Proc. Fourth International Conference on Technical Informatics (CONTI 2000), pp. 99-104.
Universitatii Politehnica din Timisoara (Romania), 2000.

Embryonics: Artificial Cells Made of Artificial Molecules

Lucian Prodan
Polytechnic University of Timisoara
RO-1900 Timisoara, Romania
E-mail: lprodan@cs.utt.ro
Gianluca Tempesti, Daniel Mange, André Stauffer
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland
E-mail: name.surname@epfl.ch

Abstract

Embryonics (embryonic electronics) is a research project which attempts to draw inspiration form the world of biology to design better digital computing machines, and notably massively parallel arrays of processors. In the course of the development of our project, we have realized that the use of programmable logic circuits (field-programmable gate arrays, or FPGAs) is, if not indispensable, at least extremely useful. This article describes the main concepts of Embryonics and some of the peculiar features of the FPGA we designed to efficiently implement our embryonic machines. More particularly, we overview the mechanisms taken from the world of biology and how we brought them in the world of silicon.

1. Introduction

A human being is made up of some 60 trillion (60×10^{12}) cells. At each instant, in each of these cells, the genome, a ribbon of 2 billion characters, is decoded to produce the proteins necessary for the survival of the organism. The parallel execution of 60 trillion genomes in as many cells occurs ceaselessly from the conception to the death of the individual. Faults are rare and, in the majority of cases, successfully detected and repaired. This astounding degree of parallelism is the inspiration of the Embryonics (*embryonic electronics*) project [3, 4, 11], which tries to adapt some of the development processes of multicellular organisms to the design of novel, robust architectures for massive parallelism in silicon. 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.

As a consequence, in designing our bio-inspired computing machines, we do not try to *imitate* life, but rather to extract some useful ideas from some the most fundamental

mechanisms of living creatures. More particularly, we are interested in the process of *ontogenesis* [2, 13], the development of a single organism from a single cell to an adult.

2. Overview and Motivations

With the exception of unicellular organisms (such as bacteria), living beings share three fundamental features [14]: m*ulticellular organization*, c*ellular division*, and *cellular differentiation*. A consequence of these features is that each cell is "universal", as it contains the whole of the organism's genetic material, the genome. This makes the living organisms capable of self-repair (cicatrization) or self-replication (cloning or budding). These two properties, based on a multicellular tissue, are essentially unique to the living world.

Taking into account the differences between the real, biological world and silicon, we developed a quasi-biological system architecture based on four levels of organization (Figure 1), described in detail in previous articles [3, 10, 11]. This article covers the mechanisms used by Embryonics to give silicon cells robustness similar to their biological siblings.

3. Artificial Cells and Artificial Molecules

Driven by the continuously changing environment, living beings developed hierarchical self-repair and self-replicating mechanisms that exhibit an almost perfect efficiency. Embryonics brings the worlds of biology and electronics closer, by implementing in silicon these features. As shown in Figure 2, 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.

Each cell stores a copy of the genes of all the cells of the organism (*operative genome*) [7], and determines which gene to execute depending on its position (X and Y coordinates) within the organism, implementing *cellular differentiation*. For example, in Figure 2 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.

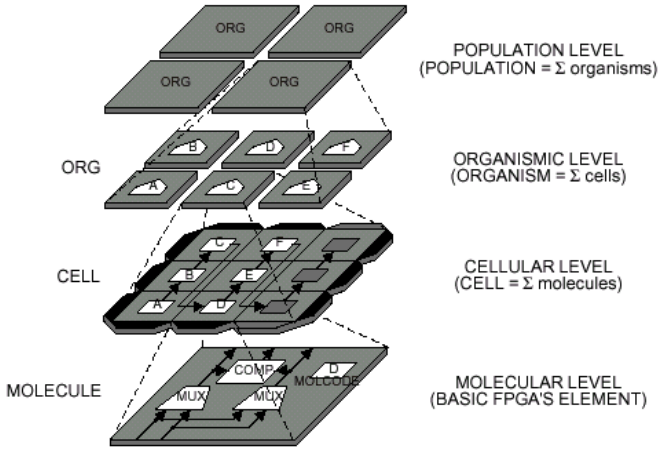

Figure 1: The four levels of organization of Embryonic systems.

In living cells, the genetic information circulates *sequentially*. A simple yet powerful enough memory structure enables data to circulate synchronously inside it, much as the ribosome processes the genome inside a living cell [1].

