Bio-Inspired Computing Architectures: The *Embryonics* Approach

(Invited Paper)

Gianluca Tempesti, Daniel Mange, André Stauffer
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Email: gianluca.tempesti@epfl.ch

***Abstract*— The promise of next-generation computer technologies, such as nano-electronics, implies a number of serious alterations to the design flow of digital circuits. One of the most serious issues is related to circuit layout, as conventional lithographic techniques do not scale to the molecular level. A second important issue concerns fault tolerance: molecular-scale devices will be subject to fault densities that are orders of magnitude greater than silicon-based circuits.**

In our work, we are investigating a different approach to the design of complex computing systems, inspired by the developmental process of multi-cellular organisms in nature. This approach has led us to define a hierarchical system based on several levels of complexity, ranging from the molecule (modeled by an element of a programmable logic device when the system is applied to silicon) to the organism, defined as an application-specific multi-processor system.

By setting aside some of the conventional circuit design priorities, namely size and (to a certain extent) performance, we are able to design fully scalable systems endowed with some properties not commonly found in digital circuits. Most notably, by exploiting a hierarchical self-repair approach, our systems are able to tolerate higher fault densities, whereas a self-replication mechanism allows our arrays of processing elements to self-organize, greatly reducing the layout complexity of the system.

I. 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*) is inspired by the basic processes of molecular biology and by the embryonic development of living beings [5] [6] [7]. By adopting certain features of cellular organization, and by transposing them to the two-dimensional world of integrated circuits on silicon, we will 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 wish however to emphasize that the goal of bio-inspiration in the context of Embryonics is not the modelization or the explication of actual biological phenomena: our final objective is the development of very large scale integrated (VLSI) digital 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 artificial systems requiring improved reliability in short, medium, or long term applications.

- 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 the increased radiation levels reduce the reliability of components, or those which exploit the latest technological advances, and 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 [23].
- Medium term applications, where our aim is to develop very complex integrated circuits capable of on-line self-repair, dispensing with the systematic detection of faults at fabrication, impossible for systems consisting of millions of logic gates [22].
- Long term applications, executed on systems built with imperfect components: this is von Neumann's historical idea [19], the basis of all present projects aimed at the realization of complex integrated circuits at the molecular scale (nanoelectronics) [20] [3] [2] [15].

Self-replication, or "cloning", can be justified independently of self-repair:

- to replicate, within a field-programmable gate array (FPGA), functionally equivalent systems [13];
- to mass-produce the future integrated circuits, imple-

Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)
0-7695-2255-6/05 $20.00 © 2005 IEEE

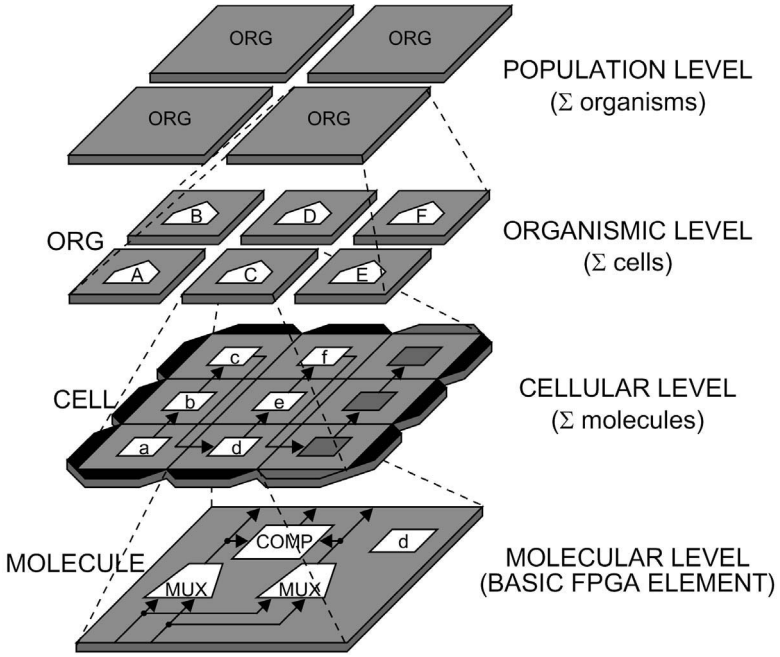

Fig. 1. The 4 hierarchical levels of complexity in Embryonics.

mented using nanoelectronics [10];

- 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 [19].

