Self-Replication for Reliability: Bio-Inspired Hardware and the Embryonics Project

Gianluca Tempesti, Daniel Mange, Pierre-André Mudry, Joël Rossier, André Stauffer
Ecole Polytechnique Fédérale de Lausanne (EPFL)
EPFL-IC-GRTEM (INN239), Station 14
CH-1015 Lausanne, Switzerland
Phone: +41-21-6932676
gianluca.tempesti@epfl.ch

ABSTRACT
The growth and operation of all living beings are directed by the interpretation, in each of their cells, of a chemical program, the DNA string or *genome*. This process is the source of inspiration for the Embryonics (embryonic electronics) project, whose final objective is the design of highly robust integrated circuits, endowed with properties usually associated with the living world: self-repair (cicatrization) and self-replication. The Embryonics architecture is based on four hierarchical levels of organization: 1) the basic primitive of our system is the *molecule*, a multiplexer-based element of a novel programmable circuit; 2) a finite set of molecules makes up a *cell*, essentially a small processor with an associated memory; 3) a finite set of cells makes up an *organism*, an application–specific multiprocessor system; 4) the organism can itself replicate, giving rise to a *population* of identical organisms. In this paper, we provide an overview of our latest research in the domain of the self-replication of processing elements within a programmable logic substrate, a key prerequisite for achieving system-level fault tolerance in our bio-inspired approach.

Categories and Subject Descriptors
B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance
C.1.3 [**Processor Architectures**]: Other Architecture Styles – *Adaptable architectures*
C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms
Design, Reliability.

Keywords
Bio-inspired architectures, growth, embryonic electronics, self-replication, self-repair, hierarchical fault tolerance.

CF'06, May 3–5, 2006, Ischia, Italy.

1. INTRODUCTION
A human being consists of approximately 60 trillion ($60x10^{12}$) cells. At each instant, in each of these 60 trillion cells, the *genome*, a ribbon of 2 billion characters, is decoded to produce the proteins needed for the survival of the organism. This genome contains the ensemble of the genetic inheritance of the individual and, at the same time, the instructions for both the construction and the operation 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 process is remarkable for its complexity and its precision. Moreover, it relies on completely discrete information: the structure of DNA (the chemical substrate of the genome) is a sequence of four bases, usually designated with the letters A (adenine), C (cytosine), G (guanine), and T (thymine).

Our *Embryonics* project (for *embryonic electronics*) [7] is inspired by the basic processes of molecular biology and by the embryonic development of living beings [6][23]. By adopting certain features of cellular organization, and by transposing them to the two-dimensional world of integrated circuits on silicon, we wish to show that properties unique to the living world, such as *self-replication* and *self-repair*, can also be applied to artificial objects (integrated circuits).

We should however emphasize that the goal of bio-inspiration is *not* the modelization or the explication of actual biological phenomena: our final objective is the development of very large scale integrated circuits capable of self-repair and self-replication. Self-repair allows partial reconstruction in case of a minor fault, while self-replication allows complete reconstruction of the original device in case of a major fault. These two properties are particularly desirable for complex systems requiring improved reliability in several contexts and for several applications:

1. Short-term applications [12], such as those which require very high levels of reliability (e.g., avionics, medical electronics), those designed for hostile environments (e.g., space) where increased radiation levels reduce the reliability of components, or those which exploit the latest technological advances, notably the drastic device shrinking, low power supply levels, and increasing operating speeds, that accompany the technological evolution to deeper submicron levels and significantly reduce the noise margins and increase the soft-error rates [1].

2. Medium-term applications, where there is a need for very complex integrated circuits capable of on-line self-repair, dispensing with the systematic detection of faults at fabrication [24].
3. Long-term applications, executed on systems built with imperfect components: this is von Neumann's historical idea [21], the basis of all present projects aimed at the realization of complex integrated circuits at the molecular scale (nanoelectronics) [4][5][15][22].

Self-replication, or "cloning", is one of the major tools exploited to achieve fault tolerance in nature. It can, however, be justified independently of self-repair:

- to replicate, within a programmable logic substrate (FPGA), functionally equivalent systems [13];
- to mass-produce future-generation integrated circuits, based on molecular-scale nanoelectronic components [11];
- to finally accomplish John von Neumann's unachieved dream, that is, the realization of a self-replicating automaton endowed with the properties of universal computation and construction [21].

These emerging needs require the development of a new design paradigm that supports efficient online testing and self-repair solutions and that can efficiently realize the self-replication of complex electronic structures. We have described elsewhere the logic-level details of the implementation of the self-repair and self-replication abilities of our approach [7][8][9][10][17][18]. In this article, we shall concentrate particularly on the connection between the two mechanisms and on the latest results of our research at the cellular (processor) level.

