POETIC

IST-2000-28027

Reconfigurable POEtic Tissue

Periodic Progress Report No: 4

Covering period 1.9.2001-28.2.2002

Report Originator: Gianluca TEMPESTI

Report Version: 1.2

Report Preparation Date: 28/02/2002

Classification:

Project start: September 1, 2001 **Duration:** 36 months**.**

Project Co-ordinator: Professor AM Tyrrell

Partners: University of York, Technical University of Catalunya, University of Glasgow, University of Lausanne, Swiss Federal Institute of Technology Lausanne.

Project funded by the European Community under the "Information Society Technologies" Programme (1998-2002)

WP 2 – Deliverable 4

Evaluation of Ontogenetic Methods in Digital Systems

G. Tempesti

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).

This ontogenetic process is not, of course, unique to human beings. In fact, it applies to the majority of living beings, with the exception of unicellular organisms such as viruses and bacteria. The “architecture” used by most living beings, to use a term familiar to computer science, is based on *multicellular organization*, which 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, and all cells contain all the genetic material (the *genome*) of the organism.

Multicellular organization relies on two basic mechanisms:

1) *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).
2) *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 on the physical position of the cell in the organism.

A consequence of these 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) [1].

To our knowledge, the only research groups trying to explicitly draw inspiration from the ontogenetic process are the Swiss Federal Institute of Technology at Lausanne, Switzerland [3][5][6], and the University of York, UK [7][8], two of the participants in this project. This limited amount of research is both surprising and *un*surprising.

It is surprising because some properties of biological organisms, namely self-replication and self-repair, stem directly from their multicellular organization, and could be extremely interesting for VLSI circuits, where self-repair would allow partial reconstruction in case of minor faults, and self-replication the complete reconstruction of the original device in case of major faults. These two properties are particularly desirable for complex systems requiring reliability in short, medium, or long term applications.

1) Short term applications [2]:
   - Applications that require very high levels of reliability, such as avionics or medical electronics.
   - Applications designed for hostile environments, such as space, where the increased radiation levels reduce the reliability of components.
   - Applications that exploit the latest technological advances (drastic device shrinking, lower power supply levels, and increasing operating speeds) that accompany the evolution to deeper and deeper submicron levels and

significantly reduce the noise margins and increase the soft-error rates [10].

2) Medium term applications, implying the development of very complex integrated circuits capable of on-line self-repair, dispensing with the systematic detection of faults at fabrication, impossible for systems consisting of hundreds or thousands of millions of logic gates [11].
3) Long term applications, executed on systems built with imperfect components: this is von Neumann's historical idea [4], the basis of all present projects aimed at the realization of complex integrated circuits at the atomic scale (nanotechnologies) [12][13][14][15].

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

- to replicate, within a field-programmable gate array (FPGA), functionally equivalent systems [16];
- to produce the massive quantity of future integrated circuits, implemented using nanotechnologies [9];
- 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 [4].

But the lack of research along the ontogenetic axis is not so surprising when considering the *requirements* of a truly ontogenetic machine:

- Cellular division, and by implication *growth*, would require that the *physical* structure of the organism be altered. Such a feat is not yet possible, given current technology (but the domain of nanotechnologies seems to promise that it *will* become possible in the medium to long term). However, the use of reprogrammable logic circuits (for example, FPGAs, or Field Programmable gate Arrays) does allow a kind of growth, if not in the physical sense, at least in the *functional* sense.
- Cellular differentiation, once again, would require that the *physical* structure of the cells be altered to suit their intended function. Again, until nanotechnologies or another similar technological breakthrough will allow this kind of manipulation of silicon circuits, FPGAs can allow a *functional* adaptation of the cells to their functions.

- Finally, the massive amount of redundancy present in all biological systems and implied in any bio-inspired multi-cellular system (e.g., the presence of the entire genome in each cell) is very expensive in terms of “wasted” resources. On the other hand, the increase in the amount of transistors that can be placed on a single VLSI is constantly eroding the price of digital logic, and thus of redundancy. Moreover, we will soon reach the point where the fabrication of perfect components will become more and more arduous, increasing the demand for fault-tolerant systems, even at the price of increased redundancy.