These emerging needs require the development of a new design paradigm that supports efficient online self-repair solutions and that can efficiently realize the self-replication of complex electronic structures. Our project is an investigation into the possibilities offered by drawing inspiration from complex biological organisms in order to address these issues.

II. Embryonics: 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 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.

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 minor (wound) or major (loss of an organ) trauma occur, living organisms are thus potentially capable of self-repair (cicatrization) or self-replication (cloning or budding) [21].

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, in order to obtain the properties of self-repair and self-replication.

To implement these features in silicon, 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 circuit.
- A finite set of molecules makes up a *cell*, essentially a small application-specific processor with the associated memory.
- A finite set of cells makes up 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.

At the core of the system is the cell: an artificial organism is realized by a matrix of identical cells distributed over the nodes of a regular two-dimensional grid. Each cell contains a small processor and a memory in which the *genome* program (the operating program for the entire system, identical for all the cells) is stored. In this multicellular organization only the state of a cell (i.e. the contents of its registers) can differentiate it from its neighbors.

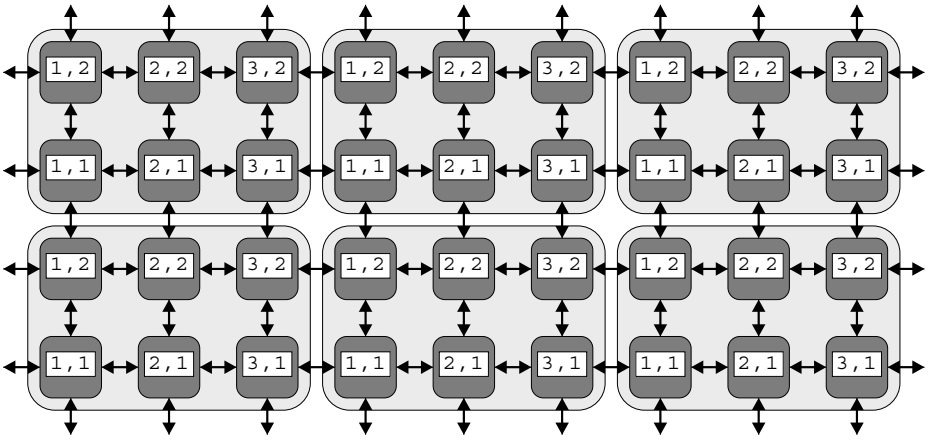

Fig. 2. Self-replication of a 6-cell organism in a limited homogeneous array of 9x4 cells.

In the organism each cell realizes a unique function, defined by a sub-program called the *gene*, which is a part of the genome. Each cell knows its position (i.e. X and Y coordinates) in the organism and uses them to define which gene of the genome it has to execute. Figure 2 shows an organism composed of $3x2 = 6$ cells: the genes are identified by the coordinates $(X, Y) = (1, 1)$ to $(X, Y) = (3, 2)$.

In this context, an artificial organism is capable of replicating itself if there is enough free space in the silicon circuit (at least six cells in the example of figure 2) to contain the new daughter organism and if the calculation of the coordinates produces a cycle ($X = 1 \rightarrow 2 \rightarrow 3 \rightarrow 1...$ and $Y = 1 \rightarrow 2 \rightarrow 1...$, implying $X = (WX + 1)modulo3$ and $Y = (SY + 1)modulo2$). Since each cell stores the same information (i.e. the genome program), the cycling of the coordinates causes the repetition of the same pattern of genes: therefore, in a sufficiently large array of cells, the self-replication process can be repeated for any number of specimens in the X and/or the Y axes.