2. FROM BIOLOGY TO HARDWARE

The majority of living beings, with the exception of unicellular organisms such as viruses and bacteria, share three fundamental features:

1. *Multicellular organization* divides the organism into a finite number of *cells*, each realizing a unique function (neuron, muscle, intestine, etc.). The same organism can contain multiple cells of the same kind.
2. *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).
3. *Cellular differentiation* defines the role and function of each cell of the organism. This specialization occurs through the expression of one or more *genes* in the genome and depends essentially on the position of the cell in the organism.

A consequence of these three features is that each cell is "universal", since it contains the whole of the organism's genetic material, the genome. Should a trauma occur, living organisms are thus potentially capable of self-repair (cicatrization) or self-replication (cloning or budding) [23].

The two properties of self-repair and self-replication based on a multicellular tissue are unique to the living world. The main goal of the Embryonics project is the implementation of the above three features of living organisms in an integrated circuit in silicon to obtain the properties of self-repair and self-replication.

Our approach is based on four hierarchical levels of organization (Fig. 1):

- The basic primitive of our system is the *molecule*, the element of a novel programmable logic circuit.
- A finite set of molecules makes up a *cell*, essentially a small processor with the associated memory, executing a program that finds a biological equivalent in the *genome* that stores the information required for the operation of any organism.
- A finite set of cells is an *organism*, an application-specific multiprocessor system.
- The organism can itself replicate, giving rise to a *population* of identical organisms, the highest level of our hierarchy.

3. THE ORGANISM

The environment of our quasi-biological approach is imposed by the structure of electronic circuits, and consists of a finite (but arbitrarily large) two-dimensional surface of silicon. This surface is divided into rows and columns, whose intersections define the cells. All the cells have an identical physical structure (i.e., an identical set of logic operators and connections), making the cellular array is homogeneous. As the program in each cell (the genome) is also identical, only the *state* of a cell (i.e., the contents of its registers) differentiates it from its neighbors.

In this Section, we first show how to implement in our artificial organisms the three fundamental features of multicellular organization, cellular differentiation, and cellular division.

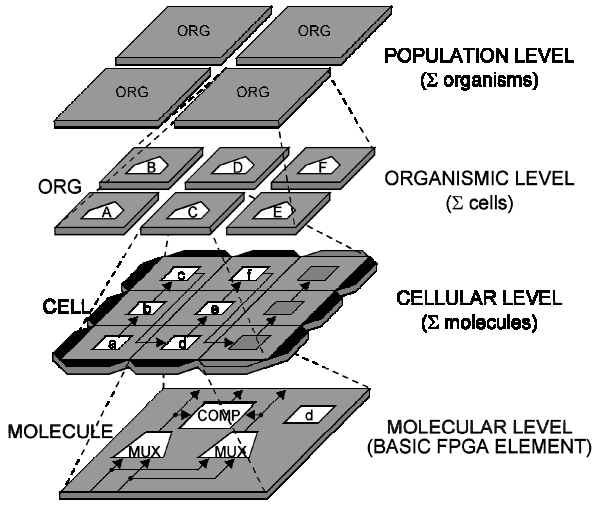

Fig. 1 The Embryonics landscape: a 4-level hierarchy.

3.1 The Organism's Features

Multicellular organization divides the artificial organism (`ORG`) into a finite number of cells (Fig. 2). Each cell (`CELL`) realizes a unique function, defined by a sub-program called the *gene* of the cell and selected as a function of the values of both the horizontal (`X`) and the vertical (`Y`) coordinates (in Fig. 2, the genes are labeled `A` to `F` for coordinates `X,Y=1,1` to `X,Y=3,2`). Let us call operative genome (`OG`) a program containing all the genes of an artificial organism, where each gene (`A` to `F`) is a sub-program characterized by a set of instructions and by the cell's position (coordinates `X,Y=1,1` to `X,Y=3,2`).

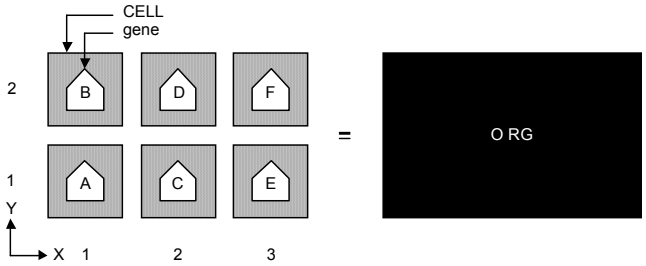

Fig. 2 Multicellular organization of a 6-cell organism ORG.

Let then each cell contain the entire operative genome OG (Fig. 3): depending on its position in the array, i.e., its place within the organism, each cell can then interpret the operative genome and extract and execute the gene which defines its function. Note that, in the majority of applications, as in natural systems, there is not a one-to-one correspondence between the cells and the gene, since many cells in an organism execute the same gene (reducing the size of the genome and limiting the overhead).