To resume the concept of ontogenetic machines, we can identify (arbitrarily, of course, but in accordance with much of the research in the domain) three key aspects of the ontogenetic process, which will have to be handled throughout the POEtic project:

- *Morphogenesis*, that is, the formation of the organism as a structure;
- *Multicellular organization*, that is, the processes of cellular division and cellular replication giving rise to properties such as self-repair;
- *Genotype-phenotype mapping*, that is, how the machine can be described in a *useful* way, a key aspect for any bio-inspired machine.

II. Morphogenesis: Generation of Form

Biological development is a highly complex, hierarchical program that operates across many different scales and at each level of scale many different mechanisms and self-organizing processes are being involved. The dynamic interactions between the different mechanisms and elements result in a highly complex system. Development in artificial systems is a very important issue and it is commonly believed that it is a key to the generation of highly complex systems (see for example [17]).

Much of the early work in developmental modeling focused on modeling a particular mechanism only. The mechanism has then be considered in isolation and not in interaction with other mechanism, although it has long been evident that the interactions of different mechanisms are critical to several issues, like for example pattern formation. Today, there is a growing trend towards developmental models based on multiple mechanisms (like cell migration, cell division, attraction forces, environmental influences, etc.). In the following, some of the main models will be presented.

Besides D'Arcy Thompson's classical work “On Growth and Form” [18], first published in 1917, Alan Turing was one of the first persons interested in modeling morphogenesis. In his 1952 seminal paper entitled “The Chemical Basis of Morphogenesis” [19][20] he proposed a mathematical theory of cell-cell interaction via chemical substances (also called *reaction-diffusion model*). Turing described several sets of differential equations that governed the interactions between different substances that diffuse and react with each other. The original intent of the reaction-diffusion model was to explain the “breakdown of symmetry and homogeneity” or the emergence of a pattern in an originally homogeneous medium.

In 1968, Aristid Lindenmayer developed a grammar-based technique called *L-systems* [21] (further developed later by Prusinkiewicz [22]). L-systems is a rule rewriting formalism that allows to efficiently describe growth, e.g., plant growth. For a particular L-system, the growth always starts from the same seed cell, called *axiom*. *Production rules* are used to describe the growth of new cells from the old cells. L-systems are often used to model development in combination with evolutionary algorithms and neural networks, are computationally feasible models, but not well suited for models based on local interactions only.

Kitano, for example, uses a graph-generation grammar based on L-systems where the structure of the network is not directly encoded on the genome [23]. His approach shows better scaling properties, is more biologically plausible, and generates more complex networks, but it is not always easy to determine whether a grammar-based encoding can express every possible network architecture. Another important grammar-based approach comes from Gruau [24][25][26]. He developed a special encoding scheme called *cellular encoding*. Instead of rewriting characters, the rewriting grammar directly rewrites neurons. A genetic algorithm can then be used to find a grammar tree suitable for a given problem. In order to speed up the search for functional networks, Gruau also included several forms of learning in his model (see for example [25]).

Odell et al. [27] studied in 1981 the mechanical aspects of morphogenesis. They presented a model for the folding and movement of cell sheets and applied it to certain elementary types of gastrulation and neurulation. Work with mechanical models has been continued later with more detailed and more complex models. Cells have different shapes and various forces are applied. Weliky and Oster [28], for example, use cells that have a polygonal shape in three dimensions. Cells are subjected to forces that are due to osmotic pressure and to the elastic membrane. To probe further: Fleischer [29] provides an overview on developmental modeling in his Ph.D. thesis.

Membrane computing (also called *P-systems*) is a new and emerging branch of computing initiated by Paun [30]. A P-system, which is related to various areas such as L-systems, formal language theory, and multiset processing, is a computing model that highly abstracts from the way the alive cells process chemical compounds in their compartmental structure. Multisets of objects are placed in compartments defined by the membrane structure and the objects evolve by means of *reaction rules* associated with each compartment. Objects are described by symbols or by strings of symbol and can pass through membranes that can change their permeability. Membranes can also be created, dissolved and divided. By using the reaction rules in a non-deterministic, maximally parallel manner, one gets transitions between the system configurations. Many P-systems are computationally universal, i.e. equal in power to Turing machines. Membrane computing has principally been motivated by mathematics and not because it is a plausible biological model. Nevertheless, it is a powerful and efficient approach that can quite realistically model many biochemical reactions of biological cells, developmental processes, cell replication, cell division, etc. To the best of our knowledge, P-systems have never been implemented in hardware.