This self-replication of the organism, achieved through the cycling of the cell's coordinates, is then an immediate consequence of the self-replication of the artificial cells. In fact, a cell has to self-replicate to obtain a collection of identical cells, which will compose the first artificial organism. The crucial hardware mechanism necessary to obtain populations of organisms is therefore the same as the one necessary to obtain a single multi-cellular organism.

III. Artificial Self-Replication

In each cell of every living being, the genome is translated sequentially by a chemical processor, the ribosome, to create the proteins needed for the organism's survival. The ribosome itself consists of molecules, whose description is an important part of the genome.

As mentioned, in the Embryonics project each cell is a small processor, sequentially executing the instructions of a first part of the artificial genome, the operative genome OG. The need to realize organisms of varying degrees of complexity has led us to design an artificial cell characterized by a flexible architecture, that is, itself configurable. It will therefore be implemented using a new kind of fine-grained, field-programmable gate array (FPGA). Each element of this FPGA (consisting essentially of a multiplexer associated with a programmable connection network) is then equivalent to a molecule, and an appropriate number of these artificial molecules allows us to realize application-specific processors.

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 the second part of our artificial genome: the *ribosomic genome* RG. Fig. 3 shows an abstract example of an extremely simple cell (CELL) consisting of six molecules, each defined by a molecular code or MOLCODE (CFG0 to CFG5), equivalent to the configuration information of a single element of the FPGA. The set of these six MOLCODEs constitutes the ribosomic genome RG of the cell.

The information contained in the ribosomic genome RG thus defines the logic function of each molecule 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;
- the presence of one or more spare columns, composed of spare molecules, required for the self-repair described below.

The definition of these pieces of information is the molecular configuration. Their injection into the FPGA will allow:

1) the definition of the function realized by each of the molecules;
2) the insertion of one or more spare columns;
3) the definition of the connections between the molecules.

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*.

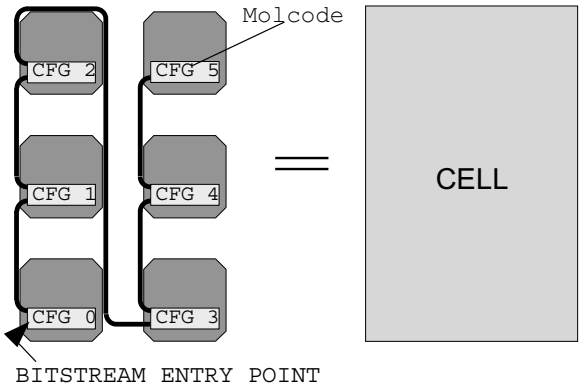

Fig. 3. Multimolecular organization of a simple cell consisting of 6 molecules. RG: ribosomic genome: CFG0 to CFG5. The arrow indicates the entry point for the configuration and the dark line is the path used to propagate this configuration to all the molecules.

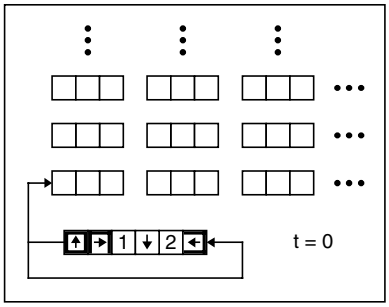

Fig. 4. The container with the genome of a minimal cell.

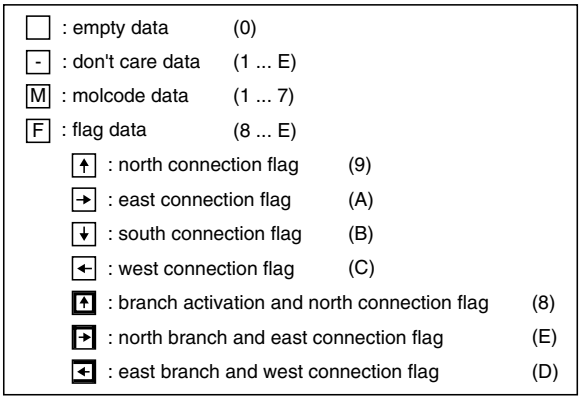

Fig. 5. Graphical and hexadecimal representations of the content symbols.