Storing the whole operative genome in each cell makes the cell *universal*: given the proper coordinates, it can execute any of the genes of the operative genome to implement *cellular differentiation*. In our artificial organism, any cell `CELL[X,Y]` continuously computes its coordinate `X` by incrementing the coordinate `WX` of its west neighbor. Likewise, it continuously computes its coordinate `Y` by incrementing the coordinate `SY` of its south neighbor. Taking into consideration these computations, Fig. 4 shows the final operative genome OG of the organism ORG.

At startup, the first cell or *zygote*, arbitrarily defined as having the coordinates `X,Y=1,1`, holds the one and only copy of the operative genome OG. After time `t1`, the genome of the zygote (*mother* cell) is copied into the neighboring (*daughter*) cells to the east (`CELL[2,1]`) and to the north (`CELL[1,2]`). This process of *cellular division* continues until the six cells of the organism are completely programmed.

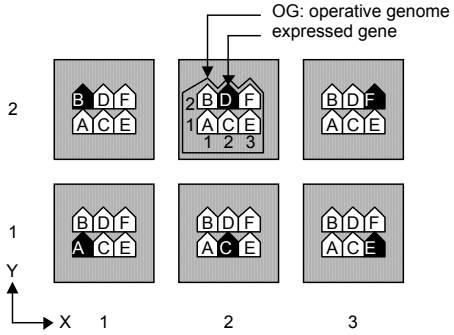

Fig. 3 Cellular differentiation.

```
OG: operative genome
X = WX+1
Y = SY+1
case of X,Y:
  X,Y = 1,1: do gene A
  X,Y = 1,2: do gene B
  X,Y = 2,1: do gene C
  X,Y = 2,2: do gene D
  X,Y = 3,1: do gene E
  X,Y = 3,2: do gene F
```

Fig. 4 The operative genome OG of the organism ORG.

3.2 The Organism's Properties

The *self-replication* or *cloning of the organism*, i.e., the production of an exact copy of the original, rests on two assumptions:

- there exists a sufficient number of spare cells in the array (at least six in the example of Fig. 5) to contain the additional organism;
- the calculation of the coordinates produces a cycle (`X=1→2→3→1…` and `Y=1→2→1…` in Fig. 5, implying `X=(WX+1) mod 3` and `Y=(SY+1) mod 2`).

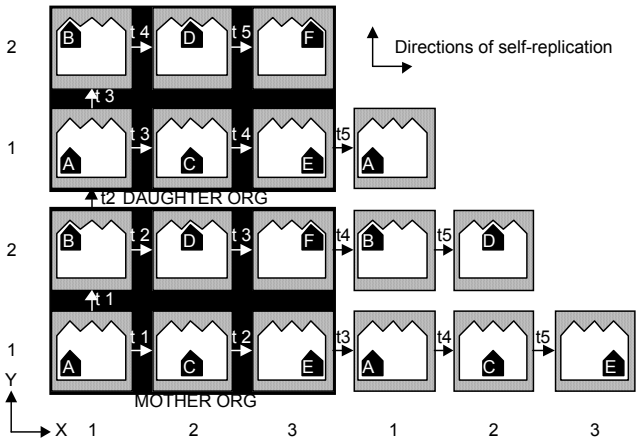

Fig. 5 Self-replication of a 6-cell organism ORG (situation at time t5 after 5 cellular divisions)

As the same pattern of coordinates produces the same pattern of genes, self-replication can be easily accomplished if the operative genome OG, associated with the homogeneous array of cells, produces several occurrences of the basic pattern of coordinates. In our example (Fig. 5), the repetition of the vertical coordinate pattern (`Y=1→2→1→2`) in a sufficiently large array of cells produces a copy, the *daughter organism*, of the original *mother organism*. Given a sufficiently large space, self-replication can be repeated for any number of specimens in the `X` and/or the `Y` axes.

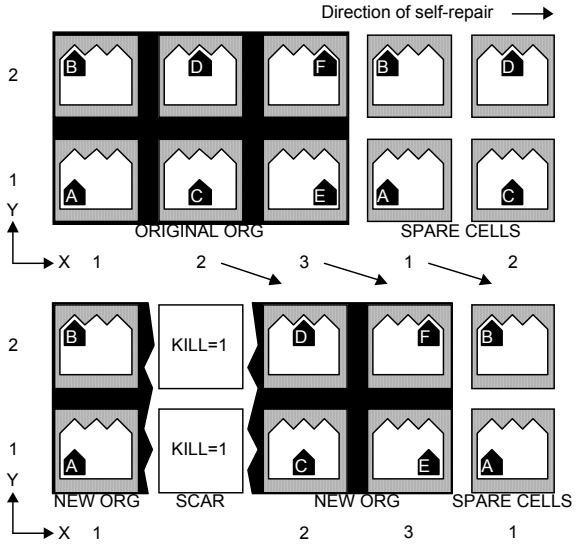

Fig. 6 Organismic self-repair.

