ontogeny and are extremely attractive for many applications. The Embryonics project presents a consistent view of ontogenetic [2, 12] hardware but it is not necessarily limited to ontogenetic processes. For the moment it can be regarded as situated in the plane defined by the ontogenetic axis and the phylogenetic axis (Fig. 1).

2.3 From Biology to Electronics: Bridging the Gap

With the exception of unicellular organisms (such as bacteria), living beings share three fundamental features [14]:

- *Multicellular organization* divides the organism into a finite number of *cells*, each realizing a unique function.
- *Cellular division* is the process whereby each cell (beginning with the first cell or *zygote*) generates one or two daughter cells. During this division, all of the genetic material of the mother cell, the *genome*, is copied into the daughter cell(s).
- *Cellular differentiation* defines the role of each cell of the organism, that is, its particular function (neuron, muscle, intestine, etc.). This specialization of the cell is obtained through the expression of part of the genome, consisting of one or more *genes*, and depends essentially on the physical position of the cell in the organism.

As each cell contains the genome, that is the whole of the organism's genetic material, the cell's "universality" comes as a consequence. This makes the living organisms capable of self-repair (regeneration, cicatrisation) or self-replication (cloning or budding). These two properties, based on a multicellular tissue, are essentially unique to the living world.

Obviously, the differences between the worlds of biology and of electronics are far too many, preventing us from simply copying the nature in silicon. One difference

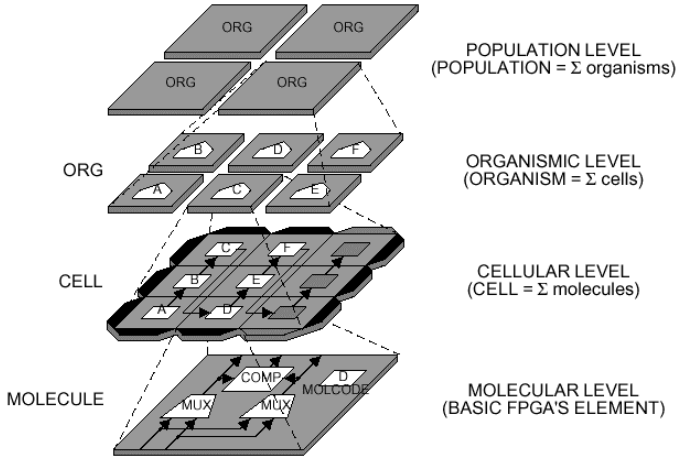

Fig. 2. Embryonics: the 4 levels of organization.

that is not addressable by present days engineering is that the environment living beings interact with is continuously changing, whereas the environment in which our quasi-biological development occurs is imposed by the structure of the electronic circuits, consisting of a finite (but arbitrarily large) two-dimensional surface of silicon and metal. Taking into account the extant differences, we developed a quasi-biological system architecture based on four levels of organization (Fig. 2), described in detail in previous articles [3, 9, 10]. The particular subject of this article lies at the *molecular level*, the bottom layer of our system, and concerns the implementation of fault tolerant memory structures. We will therefore introduce the other levels and discuss them only insofar as they are useful for a clearer understanding of our subject matter.

3 The Two Level Organization

3.1 The Cellular Level

As shown in Fig. 3, our artificial organisms are divided into a finite number of cells. Each cell is a simple processor (a binary decision machine), which realizes a unique function within the organism, defined by a set of instructions (program), which we will call the *gene* of the cell. The functionality of the organism is therefore obtained by the parallel operation of all the cells.

Cells are delimited by the existence of a cellular *membrane*, which is in fact an automaton receiving its configuration at initialization time. The information specifying the cellular membrane is known as *polymerase genome*, and it is part of the genome. The dimensions of the cells are programmable, and the mechanism specifying the cellular membrane is called *space divider*. The space divider extends its operations also in the next, more basic level.