Cellular self-replication is a prerequisite for the cellular division 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 in order to create a copy of the genome from the mother cell aimed at programming the daughter cells (*genome transcription*).

Our developmental mechanism shall operate by allowing the set of molecular configurations (the MOLCODE of our artificial molecules) that implement a cell to replicate itself, realizing a process not unlike the cellular division that underlies the growth of biological organisms. We called this mechanism the *Tom Thumb algorithm* [8] [8].

A practical way to verify the realization of such novel mechanisms in the world of computer science is to approach the problem through the creation of an artificial Universe, defined by a *container*, a *content*, and a set of *rules*. We shall again use an abstract example by applying the algorithm to a very simple cell of $2x2$ molecules, with the further simplification that the MOLCODE of the cell will consists of only 3 bits. Note however that the algorithm is perfectly scalable for arbitrary cell and MOLCODE sizes.

The container is a two-dimensional flat space, divided in rows and columns (Fig. 4). Each intersection of a row and a column defines a rectangle or molecule, which divides in three memory positions: left, central, and right. Time flows in discrete clock times, the time steps, identified by integers (t=-1,0,1,2,...). In practice, this universe corresponds to the configuration layer of our FPGA, where the MOLCODE is shifted into the molecules at each clock cycle via a shift register.

The content of this Universe is constituted by a finite number of symbols, each represented by a hexadecimal character ranging from 0 to E, that is, from 0000 to 1110 in binary (Fig. 5). These symbols are either empty data (0), MOLCODE data (for molecule code data, M=1 to 7) or flag data, each indicating one of the four cardinal directions: north, east, south, west (F=8 to E). MOLCODE data will be used for configuring our final artificial organism (i.e., they correspond to the configuration data of our FPGA), while flag data are needed for constructing the skeleton of the cell. The original genome for the minimal cell is organized as a string of six hexadecimal characters, i.e. half the number of characters in the cell, moving counterclockwise by one character at each time step (t=0,1,2,...).

The set of rules defines the behavior of the content of the Universe. It is defined by a set of 9 rules, used to construct the cells and to implement cellular division (growth). Even if they are too complex to be detailed in this article, we shall mention that the rules are *local*, that is, they rely only on the information stored on neighboring molecules. This feature, required to reduce the need for global synchronicity, addresses an important issue in very large scale digital circuits. The net results of the application of the rules on the container is the ability, in the first place, to construct cells of arbitrary size by chaining their MOLCODES into the shape of a loop (Fig. 6) and in the second place to realize the self-replication of these patterns into the empty space. This process corresponds to the self-replication of our cells and thus implements the functionality required to realize cellular division and, eventually, the self-replication of an organism.