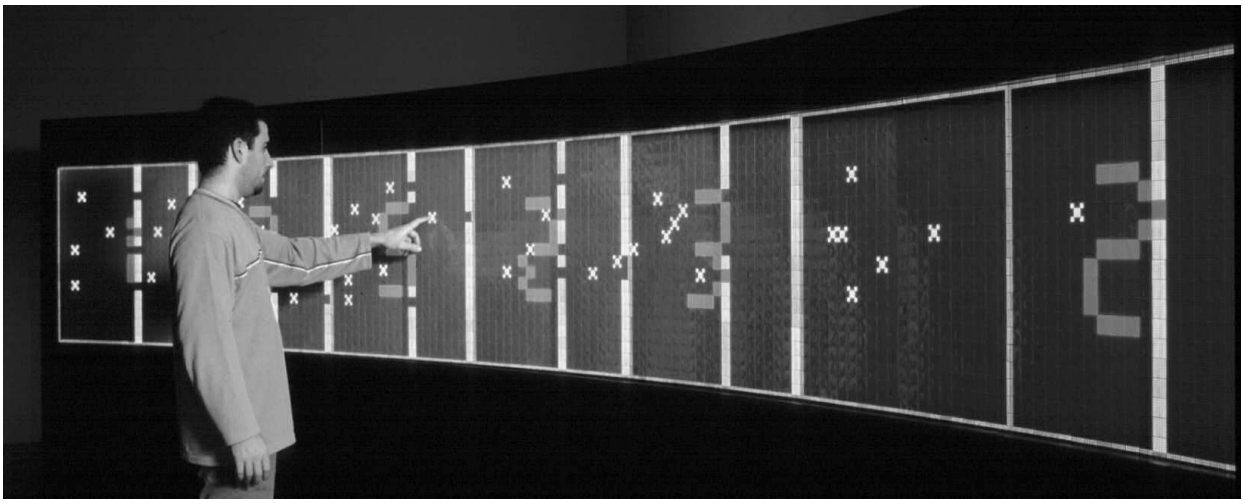

Fig. 7 Implementation of a self-repairing and self-replicating system on the BioWall.

To implement the *self-repair of the organism*, we decided to use spare cells to the right of the original organism (Fig. 6). The existence of a fault is detected by a `KILL` signal which is calculated in each cell by a built-in self-test mechanism realized at the molecular level (see below). The state `KILL=1` identifies the faulty cell, and the entire column to which the faulty cell belongs is considered faulty, and is deactivated (column `X=2` in Fig. 6).

All the functions (`X` coordinate and gene) of the cells to the right of the column `X=1` are shifted by one column to the right. Obviously, this process requires as many spare columns to the right of the array as there are faulty cells or columns to repair (e.g., two spare columns, tolerating two successive faulty cells, in Fig. 6). It also implies that the cell needs to be able to bypass the faulty column and divert to the right all the required signals (such as the operative genome, the `X` coordinate, and the data busses).

It is this latter consideration that led us to the choice to destroy an entire column of cells whenever a faulty cell is detected. This choice, while costly, does represent a considerable gain in routing resources, and the increased penalty is offset, as described below, by the presence of a molecular fault-tolerance mechanism which considerably reduces the need for this kind of self-repair.

Of course, given a sufficient number of cells, it is possible to combine self-repair in the `X` direction, and self-replication in both the `X` and `Y` directions.

The approach described herein has been tested and verified in actual hardware on several small applications, implemented on the BioWall [19], a machine specifically designed for the prototyping of cellular systems. In particular, the self-repair and self-replication properties were both implemented in hardware and verified (Fig. 7).

4. THE CELL

In each cell of every living being, the genome is translated sequentially by a chemical processor, the *ribosome*, to create the proteins needed by the organism. The ribosome itself consists of molecules and its description is part of the genome.

As mentioned, in the Embryonics project each cell is a small processor, sequentially executing the instructions of the operative genome `OG`. The need to realize organisms of varying complexity has led us to design an artificial cell characterized by a flexible architecture implemented using a new kind of field-programmable gate array (FPGA).

Each element of this FPGA is then equivalent to a *molecule*, and an appropriate number of these artificial molecules allows us to realize our application-specific processors (the artificial cells).

4.1 Cellular Architecture

The requirements of the cellular layer of our systems led us to define a new family of customizable processors based on the MOVE paradigm, also known as the Transport-Triggered Architecture (TTA) [2][3][16], originally developed for the design of application-specific dataflow processors (processors where the instructions define the flow of data, rather than the operations to be executed).

In many respects, the overall structure of a TTA-based system is fairly conventional (an advantage since our ultimate goal is the realization of conventional computation on our bio-inspired systems): data and instructions are fetched to the processor from the main memory using standard mechanisms (caches, memory management units, etc.) and are decoded as in conventional processors. The basic differences lay in the architecture itself, and hence in the instruction set.