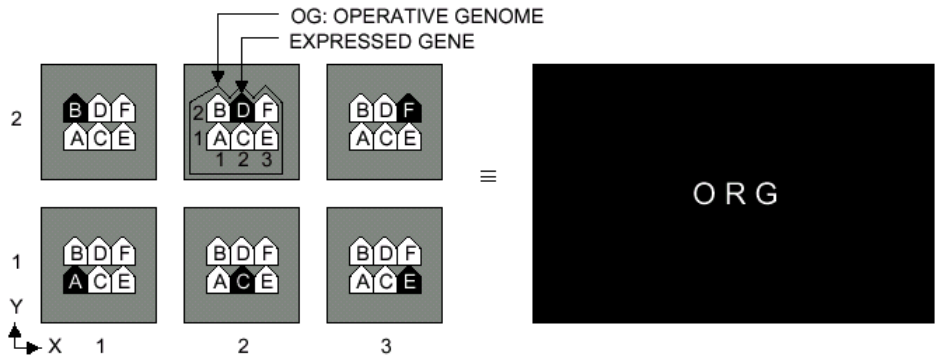

Figure 2: The multi-cellular organization of our artificial organisms.

4. Artificial Molecules

The lowest, most basic level of organization of our system is the molecular level, implemented as a two-dimensional array of programmable logic elements (the molecules), a type of circuit known as a *field-programmable gate array* (FPGA). The reasons why we decided to introduce a molecular level in our systems are explained in detail elsewhere [3, 10, 11]. Essentially, we require a substrate of programmable logic to be able to adapt the size and structure of our cells to a given application. FPGAs are an obvious answer to this problem: they provide us with a uniform surface of programmable elements (our *molecule*s) which can be assigned a function at runtime via a software configuration (in our case, the MOLCODE). These elements can then be put together through a set of programmable connections to realize virtually any kind of digital circuit, and notably our artificial cells (Figure 3). Our new FPGA, known as *MuxTree* [7, 11], implements all the features required of our artificial molecules to efficiently implement our bio-inspired machines.

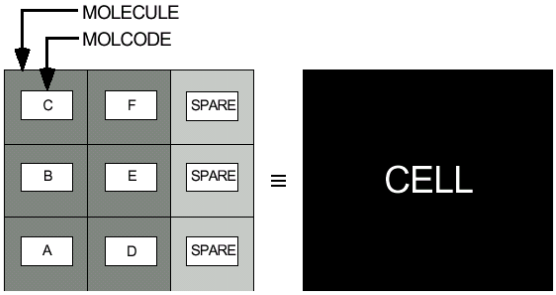

Figure 3: Multi-molecular structure of our artificial cell.

The molecule's architecture is based on a multiplexer coupled with a D-type flip-flop, allowing implementations of complex sequential systems [14]. It also features two sets of connections: a fixed, directional network for communication between neighbors, and a

programmable, non-directional network for long-distance communication (routed through the switch block SB). The bits required to assign the molecule's functionality are stored in the 20-bit wide shift register CREG. The configuration process is detailed elsewhere [14].

5. Self-Repair

To endow our artificial organisms with features similar to those of living beings, we provide a two-level mechanism for self-repair, involving both the cellular and the molecular level. The two levels cooperate to produce a higher level of robustness than would be allowed by a single level.

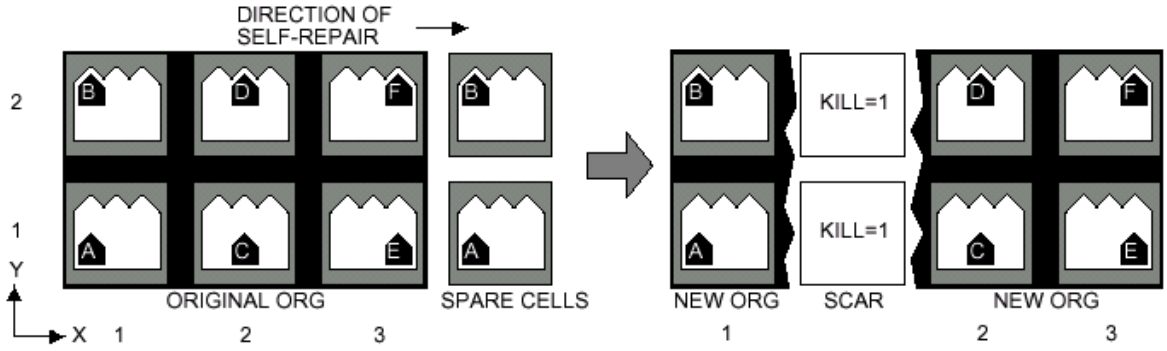

Figure 4: Self-repair of the cellular level through coordinate recomputation.