Each cell stores a copy of all the genes of the organism (which, together, represent the operative part of our artificial genome, the *operative genome*), and determines which gene to execute depending on its position (X and Y coordinates) within the organism, implementing *cellular differentiation*. In Fig. 3 each cell of a 6-cell organism realizes one of the six possible genes (A to F), but stores a copy of all the genes.

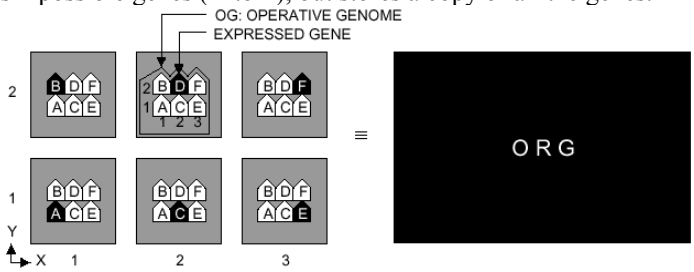

Fig. 3. The multi-cellular organization of an organism.

3.2 The Molecular Level

As seen previously, our artificial organisms decompose into cells, which at their turn decompose into molecules. The reasons for such implementation of our artificial cells are detailed elsewhere [3, 9, 10]. As a programmable substrate of logic, the molecular level is very suited to be implemented using FPGAs. In our case, Embryonics relies on a new type of FPGA, called MuxTree (standing for tree of multiplexers). The molecule is essentially a multiplexer and a D-type flip-flop, linked with the other molecules via a set of programmable connections (Fig. 4). The bit stream configuring the molecule (that is, the connections and the preset value of the flip-flop) is stored into the configuration register CREG.

The unit containing the flip-flop is called the *functional unit* FU, and it provides on-line self-testing and self-repairing. There exist 3 copies of the flip-flop and a simple 2-out-of-3 majority mechanism ensures the fault tolerant operating of the unit.

The switch block SB drives the connections between molecules. Since implementing fault tolerant techniques into the SB would greatly increase the size of the actual implementation of the design, this unit does not provide any such techniques.

4 Memory Structures

Our molecule was initially designed to be able to implement any combinational and/or sequential machines, provided that a sufficient number of molecules were available. But one of the issues raised by our cellular processors was the implementation of the memory required to store the genome program. While capable of fulfilling this role, conventional addressable memory systems present the handicap of requiring relatively complex addressing and decoding logic. Embryonics also had as a disadvantage the

fact that one molecule could only store one bit with its functional unit. To store the operative genome we therefore took into consideration other methods.

In living cells, the genetic information is processed *sequentially*. Designing a memory that is inspired by biology suggests a different type of memory, which we called *cyclic memory*. Cyclic memory does not require any addressing mechanism. Instead, it consists of a simple storage structure that circulates synchronously its data in a closed circle, much as the ribosome processes the genome inside a living cell [1].

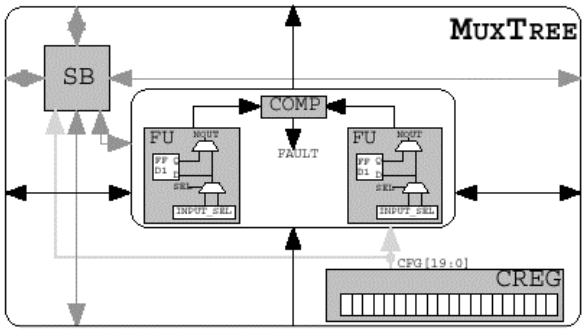

Fig. 4. Internal architecture of the artificial molecule.

The CREG's extended role meant that molecules could operate in two modes, set by special bits in CREG [14]. The first is the *active mode*, in which the molecule acts in its initial way: it makes use of the FU and the SB. The second is the *memory mode*, in which the molecule has two operating sub-modes: short memory (8 bits of CREG are used for data storage) with only SB being active and long memory (16 bits of CREG are used to store data) where neither SB nor FU are active. This was the chosen way to enlarge the storage capabilities of one molecule in order to implement our genome memory.