Rather than being structured, as is usual, around a more or less serial pipeline, a MOVE processor (Fig. 8) relies on a set of Functional Units (FUs) connected together by one or more transport busses. All the computation is carried out by the functional units (examples of such units can be adders, multipliers, register files, etc.) and the role of the instructions is simply to move data from and to the FUs in the order required to implement the desired operations. Since all the functional units are uniformly accessed through input and output registers, instruction decoding is reduced to its simplest expression, as only one instruction is needed: `move`.

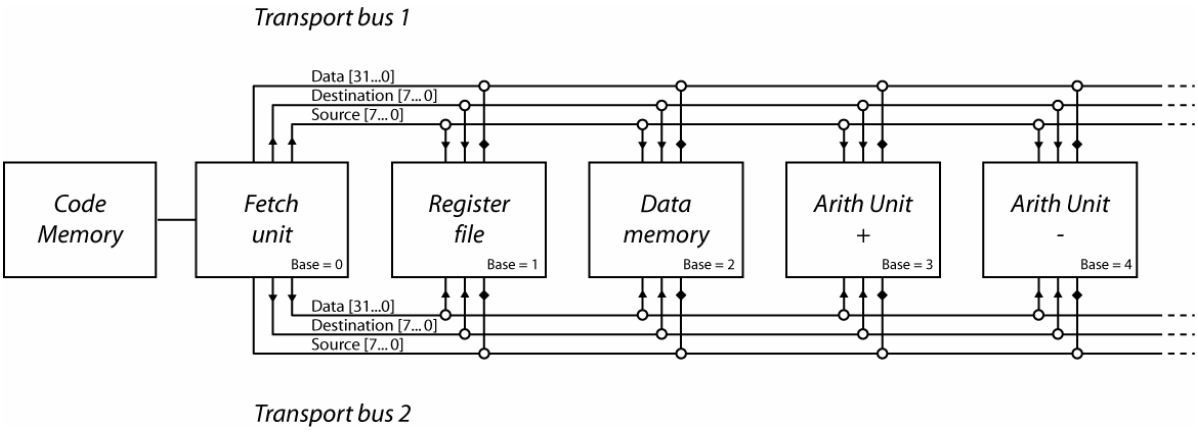

Fig. 8 Internal structure of a TTA processor.

TTA `move` instructions trigger operations which, in the simplest case, correspond to normal RISC instructions. For example, in order to add two numbers a RISC `add` instruction has to specify two operands and, most of the time, a destination register to store the result. The MOVE paradigm requires a slightly different approach to obtain the same result: instead of using a specific `add` instruction, the program moves the two operands to the input registers of a functional unit that implements the add operation. The result can then be retrieved in the output register of this functional unit and moved wherever it is needed (either to a register bank or, more interestingly, to the input of another unit, bypassing the register bank entirely).

This architecture, while obviously not directly inspired by biology, does meet some of the most relevant requirements for the implementation of the kind of bio-inspired systems defined within our approach. For example, since the TTA approach was designed for conventional computing, it is sufficiently powerful to allow the implementation of high-performance computing. Also, it is relatively compact and well-suited to the realization of complex networks of processors. But its key feature remains its versatility: since it allows FUs to be changed (mostly) without affecting the decode logic and the assembly language, MOVE processors are an ideal platform for the implementation of mechanisms related to cellular differentiation, allowing the processors to specialize for the desired application.

4.2 Cellular Features

Of course, in order to exploit our architecture's capability to specialize to execute a given application, the structure of the cells must not be fixed, but rather must be able to change structure depending on the task's requirements, much like in nature cells assume different sizes and shapes depending on their function. To allow this versatility, our cells are implemented on custom-designed programmable logic devices dedicated to the implementation of our systems and the concept of *molecule* makes its apparition within out hierarchy to represent the elements of our FPGA.

We will call *multimolecular organization* the use of many molecules to realize one cell. The configuration of the FPGA (that is, the information required to assign the logic function of each molecule) constitutes a second part of our artificial genome: the *ribosomic genome* `RG`. Fig. 9 shows an abstract example of a simple cell (`CELL`) consisting of six molecules, each defined by a *molecular code* or `MOLCODE` (`a` to `f`). The set of these six `MOLCODE`s constitutes the ribosomic genome `RG` of the cell.

The information contained in the ribosomic genome `RG` thus defines the logic function of each molecule and its connections to the other molecules in the cellular space by assigning a molecular code `MOLCODE` to it. To obtain a functional cell, we require two additional pieces of information:

- the *physical position* of each molecule in the cellular space (i.e., within the cell);
- the presence of one or more *spare columns*, composed of *spare molecules*, required for self-repair, as we shall see.

Normally hidden or implicit in conventional FPGA devices, all these aspects of the *molecular configuration* have to be treated explicitly in order to achieve the sought properties of self-replication and self-repair.