We begin with the self-repair at the cellular level. The redundant storage of the entire genome in every cell is obviously expensive. But the advantage of it is making the cell *universa*l, that is, potentially capable of executing any one of the functions required by the organism. This is a huge advantage for implementing *self-repai*r, the electronic equivalent of what we call healing of living systems. In fact, 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 are available (Figure. 6) [14]. All the cells in the faulty column "disappear" from the array, that is, become transparent with respect to all horizontal signals. Since the coordinates are computed locally from the neighbors' coordinates, any disappearance forces all the cells to right of the dead column to recalculate their coordinates, completing the reconfiguration of the array [6, 11].

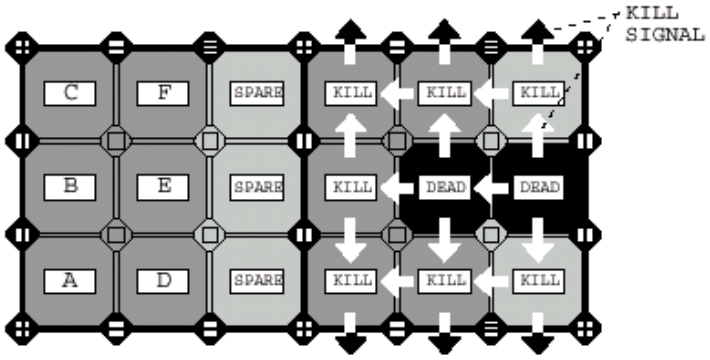

Figure 5: Faults in adjacent molecules activate the KILL signal.

Let us talk now about self-repair at the molecular level. In order to avoid the stiff penalty inherent in killing a column of processors for every fault in the array, we introduced a certain degree of fault tolerance at the molecular level [10]. The homogeneous architecture of our FPGA simplifies reconfiguration to a considerable extent [5, 8], becoming a simple question of shifting the configuration of the faulty molecule to its right (similarly to what happens during configuration, as shown in Figure 4) and redirecting the array's connections. The determination of these spare molecules is in fact one of the most powerful features of our system [12], since we can exploit the space divider to dynamically allocate some columns as spares. This is done at configuration time, and thus the fault tolerance of our FPGA becomes programmable (and can thus be adapted to the circumstances and the operating conditions).

Even if the robustness of the self-repair mechanism at the molecular level is programmable, there are limits to the faults which can be repaired at this level. Notably, if a fault is large enough to affect multiple adjacent molecules on the same row or if it occurs in a non-repairable part of the molecule, no amount of redundancy will let the FPGA repair itself. Whenever the molecular self-repair fails, the cellular level of self-repair is activated. This occurs by generating a KILL signal which propagates outwards from the non-repairable molecule (Figure 5). This signal will "destroy" all the molecules within a column of cells (as defined by the space divider), rendering them transparent to horizontal signals (in the same way as the unused spare molecules are transparent), thus activating the cellular-level self-repair described above. The molecular and the cellular levels of our system thus cooperate to assure a high degree of robustness to the system.