More recently, the *Embryonics project* (embryonic electronics), an undergoing project the Swiss Federal Institute of Technology at Lausanne, Switzerland [3][5][6], and at the University of York, UK [7][8], implements quasi-biological development in real hardware. Morphogenesis is an important feature in the development process of the Embryonics machines, but is interpreted more from the point of view of an hardware implementation that that of a biological inspiration. At the core of the morphogenesis of these machines is a simple cellular automaton, and only the simplest structures (two-dimensional arrays of identical cells) can be realized. Because of the interest of its *other* features, notably cellular differentiation and self-repair, the project will be treated in detail in the next section.

III. Multi-Cellular Organization: The Embryonics Project

The morphogenetic approaches discussed above have one important drawback with respect to the POEtic project: they concentrate, as the name says, on the definition of the *shape* of an organism, ignoring another, fundamental aspect, the *function*. The one exception to this rule is the Embryonics project, whose main goal is the design of highly-robust integrated circuits, endowed with the properties usually associated with the living world: self-repair and self-replication, and capable of executing any given application through a multiprocessing (multi-cellular) approach. To the best of our knowledge, Embryonics is the only project explicitly dealing with the ontogenetic development of complex computation-capable machines (with the exception of ontogenetic artificial neural networks, which will be investigated and analyzed later in the POEtic project).

As a consequence, we will use the above-mentioned Embryonics (*Embryonic Electronics*) project as a basis for a more general discussion of ontogenetic machines, trying to identify the salient points that should be considered necessary for any kind of machine that draws inspiration from the ontogenetic processes of living beings.

A. From Biology to Hardware

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 (Fig. 2.1): 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.

The artificial organism will ultimately be divided into cells, themselves decomposed into molecules, a structure which determines the plan of this Section: Subsection B describes the three fundamental features of the organism (multicellular organization, cellular differentiation, and cellular division), while in Subsection C it is demonstrated that the organism, thanks to these three features, exhibits the two sought properties (self-replication and self-repair). Subsection D describes the cells and presents their essential features (multimolecular organization, molecular configuration, and molecular error detection), while Subsection E shows that the two sought properties (self-replication and self-repair) apply both at the cellular level as well as at the organismic level. Subsection F resumes the four organizational levels of the hierarchy of the Embryonics project.