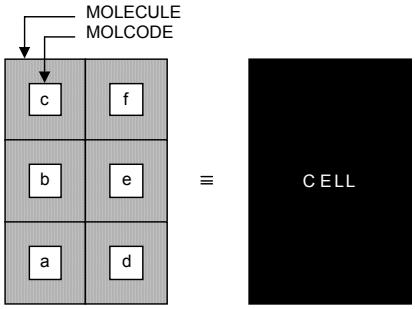

Fig. 9 Multimolecular organization.

Fault detection is a fundamental prerequisite for the introduction of self-repair in a system. The ability to detect that a fault has occurred is undoubtedly one of the most complex tasks in any repair process and conventional FPGAs are sorely lacking in this respect. In fact, while they allow fault detection at the circuit level (i.e., FPGAs can be used to implement self-checking circuits), the absence of fault detection mechanisms within the FPGA elements themselves severely limits the scope of this feature. In order to address this issue, we have developed several mechanisms to allow fault detection in our FPGAs. Described in detail elsewhere [7][17][18], these mechanisms can then be used to activate the self-repair processes within our systems.

4.3 Cellular Properties

A consequence of the multimolecular organization and of the molecular configuration of the FPGA is the ability, for any given cell, to propagate its ribosomic genome RG in order to automatically configure two daughter cells, architecturally identical to the mother cell, to the east and to the north, thus implementing *cellular self-replication*.

Cellular self-replication is a prerequisite for cellular division at the organismic level described above, during which the operative genome is copied from the mother cell into the daughter cells. We can summarize the two key roles of cellular self-replication:

- The construction of two daughter cells in order to grow a new organism or to repair an already existing one (genome *translation*).
- The distribution of an identical set of chromosomes to create a copy of the genome from the mother cell and program the daughter cells (genome *transcription*).

Our developmental mechanisms operate by allowing the set of molecular configurations that implement a cell to replicate itself, realizing a process not unlike the cellular division that underlies the growth of biological organisms. The latest incarnation of self-replication mechanisms within our project goes under the label of *Tom Thumb algorithm* [8][9] and can be seen as a universal approach to introduce self-replication in a programmable device.

Cellular self-replication plays another crucial role in our systems, like in biological organisms, as the basic mechanism that supports fault tolerance by introducing redundancy and by allowing the definition of spare cells in the system. Coupled with the organismic self-repair described above and with the appropriate fault detection mechanisms, self-replication enables robustness through a set or processes not unlike those occurring in nature during cicatrisation.

However, one of the most important lessons that can be drawn from nature in the domain of fault tolerance is that the presence of several mechanisms operating together at different levels of complexity provides considerable advantages over any single-level system. We tried to apply this lesson, normally ignored in conventional fault-tolerant approaches, within our systems by introducing a set of self-repair mechanisms within our molecular layer. These mechanisms are in charge of trying to repair small, isolated faults within the circuit and operate in cooperation with the higher-level organismic self-repair process.

In fact, if we consider that the death of a cell is quite expensive in terms of wasted resources, the ability to repair at least some of these faults at the cellular level (that is, without invoking the organismic self-repair mechanism) becomes highly desirable. The biological inspiration for this process derives from the set of molecular-level mechanisms that routinely repair radiation-induced or chemical errors within cells in nature. To mention but the best-known example, the DNA's double helix, the physical support of natural genomes, provides complete redundancy of the genome though the presence of complementary bases in the opposing branches of the helix.

This lesson led us to define a set of comparison-based mechanisms to detect the occurrence of a fault within our molecular substrate and to add to our system a *cellular self-repair* process that occurs at the molecular level within each single cell (Fig. 10). Based on the presence of columns of spare molecules (which can be specified dynamically within the Tom Thumb algorithm), this process allows minor faults to be repaired locally, without resorting to the organismic self-repair process.