The molecules operating in either memory modes form rectangular memory structures, the smallest possible such structure being a column composed of only two molecules. There are data output ports at each molecule situated in the top row of the structure. Similar to the way cells are separated from each other by the cellular membrane, we implemented a programmable memory membrane that separates different memory structures. This is to say in the same cell there can exist more than one memory structure, with different shapes and sizes.

To describe the way our cyclic memory operates is not the purpose of this article, it being described in detail in [14]. Until we opted for the cyclic memory architecture the sole purpose of the configuration register CREG was to store the molecule's configuration. For this, an off-line self-testing technique at initialization time was employed. However, adapting the cyclic memory extended its features and there is need to also extend its fault tolerance to suit them, aspects discussed in the next Section.

5 Reliability and Fault Tolerance

The very essence of the Embryonics project is to deliver unprecedented reliability through massive fault-tolerance achieved by bio-inspired design. As difficult as reaching this goal might seem for us, engineers, nature found solutions to fault-tolerance and perfected them throughout hundreds of millions of years. The choice of trying to draw the best of its advantages into the world of silicon seems to be at least worth trying.

Biological entities live continuously under environmental stress. Wounds and illnesses resulting from such stress often cause incapacitating physical modifications. Fortunately, living beings are capable of successfully fighting the great majority of such wounds and illnesses, showing a remarkable robustness through a process that we call *healing*. To reach similar features, a two-level mechanism for self-repair, involving both the cellular and the molecular level is provided in Embryonics. What follows is a description of healing at the cellular and molecular level and of the way the two levels cooperate to produce a higher level of robustness than would be allowed by a single level.

5.1 Self-Repair at the Cellular Level

The redundant storage of the entire genome in every cell is obviously expensive in terms of additional memory. However, it has the important advantage of making the cell *universal*, that is, potentially capable of executing any one of the functions required by the organism. This property is a huge advantage for implementing *self-repair*, the electronic equivalent of biological healing.

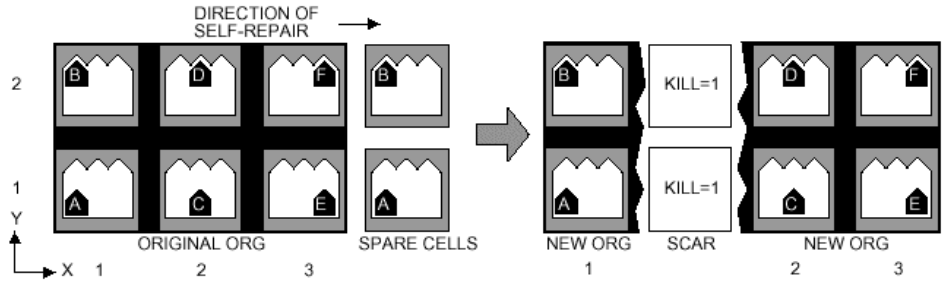

Fig. 5. Self-repair at the cellular level.

Since our cells are universal, the system can survive the "death" of any one cell simply by re-computing the cells' coordinates within the array, provided of course that "spare" cells (i.e., cells which are not necessary for the organism, but are held in reserve during normal operation) are available (Fig. 5).

Self-repair at the cellular level thus consists simply of deactivating the column containing the faulty cell: all the cells in the column "disappear" from the array, that is, become transparent with respect to all horizontal signals. More details about cellular self-repair are provided elsewhere [6, 10, 14].

5.2 Self-Repair at the Molecular Level

Killing a column of processors for every fault in the array represents a penalty we wished to avoid and therefore a certain degree of fault tolerance at the molecular level was introduced. Self-repair in an FPGA implies two separate processes: self-test and reconfiguration. Of these two processes, self-test is undoubtedly the costliest, and we adopted a relatively complex hybrid solution mixing duplication and fixed-pattern testing [9] too complicated to be described here.