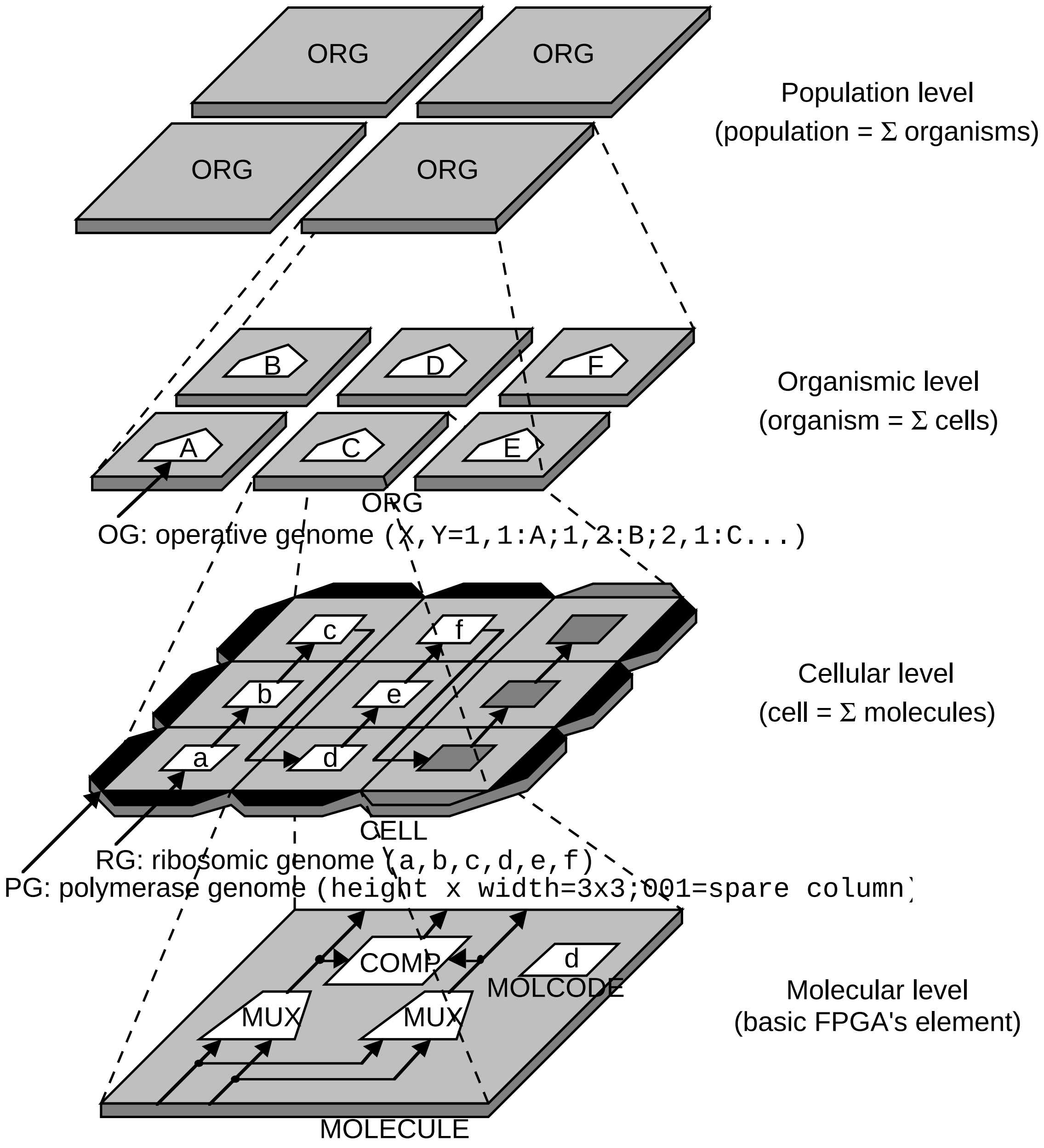


Fig. 2.1 The Embryonics landscape: a 4-level hierarchy.

B. The Organism's Features: Multicellular Organization, Cellular Differentiation, and Cellular Division

The environment in which this quasi-biological development occurs 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. Since such cells (small processors and their memory) have an identical physical structure (i.e., an identical set of logic operators and of connections), the cellular array is homogeneous. As the program in each cell (our artificial genome) is identical, only the state of a cell (i.e., the contents of its registers) can differentiate it from its neighbors.

Each of the cells (CELL) of an organism (ORG) 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. 2.2, the genes are labeled A to F for coordinates X,Y=1,1 to X,Y=3,2). Our final artificial genome will be divided into three main parts: the *operative genome* (OG), the *ribosomic genome* (RG), and the *polymerase genome* (PG). Let the operative genome (OG) be 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. 2.2 is then a graphical representation of organism ORG's operative genome.