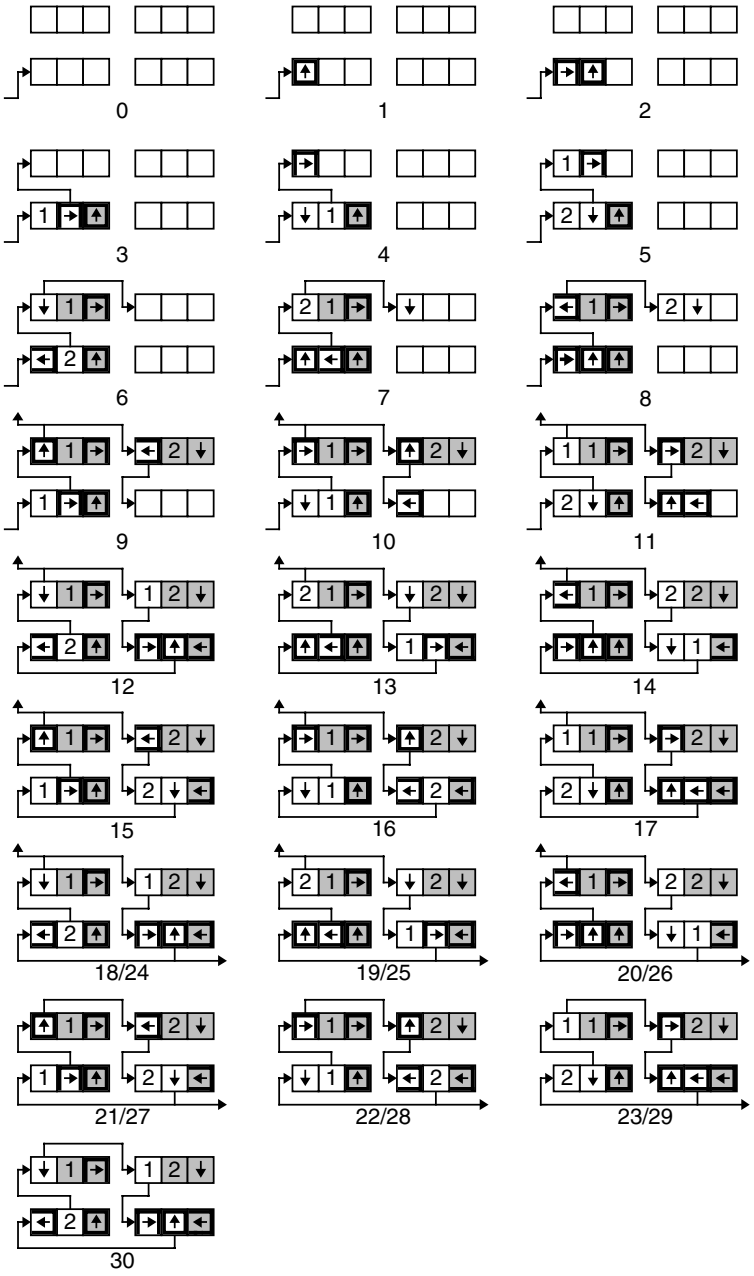

Fig. 6. Construction of the simple cell in the artificial universe.

IV. Self-Repair

There exist a number of well-known approaches to implementing self-repair in two-dimensional arrays of identical elements [1] [4] [11] (we shall not address the issue of self-test in this article). Most rely on two mechanisms: since physically repairing a hardware fault is impossible, we must provide a set of spare elements (redundancy) and a way to let them replace faulty elements in the array, that is, to reroute the connections between the elements (reconfiguration). The self-repair system we developed is no exception, even if it had to satisfy a set of relatively non-standard constraints imposed by the unique features of our systems.

The key observations that relates to self-repair in our systems is that test and reconfiguration are exclusively *local, on-line mechanisms* and that they operate *on all levels*. The first property implies that there is no centralized control, which would not scale well oven arbitrary sizes, and that the repair occurs as the system is working. The second property, crucial for very complex systems, addresses the limitations that are inherent to any self-repair approach: a single level of redundancy is not sufficient to guarantee adequate fault-tolerance and by combining mechanisms that operate at different levels of complexity (organism, cell, molecule) we can obtain systems that are more reliable than would otherwise be possible.

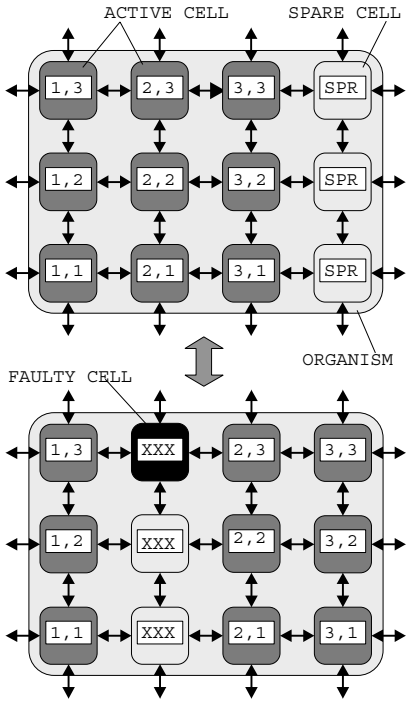

Fig. 7. The presence of spare cells in the array and of the complete genome in each cell allows self-repair through a recomputation of the coordinates.