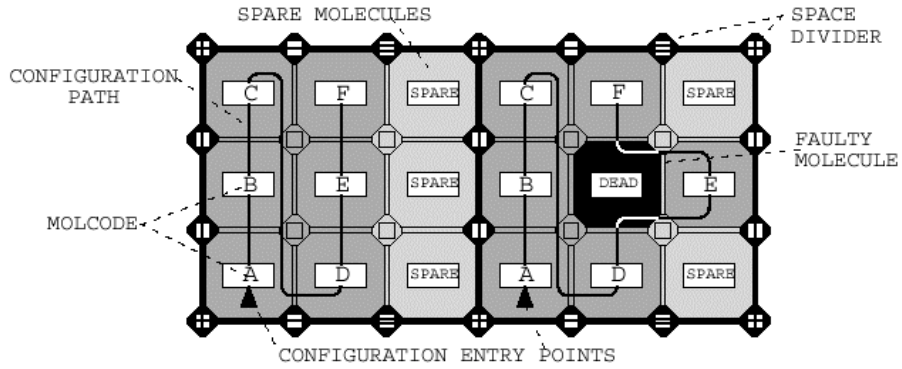

Fig. 6. Self-repair at the molecular level.

On the other hand, the homogeneous architecture of our FPGA simplifies reconfiguration to a considerable extent [5, 7]. Since all molecules are identical, and the connection network is homogeneously distributed throughout the array, reconfiguration becomes a simple question of shifting the configuration of the faulty molecule to its right (similarly to what happens during configuration, as shown in Fig. 6) and redirecting the array's connections. This procedure can be accomplished quite easily, assuming that a set of spare molecules is available. The determination of these spare molecules is in fact one of the most powerful features of our system, since we can exploit the space divider to dynamically allocate some columns as spares. The position and frequency of spares can then be determined at configuration time, and the fault tolerance of our FPGA becomes programmable (and can thus be adapted to the circumstances and the operating conditions).

5.3 Self-Testing Memory Structures

The analysis of Ortega et al. [6] proves that the above strategies of self-repair provide quite a robust architecture. However, when memory structures are employed, the off-line self-testing features implemented in the configuration register (CREG), which is the main memory unit, is insufficient for robust structures of this type. The technique employed to test the CREG was sufficient only for molecules operating in active mode, when the data stored by the CREG remains the same at any given moment. But when molecules are in one of the memory modes, the CREG continuously shifts its

data, thus being subject to failure. Therefore the implemented off-line self-testing technique, which takes place at initialization time only does not cover memory data.

The biological DNA, with its two twisted helices (Fig. 7B), provides an intricate means of storing information. As strange as it might seem, information encoded in the DNA takes advantage of techniques used in the designing of fault-tolerant systems: the information is redundant, it is coded in a digital way and it has error control implemented. Each of the two DNA strands is composed of genes, that are strings made of sequences of one of the four bases: A (adenine), T (thymine), C (cytosine) and G (guanine). This is to say that DNA information is stored based on only four characters – the four bases – thus proving the discrete manner of DNA encoding. The error control is achieved by the existence of the second strand, which can be considered as the *complementary* form of the first one. This is because the two strands are in fact linked, and links can only be established between bases A and T, or C and G. Actually, the robustness achieved is so remarkable, DNA can keep its information unaltered in spite of UV radiations, EM fields and other natural stress.

It was discovered that cells have a variety of DNA polymerase enzymes that serve for DNA repair [15]. Since any damage to the DNA would be lethal, biological cells often spend much more energy repairing the DNA than synthesizing it. The correcting process of DNA damage due to environmental effects or proofreading during replication is, unsurprisingly, quite complicated; it assumes the detection of such errors, cutting them out, and then using the remaining good strand as a template for repair synthesis. So even if one strand becomes erroneous, it will be repaired based on the complementary information provided by the second strand.