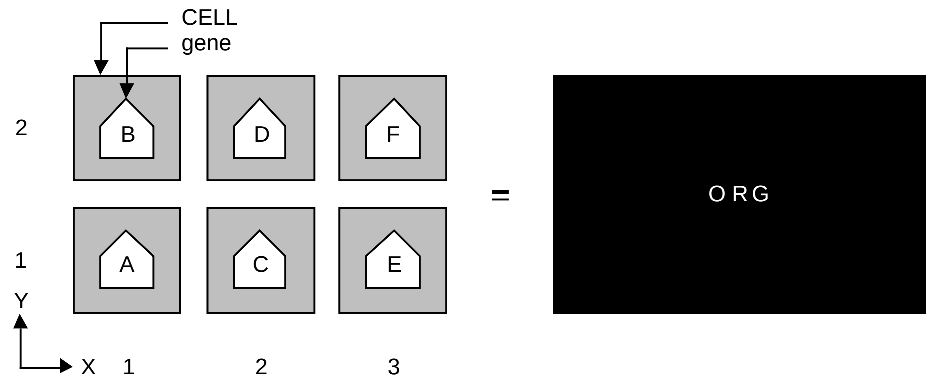
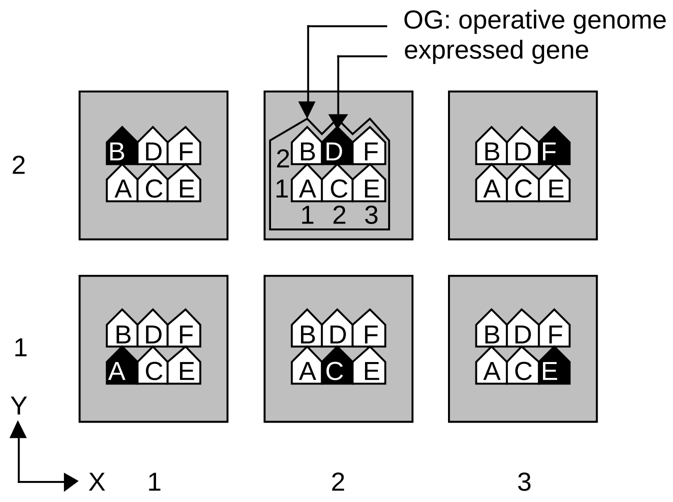


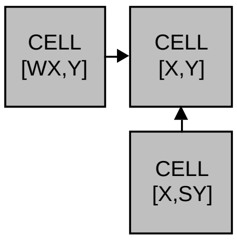
Fig. 2.2 Multicellular organization of a 6-cell organism ORG.

Let then each cell contain the entire operative genome OG (Fig. 2.3a): 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. In summary, storing the whole operative genome in each cell makes the cell universal:

given the proper coordinates, it can execute any one of the genes of the operative genome and thus implement *cellular differentiation*. In our artificial organism, any cell `CELL[X,Y]` continuously computes its coordinate `X` by incrementing the coordinate `WX` of its neighbor immediately to the west (Fig. 2.3b). Likewise, it continuously computes its coordinate `Y` by incrementing the coordinate `SY` of its neighbor immediately to the south.



(a)



(b)

Fig. 2.3 Cellular differentiation. (a) Global organization. (b) A cell `CELL[X,Y]` with its west neighbor `CELL[WX,Y]` and its south neighbor `CELL[X,SY]`; `X=WX+1`; `Y=SY+1`.

At startup, the first cell or *zygote* (Fig. 2.4), 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 `ORG` are completely programmed (in our example, the farthest cell is programmed after time `t3`).

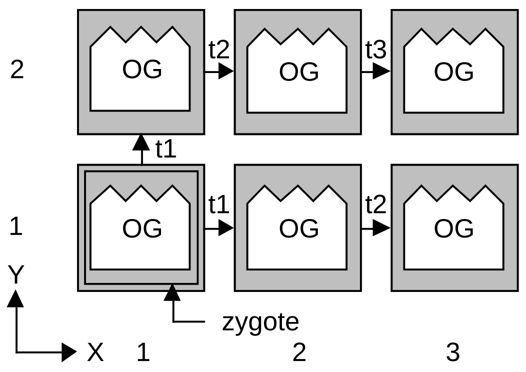


Fig. 2.4 Cellular division; `OG`: operative genome; `t1` ... `t3`: three cellular divisions.