This multi-level approach is perfectly in keeping with biological inspiration: nature guarantees the survival of organisms by operating on systems which range from the population level (where the genetic information of a species is stored in each individual) through the organism level (where cicatrization allows the substitution of dead cells by others, thanks to the presence of a complete copy of the genome in every cell) to the molecular level (where complex biochemical mechanisms guarantee the correctness of the operation of each single cell).

Given our setup, the redundancy of the population level is, in a way, "free": if sufficient space exists in the system, self-replication will automatically create multiple copies of the entire organism, each containing the genome of the "species". We have not yet implemented mechanisms to introduce mutations from one individual to the next (a difficult problem when dealing with electronics), which means that we obtain populations of *clones*, but the redundancy is still present and can be exploited for fault-tolerance purposes.

For the other levels, cellular and molecular, we have on the other hand introduced some specific mechanisms to add self-repair in our systems.

A. Self-Repair at the Cellular Level

While not "free" as for the organisms, self-repair at the cellular level is greatly simplified by the structure of our systems. In fact, since each cell, as we mentioned, contains a copy of the genome (the executable program) of the entire system, it is in theory capable of replacing any other cell (a bit like *stem or undifferentiated cells* [14] in biological organisms.

Since the gene to be execute in a cell depends on its coordinates within the organism, in order to reconfigure the array it is sufficient to change the coordinates of the cells (Fig. 7). For this kind of mechanism to work, it is of course necessary to specify a set of *spare cells*, that is, cells that are not active during the normal operation of the array, but are ready to take the place of faulty cells. Our bio-inspired approach provides a very simple solution to this problem, since it is possible to define a special gene that will instruct the cells to be at the organism's disposal in case of faults.

To reduce the complexity of the mechanisms involved, we opted for a simplified approach where a dead cell will propagate its fault to its entire column within the organism. This approach, while of course wasteful in resources, greatly simplifies the reconfiguration of the system (Fig. 7):

1) a set of spare columns (as many as desired) are introduced to the right of the array by modifying the growth pattern of the organism;
2) whenever a fault is detected in a cell, the column of cells to which it belongs is deactivated and becomes transparent to the array, leaving a "scar" in the array;
3) the disappearance of the faulty column automatically launches the recomputation of the coordinates throughout the array;
4) the recomputation causes a right-shift of the coordinates of the faulty column and of all the columns to its right, until the first spare column is reached.

Once this process is finished, the array is ready to resume operation.

It should however be mentioned that, in order to achieve online self-repair and to allow the system to resume operation from the state it had when the faulty cell was detected, another mechanism is necessary to recover the state of the faulty cell. We have in fact implemented and tested one such mechanism, based on redundant computation (the same data was computed in parallel by two cells) and we have shown that this kind of fault-tolerance is indeed achievable without too much trouble thanks, once again, to the bio-inspired approach.

B. Self-Repair at the Molecular Level

The molecular layer presents problems that are somewhat different from the cellular layer, since this layer represents the actual hardware of our system and is thus most closely tied to technological issues which render biological inspiration difficult: self-repair in an FPGA is very architecture-dependent, and the architecture of programmable logic circuits changes quickly. We have ourselves changed the basic structure of our molecular layer at least twice [16] [18], requiring a redefinition of the mechanisms involved in self-repair. In this section, we will then limit ourselves to outlining our approach, without entering into the details of the implementation.

In general, the FPGAs we developed in our project have the characteristic of being perfectly *homogenous*, obviously simplifying the self-repair process. This process relies, as for the cellular layer, on reconfiguration and exploits the presence of columns of spare molecules.