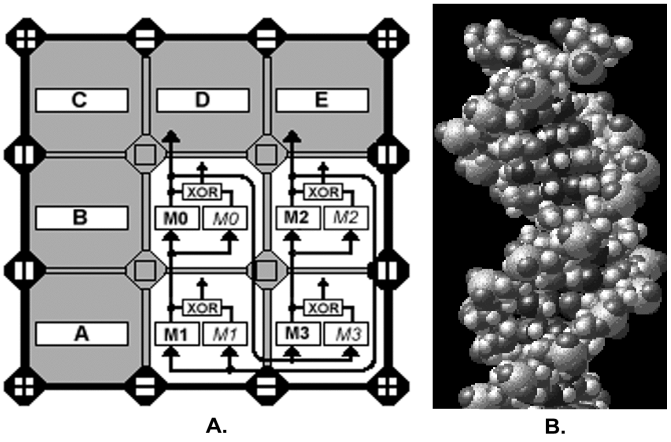

Fig. 7. The new memory implementation (A) and the DNA (B).

After all biological features described above, it would seem that Embryonics could also draw some advantages for implementing an error control inside memory structures. But designing a memory that would exhibit on-line self-testing and self-repairing is a difficult and very expensive task. From our standpoint, using even a simple 2-out-of-3 majority function at the configuration register CREG level would almost triple the amount of logic needed for a molecule.

Instead, we could use the two halves of the configuration register in a similar way DNA uses two complementary strands. In memory mode, one molecule offers a maximum of 16 bits of storage space (the *long memory mode*). It is possible to provide fault detection by splitting the CREG in two halves and using them just like the DNA strands: one half stores the complementary data of the other, at any given moment a comparison between the contents of the two halves being made. Because the storage data is continuously shifted [14] there is need only for a few logic gates to be added onto the existing design. A XOR gate implements the comparison process (Fig. 7A). This ensures the detection of any single fault at the expense of storage space. Its correction still remains unsolved. This is somewhat similar to the situation when the biological DNA suffers modifications. In the living cells, correct information is retrieved using the neighborhood around the spotted fault and the complementary strand. In our case, the neighborhood has usually no correlation with the fault, so the data recovery seems not to be possible. In biological terms this is equivalent to non-repairable DNA errors and these typically lead to the death of the cell. The policy considered in the case of errors in the memory molecule is to activate the KILL signal in order to deactivate the entire cell.

For example, instead of storing 16 bits of data, one memory molecule stores 8 bits of data (**M0**) and 8 bits of complementary data (*M0*) as indicated in Fig. 7A. At each clock cycle, the content of both **M0** and *M0* is shifted one position. Specifying the memory mode a molecule may operate in is done through three CREG configuration bits; there are 2 combinations still left unused, there is no problem implementing this new memory mode, which we will call *DNA memory mode*. The presence of the supplementary XOR gate in each molecule, supervising the correct shifting of data, is similar to the existence of the A-T and C-G links between the DNA strands. If an error occurs, the XOR gate will detect it and will forward its output to the KILL signal generator that will trigger hierarchical mechanisms of reconfiguration and re-initialization.

5.4 The KILL and UNKILL Mechanisms

As described in previous articles [9, 10, 14], the robustness of the self-repair mechanism at the molecular level is programmable (through the frequency of columns made of spare molecules). But even so, there are limits to the faults that can be repaired at this level. With the continuous extending of the versatility of molecules, such as introducing the memory modes, the existence of these limits is even more obvious.

The KILL signal is simply generated whenever a non-repairable fault occurs (that is, no more spare molecules are available). It propagates outwards from the non-repairable molecule, rendering al molecules transparent to horizontal signals and thus triggering the cellular-level self-repair briefly described in Subsection 5.1. In the case of a fault being detected inside a memory structure, it being non-repairable, the KILL signal will also be generated and the whole cell will be “killed”, meaning that it will cease to operate and will become transparent for the other cells.