C. The Organism's Properties: Organismic Self-Replication and Organismic Self-Repair

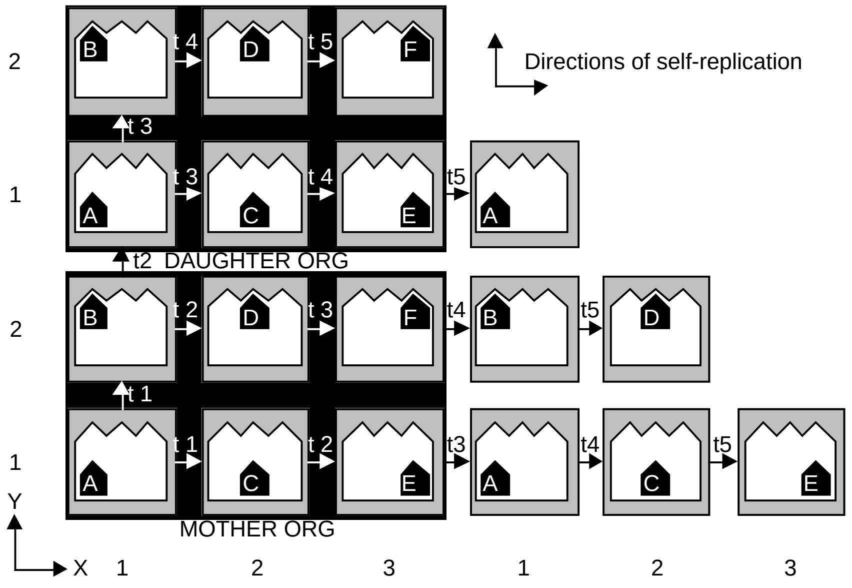


Fig. 2.5 Self-replication of a 6-cell organism `ORG` in a limited homogeneous array of 6x4 cells (situation at time `t5` after 5 cellular divisions); `MOTHER ORG` = mother organism; `DAUGHTER ORG` = daughter organism.

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

1) there exists a sufficient number of spare cells in the array (at least six in the example of Fig. 2.5) to contain the additional organism;
2) the calculation of the coordinates produces a cycle (`X=1`$\rightarrow$`2`$\rightarrow$`3`$\rightarrow$`1`... and `Y=1`$\rightarrow$`2`$\rightarrow$`1`... in Fig. 2.5, implying `X=(WX+1) modulo 3` and `Y=(SY+1) modulo 2`).

As the same pattern of coordinates produces the same pattern of genes, self-replication can be easily accomplished if the program of the operative genome `OG`, associated with the homogeneous array of cells, produces several occurrences of the basic pattern of coordinates. In our example (Fig. 2.5), the repetition of the vertical coordinate pattern ($Y=1\rightarrow2\rightarrow1\rightarrow2$) in a sufficiently large array of cells produces one copy, the *daughter organism*, of the original *mother organism*. Given sufficient space, the self-replication process can be repeated for any number of specimens in the X and/or the Y axes.

To implement the *self-repair of the organism*, we decided to use spare cells to the right of the original organism (Fig. 2.6). 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 Subsection E 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. 2.6). 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 (two spare columns, tolerating two successive faulty cells, in the example of Fig. 2.6). It also implies that the cell needs to be able to bypass the faulty column and to divert to the right all the required signals (such as the operative genome and the X coordinate, as well as the data busses). Given a sufficient number of cells, it is obviously possible to combine self-repair in the X direction, and self-replication in both the X and Y directions.

D. The Cell's Features: Multimolecular Organization, Molecular Configuration, and Molecular Fault Detection

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 a major 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 FPGA.

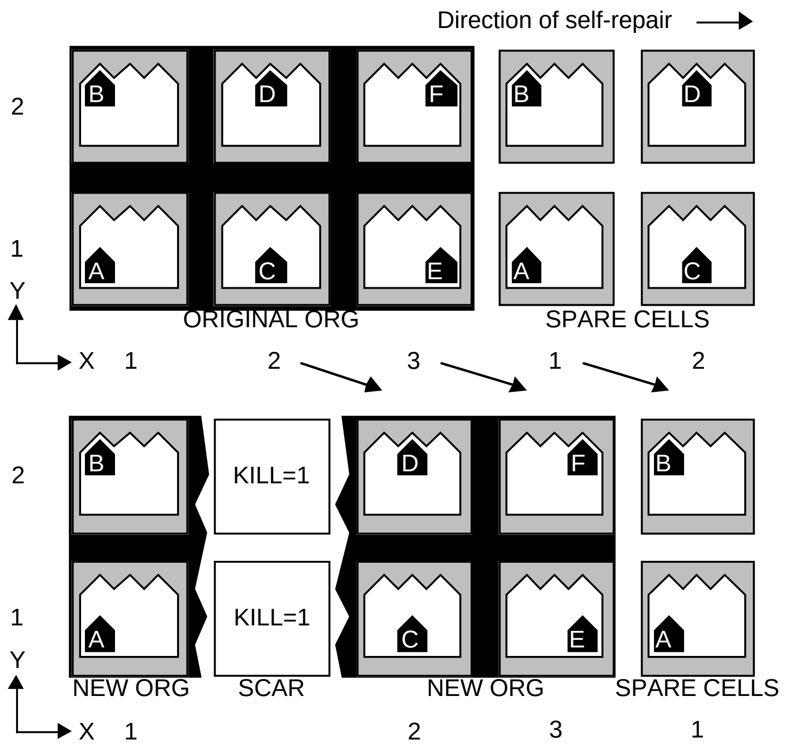
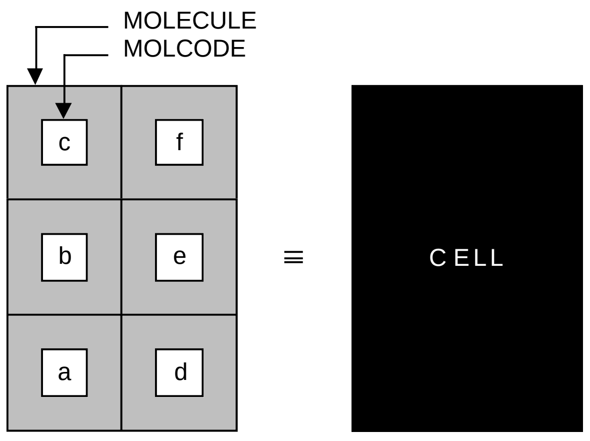