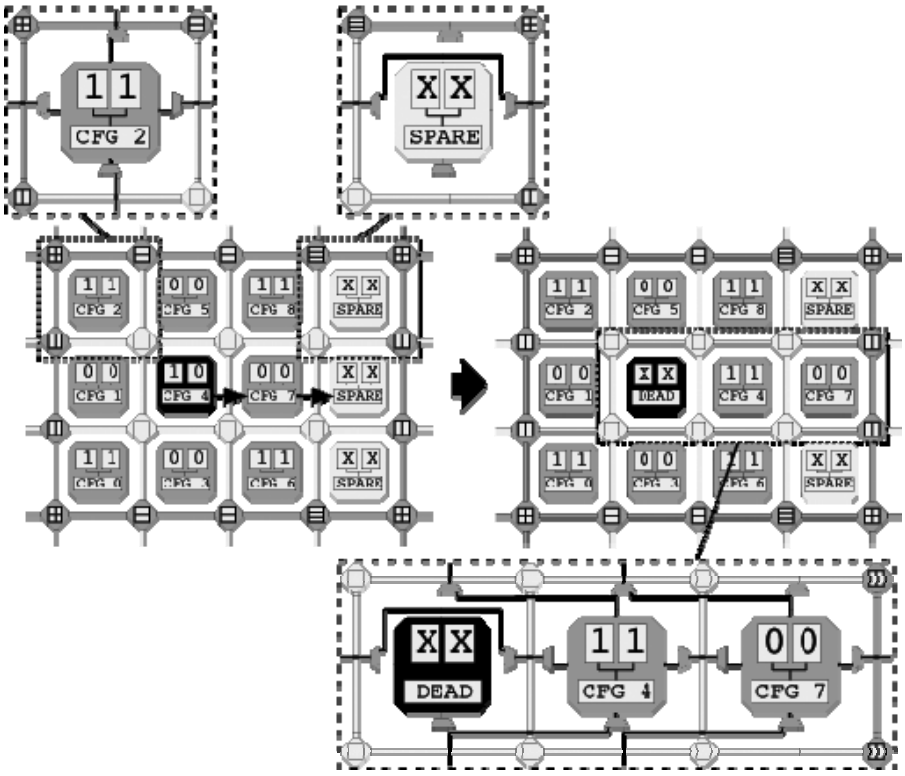

Fig. 10 Cellular self-repair process.

In this cellular self-repair process, each faulty molecule is deactivated, isolated from the rest of the FPGA, and replaced by a neighboring molecule, which will itself be replaced by a neighbor, and so on until a spare molecule (`SPARE` in Fig. 10) is reached, exploiting hardware mechanisms described in some detail elsewhere [7][17][18]. When too many molecules are defective and the self-repair mechanism at the molecular level cannot repair the system, the molecules generate the `KILL` signal required to activate the cellular-level self-repair mechanism described above (Fig. 6). The combination of these two processes allows the system to achieve a level of fault tolerance that could not be obtained by operating at a single level of complexity.

5. CONCLUSIONS

To resume, the final architecture of the Embryonics project is based on four hierarchical levels of organization which, described from the bottom up, are the following (Fig. 1):

- The basic primitive of our system is the *molecule*, the element of an FPGA that incorporates mechanisms for fault detection, self-replication, and self-repair. The logic function of each molecule is defined by its molecular code or `MOLCODE`.
- A finite set of molecules makes up a *cell*, essentially a processor with the associated memory. In a first programming step of the FPGA, the ribosomic genome `RG` defines the topology of the cell (that is, its width, height, and the presence and positions of spare molecules) and the logic function and connections of each molecule by assigning its molecular code or `MOLCODE`.
- A finite set of cells makes up an *organism*, an application-specific multiprocessor system. In a second programming step, the operative genome `OG` is copied into the memory of each cell to define the particular application executed by the organism.
- The organism can itself self-replicate, giving rise to a *population* of identical organisms, the highest level of our hierarchy.

The design process for implementing an application (described as a set of specifications) in an Embryonics system requires then the following stages:

- The original specifications are mapped onto a homogeneous array of cells. The software (a program) and the hardware (the architecture of the cell) are tailored according to the needs of the specific application. In biological terms, this program can be seen as the *operative genome* `OG`, or, in other words, the *operative part* of the final artificial genome.
- The hardware of the cell is implemented with a homogeneous array of artificial molecules. Spare columns are introduced in order to improve the global reliability. With our artificial cell, in analogy to the ribosome of a natural cell, the string of the molecular codes `MOLCODEs` can be considered as the *ribosomic genome* `RG` or the *ribosomic part* of the final genome.

The definition of a novel FPGA dedicated to the implementation of our bio-inspired systems is justified by an analysis of the complexity of our systems. While in fact the mechanisms we have developed to implement the different properties of our systems are digital in nature and could therefore be implemented on any commercial FPGA, the development of an FPGA adapted to our project has allowed us to greatly diminish their complexity. This observation motivated us to embark in a European project [20] aimed at the development of a novel electronic tissue for the implementation of bio-inspired systems. The "Reconfigurable POEtic Tissue" project, funded by the Future and Emerging Technologies programme (IST-FET) for the European Community, ran from 2001 to 2004 in collaboration with the universities of York, Barcelona (UPC), Lausanne, and Glasgow, and defined a novel programmable logic circuit specifically designed for the implementation of bio-inspired systems.

Keeping in mind that our final objective is the development of very large scale integrated (VLSI) circuits capable of self-repair and self-replication, we have shown that a hierarchical organization based on four levels (molecule, cell, organism, population of organisms) allows us to confront the complexity of real systems. The realization of several applications [19] demonstrates that our approach can satisfy the requirements of highly diverse artificial organisms and attain the two sought-after properties of self-repair and self-replication. Our current effort is leading us to the definition of a *methodology* for the design of such systems, forming the core of a complete *design environment* that will allow these properties to be seamlessly integrated to existing design and to existing design flows.