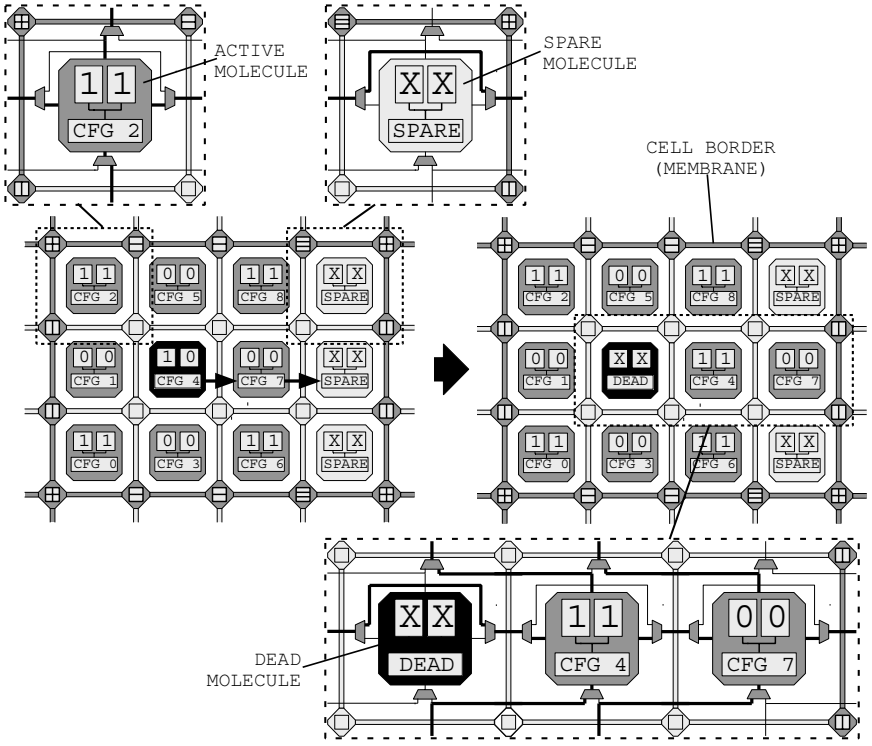

Fig. 8. The information stored in a faulty element and in its neighbors is shifted to a spare column.

To find an efficient mechanism to implement redundancy at the molecular level, we turned our attention back to the self-replication mechanism. Since the role of this mechanism is to assign a specific function to each of the molecules in the cell, it is possible to modify it to define which of the molecules will act as spare elements. We obtain then a very powerful system, as this approach allows us to program the robustness of the system. In fact, since the growth sequence is part of the configuration of the FPGA, we can modify the frequency of spare columns, and thus the fault tolerance of the system. Just by changing the control states and without altering the actual configuration data (an advantage, since generating a bitstream can be a time-consuming process), we can introduce varying degrees of redundancy, from zero (no spare columns) to 100% (one spare for every active column).

To take advantage of the spare elements and to realize online self-repair, we also require a mechanism to transfer the information stored in a faulty element (its configuration plus the value stored in its flip-flops) to one of the spare elements. Our mechanism for repairing faults (Fig. 8) relies on the reconfiguration of the network through the replacement of the faulty element by its right-hand neighbor: the configuration of the faulty element (the dead molecule) is shifted into the neighbor. The configuration of the neighbor is itself shifted to the right, and so on until a spare element is reached. Once the shift is completed, the faulty element ”dies” with respect to the network: the connections are rerouted to avoid it, an operation which can be effected very simply by diverting the north-south connections to the right and by rendering the element transparent to the east-west connections. The array, thus reconfigured and rerouted, can then resume executing the application from the same state it held when the fault was detected. When a fault is detected, the FPGA therefore goes off-line for the time required by the reconfiguration, somewhat like an organism becoming incapacitated during an illness, and then resumes operation.

Like all such mechanisms, our molecular self-repair is subject to failure, either because of saturation (if all spare elements are exhausted) or because a non-repairable fault is detected. Should such a failure occur, we need to activate the self-repair mechanism at the cellular level (described above). To this end, we designed a KILL signal that is propagated through an entire column of cells, deactivating it. At the cellular level, this event will trigger a recomputation of the coordinates of all cells, that is, will activate the cellular-level reconfiguration mechanism (Fig. 9). In other words, the robustness of the system is not based on a single self-repair mechanism, which might fail under extreme conditions, but rather on two separate mechanisms which cooperate to prevent a fault from causing a catastrophic failure of the entire system.