In digital electronic systems, the majority of hardware faults that occur during operation are in fact *transient*, that is, they disappear after some time. Based on this

observation we might be able to avoid the penalty induced by killing an entire cell due to probably transient errors. This is to say that the parts of the circuit "killed" because of the detection of a fault could potentially come back to "life" after a brief delay. Detecting the disappearance of a fault and handling the "unkilling" at the cellular level proved to be quite a simple task. A killed cell is transparent to the array; being reset, nothing prevents us to resend once again the configuration stream that will restore the functionality of the cell if a sufficient number of detected errors were transient. Since the memory structures are configured in the very same way as the other molecules, the unkilling mechanism does not require to be changed.

6 Conclusions

As a long-term research project, Embryonics is going well on track for many years now. Throughout this time, we have been accumulating considerable experience and were witnesses to the advent of technology that enabled us to actually experiment our ideas. While we adapt continuously to the technological advances, it may be possible that these advances will render some of our specific mechanisms obsolete. But with self-replication [8] being one of the key issues in nanotechnology, we feel that our efforts through the Embryonics project on creating and perfecting bio-inspired computing systems are rewarded.

Though Embryonics is quite at an advanced stage, the technological limits prevent us from experimenting with a great number of cells. A lot of work remains to be done in this direction.

References

1. Barbieri, M.: The Organic Codes: The Basic Mechanism of Macroevolution. Rivista di Biologia / Biology Forum 91 (1998) 481-514.
2. Gilbert, S. F.: Developmental Biology. Sinauer Associates Inc., MA, 3rd ed. (1991).
3. Mange, D., Tomassini, M., eds.: Bio-inspired Computing Machines: Towards Novel Computational Architectures. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland (1998).
4. Mange, D., Sanchez, E., Stauffer, A., Tempesti, G., Marchal, P., Piguet, C.: Embryonics: A New Methodology for Designing Field-Programmable Gate Arrays with Self-Repair and Self-Replicating Properties. IEEE Transactions on VLSI Systems, 6(3), (1998) 387-399.
5. Negrini, R., Sami, M. G., Stefanelli, R.: Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays. The MIT Press, Cambridge, MA (1989).
6. Ortega, C., Tyrrell, A.: Reliability Analysis in Self-Repairing Embryonic Systems. Proc. 1st NASA/DoD Workshop on Evolvable Hardware, Pasadena, CA (1999) 120-128.
7. Shibayama, A., Igura, H., Mizuno, M., Yamashina, M.: An Autonomous Reconfigurable Cell Array for Fault-Tolerant LSIs. Proc. 44th IEEE International Solid-State Circuits Conference, San Francisco, CA (1997) 230-231, 462.

8. Sipper, M.: Fifty Years of Research on Self-Replication: an Overview. Artificial Life, 4(3) (1998) 237-257.
9. Tempesti, G.: A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes. Ph.D. Thesis No. 1827, EPFL, Lausanne (1998).
10. Tempesti, G., Mange, D., Stauffer, A.: Self-Replicating and Self-Repairing Multicellular Automata. Artificial Life, 4(3) (1998) 259-282.
11. Wolfram, S.: Theory and Applications of Cellular Automata. World Scientific, Singapore (1986).
12. Wolpert, L.: The Triumph of the Embryo. Oxford University Press, New York (1991).
13. Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Perez-Uribe, A., Stauffer, A.: A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems. IEEE Transactions on Evolutionary Computation, 1(1) (1997) 83-97.
14. Prodan, L., Tempesti, G., Mange, D., Stauffer, A.: Biololy Meets Electronics: The Path to a Bio-Inpired FPGA. Proc. 3rd International Conference on Evolvable Systems: From Biology to Hardware, Edinburgh, Scotland, UK (2000) 187-196.
15. Terry, T. M.: www.sp.uconn.edu/~bi107vc/fa99/terry/DNA.html