The future technical application of the Embryonics project is in the domain of nanoelectronics [11]. The concept of a self-replicating machine, or "assembler", capable of arranging "the very atoms" was first introduced by Drexler as a possible solution to the problem of the increasing miniaturization of VLSI circuits: as manufacturing technology advances beyond conventional lithography, some new, accurate, and low-cost approach to the fabrication of VLSI circuits is required, and self-replicating assemblers could be a remarkably powerful tool for this kind of application. Fault tolerance, a crucial feature for this kind of technology, would then be achieved by exploiting the massive redundancy introduced by this kind of approach (the same kind of redundancy that is omnipresent in biological systems) and by the use of self-repair processes operating at all levels of complexity.

6. REFERENCES

[1] "A D&T Roundtable: Online Test". *IEEE Design & Test of Computers*, Vol. 16, No. 1, January-March 1999, pp. 80-86.

[2] H. Corporaal. Microprocessor Architectures from VLIW to TTA. John Wiley, 1998.

[3] H. Corporaal, H. Mulder. "MOVE: A framework for high-performance processor design". *Proc. Intl. Conf. on Supercomputing*, pp. 692-701, 1991.

[4] J. R. Heath, P. J. Kuekes, G. S. Snider, R. S. Williams. "A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology". *Science*, Vol. 280, No. 5370, 12 June 1998, pp. 1716-1721.

[5] P. Kuekes. "Molecular Manufacturing: Beyond Moore's Law". Invited Talk. *Proc. Field-Programmable Custom Computing Machines (FCCM'99)*, Napa, CA, Apr. 1999.

[6] D. Mange, M. Sipper, P. Marchal. "Embryonic electronics". *BioSystems*, Vol. 51, No. 3, 1999, pp. 145-152.

[7] D. Mange, M. Sipper, A. Stauffer, G. Tempesti. "Towards Robust Integrated Circuits: The Embryonics Approach". *Proceedings of the IEEE*, 88(4), April 2000, pp. 516-541.

[8] D. Mange, A. Stauffer, E. Petraglio, G. Tempesti. "Embryonic Machines that Divide and Differentiate". In A. J. Ijspeert, D. Mange, M. Murata, S. Nishio, Eds., *Bio-ADIT 2004 On-Conference Proceedings*, pp. 328-343. Osaka University Forum 2004, Osaka, 2004.

[9] D. Mange, A. Stauffer, E. Petraglio, G. Tempesti. "Self-replicating loop with universal construction", *Physica D*, Vol. 191, No 1-2, 15 April 2004, pp. 178-192.

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

[11] R. C. Merkle. "Making Smaller, Faster, Cheaper Computers". *Proceedings of the IEEE*, Vol. 86, No. 11, November 1998, pp. 2384-2386.

[12] M. Nicolaidis. "Future Trends in Online Testing: a New VLSI Design Paradigm?". *IEEE Design and Test of Computers*, Vol. 15, No. 4, 1998, p. 15.

[13] S. R. Park, W. Burleson. "Configuration Cloning: Exploiting Regularity in Dynamic DSP Architectures". *Proc. ACM/SIGDA Intl. Symp. on Field Programmable Gate Arrays (FPGA'99)*, Monterey, CA, Feb. 1999, pp. 81-89.

[14] M. Sipper, D. Mange, E. Sanchez. "Quo Vadis Evolvable Hardware?". *Communications of the ACM*, Vol. 42, No. 4, April 1999, pp. 50-56.

[15] R. F. Service. "Organic Molecule Rewires Chip Design". *Science*, Vol. 285, No. 5426, 16 July 1999, pp. 313-315.

[16] D. Tabak, G.J. Lipovski. "MOVE architecture in digital controllers". *IEEE Transactions on Computer*s C-29, pp. 180-190, 1980.

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

[18] G. Tempesti, D. Mange, A. Stauffer. "A Robust Multiplexer-Based FPGA Inspired by Biological Systems". *Journal of Systems Architecture: Special Issue on Dependable Parallel Computer Systems*, Vol. 43, No. 10, 1997.

[19] G. Tempesti, C. Teuscher. "Biology Goes Digital", *Xcell Journal*, No 47, Fall 2003, pp. 40-45.

[20] A.M. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.-M. Moreno, J. Rosenberg, A. Villa. "POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware". *From Biology to Hardware: Proc. 5th Int. Conf. on Evolvable Systems (ICES '03)*, LNCS 2606, Springer-Verlag, 2003, pp.129-140.

[21] J. von Neumann. *The Theory of Self-Reproducing Automata*. A. W. Burks, ed. University of Illinois Press, Urbana, IL, 1966.

[22] G. D. Watkins. "Novel Electronic Circuitry", Predictive Paper Reprint. *Proceedings of the IEEE*, Vol. 86, No. 11, November 1998, p. 2383.

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

[24] Y. Zorian. "Testing the Monster Chip". *IEEE Spectrum*, 36(7), 1999, pp. 54-60.