V. Conclusions

In this article, we presented an overview of our Embryonics project, in which we try to draw inspiration from the structure and operation of multicellular organisms in nature to develop a new approach for the design of very complex digital systems in silicon and beyond. It is an ongoing project and we are still researching novel solutions along all the research axes that are touched by the project.

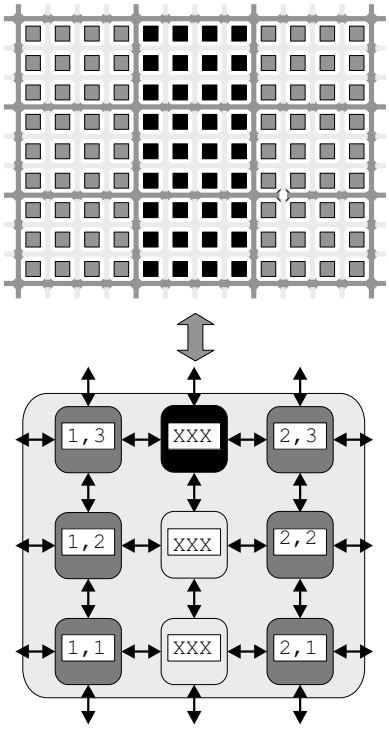

Fig. 9. The death of a column of blocks at the molecular level is equivalent to the death of a column of cells at the cellular level.

The Tom Thumb algorithm, for example, represents a new development in the field of self-replication. We are currently working to move the algorithm from its theoretical implementations as a self-replicating inert structure towards an actual realization as the configuration mechanism for the FPGA we recently developed in a EU-sponsored project [18].

Where self-repair is concerned, we are working on more versatile reconfiguration mechanisms that exploit the growth characteristics of our systems at the processor level. We are trying to show how we can achieve a higher fault tolerance in our systems by drawing inspiration from the hierarchical systems present in nature. Through well-applied redundancy and by setting up cooperation between the different layers of our system, we are exploring the possibility of automatically integrating fault tolerance in arrays of processors.

Finally, we would like to mention that all the systems we have developed have been implemented and tested in actual hardware, often exploiting the *BioWall* [17], a wall-sized platform of reconfigurable logic we have developed and built to prototype cellular systems. In another direction, we are working on the development of a design environment to automatically attach bio-inspired features to arrays of application-specific processors.

REFERENCES

[1] F. Hanchek, S. Dutt. Methodologies for Tolerating Cell and Interconnect Faults in FPGAs. *IEEE Transactions on Computers*, Vol.47, No.1, January 1998.

[2] 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.

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

[4] J. Lach, W.H. Mangione-Smith, M. Potkonjak. Efficiently Supporting Fault-Tolerance in FPGAs. *Proc. FPGA'98*, Monterey, CA, February 1998, pp. 105-115.

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

[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, and G. Tempesti. "Towards Robust Integrated Circuits: The Embryonics Approach". *Proceedings of the IEEE*, Vol.88, No.4, 2000, pp.516-541.

[8] D. Mange, A. Stauffer, E. Petraglio, and G. Tempesti. "Embryonic Machines That Divide and Differentiate". In A.J. Ijspeert, M. Murata, and N. Wakamiya, Eds., *Biologically Inspired Approaches to Advanced Information Theory*, LNCS 3141, Springer-Verlag, Berlin, 2004, pp. 201-216.

[9] D. Mange, A. Stauffer, E. Petraglio, and G. Tempesti "Self-Replicating Loop with Universal Construction". *Physica D*, Vol.191, No.1-2, 15 April 2004, pp. 178-192.

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

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

[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 International Symposium on Field Programmable Gate Arrays (FPGA'99)*, Monterey, CA, February 1999, pp. 81-89.

[14] H. Pearson. The Regeneration Gap. *Nature*, Vol.414, 2001, p.388.

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

[16] 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.

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

[18] 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". *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES '03)*, LNCS 2606, Springer-Verlag, Berlin, 2003, pp.129-140.

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

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

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

[22] Y. Zorian. "Testing the Monster Chip". *IEEE Spectrum*, Vol. 36, No. 7, July 1999, pp. 54-60.

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