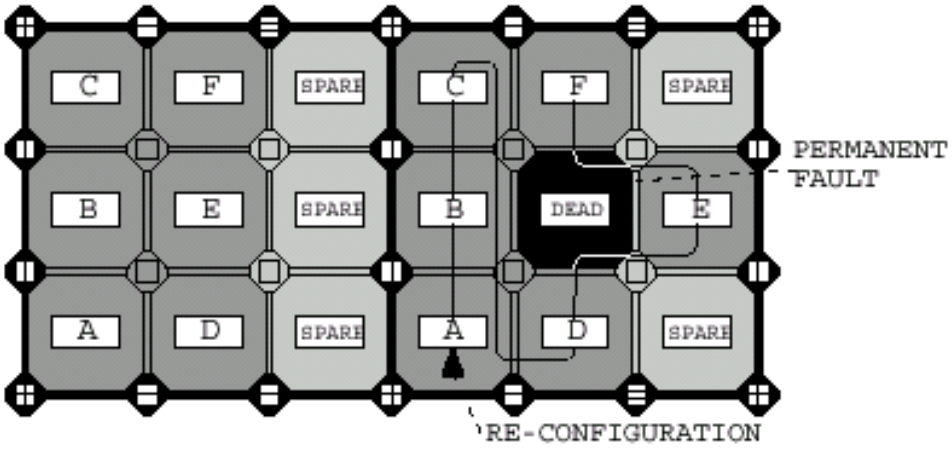

Figure 6: While the left-hand side cell operates, the right-hand side cell is “unkilled”.

In digital electronic systems, the majority of hardware faults occurring in the silicon substrate are in fact *transient*, that is, disappear after a short span of time. This led us to the following optimization: the parts of the circuit which have been "killed" because of the detection of a fault could potentially come back to "life" after a brief delay. This feature proved easy to be implemented at the cellular level. We have seen that self-repair at the cellular level consists of "destroying" (i.e., resetting) the configuration of all the molecules within a column of cells, with the effect of making the column "invisible" to the array. As it has been reset, however, nothing prevents us from sending once more the configuration bitstream to the deactivated molecules (Figure 6); if some or all the faults have vanished, then the functionality of the cells will be recovered.

6. Conclusions and Future Directions

The Embryonics project has been steadily advancing for many years now. Throughout this time, we have been accumulating considerable experience in trying to adapt biological concepts to the world of electronics. Of course, Embryonics represents only one of the many possible approaches to bio-inspiration. However, we feel that other projects wishing to draw inspiration from the ontogenetic development [2, 12] of living beings will be faced with many of the same problems, and will be likely to find solutions not too dissimilar from our own. Self-replication [9], for example, is one of the key issues in nanotechnology. Of course, in the long run, technological advances (for example, the development of nanotechnologies) will probably render many of our specific mechanisms obsolete.

References

[1] M. Barbieri. "The Organic Codes: The Basic Mechanism of Macroevolution". *Rivista di Biologia / Biology Forum* 91, 1998, pp. 481-514.

[2] S. F. Gilbert. *Developmental Biology*. Sinauer Associates Inc., Massachusetts, 3rd edition, 1991.

[3] D. Mange, M. Tomassini, eds. *Bio-inspired Computing Machines: Towards Novel Computational Architectures*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.

[4] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, C. Piguet. "Embryonics: A New Methodology for Designing Field-Programmable Gate Arrays with Self-Repair and Self-Replicating Properties". *IEEE Transactions on VLSI Systems*, 6(3), September 1998, pp. 387-399.

[5] R. Negrini, M.G. Sami, R. Stefanelli. *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays*. The MIT Press, Cambridge, MA, 1989.

[6] C. Ortega, A. Tyrrell, "Reliability Analysis in Self-Repairing Embryonic Systems". *Proc. 11th NASA/DoD Workshop on Evolvable Hardware*, Pasadena, CA, July 1999, pp.120-128.

[7] L. Prodan. *MuxTree Specifications*. Internal Report. Logic Systems Laboratory, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.

[8] Shibayama, H. Igura, M. Mizuno, M. Yamashina. "An Autonomous Reconfigurable Cell Array for Fault-Tolerant LSIs". In: *Proc. 44th IEEE International Solid-State Circuits Conference*, San Francisco, California, February 1997, pp. 230-231 and 462.

[9] M. Sipper. "Fifty Years of Research on Self-Replication: an Overview". *Artificial Life*, 4(3), 1998, pp. 237-257.

[10] G. Tempesti. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes*. Ph.D. Thesis No. 1827, EPFL, Lausanne, 1998.

[11] G. Tempesti, D. Mange, A. Stauffer. "Self-Replicating and Self-Repairing Multicellular Automata". *Artificial Life*, 4(3), 1998, pp. 259-282.

[12] S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[13] L. Wolpert. *The Triumph of the Embryo*. Oxford University Press, New York, 1991.

[14] L. Prodan, G. Tempesti, D. Mange, A. Stauffer. *Biology Meets Electronics: the Path to a Bio-Inspired FPGA*. Proc. 3rd International Conference, ICES2000, Edinburgh, Scotland, UK, April 2000, pp. 187-196.