Fig. 2.6 Organismic self-repair.

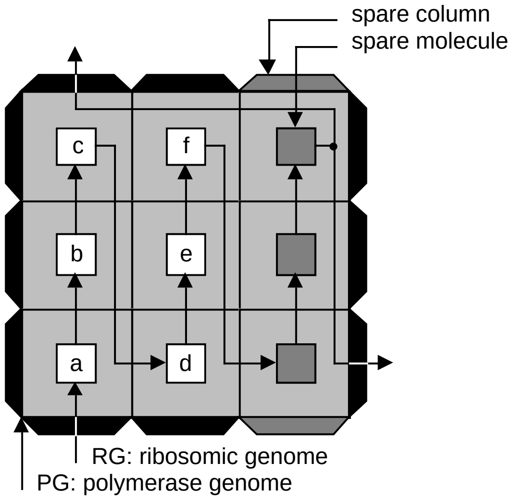
Each element of this FPGA (consisting essentially of a multiplexer associated with a programmable connection network) is then equivalent to a *molecule*, and an appropriate number of these artificial molecules allows us to realize application-specific processors. We will call *multimolecular organization* the use of many molecules to realize one cell. The configuration string 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. 2.7a shows a generic and abstract example of an extremely 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 by assigning a molecular code `MOLCODE` to it. To obtain a functional cell, we require two additional pieces of information:

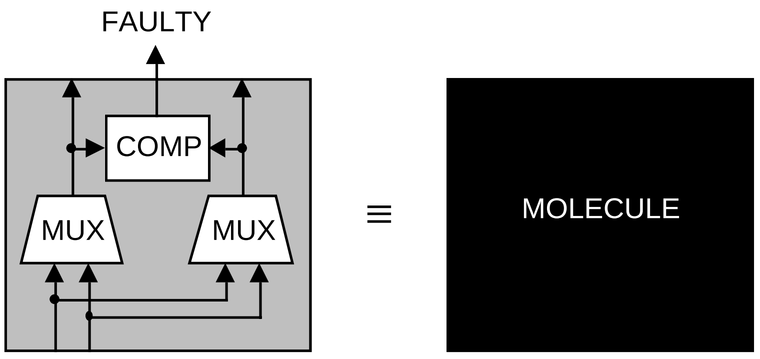
1) the *physical position* of each molecule in the cellular space;
2) the presence of one or more *spare columns*, composed of *spare molecules*, required for the self-repair described below (Subsection E).



(a)



(b)



(c)

Fig. 2.7 The cell's features. (a) Multimolecular organization; `RG`: ribosomic genome: `a, b, c, d, e, f`. (b) Molecular configuration; `PG`: polymerase genome: height x width = 3x3; 001 = spare column. (c) Molecular fault detection; `MUX`: multiplexer; `COMP`: comparator.

The definition of these pieces of information is the *molecular configuration* (Fig. 2.7b). Their injection into the FPGA will allow:

1) the creation of a border surrounding the molecules of a given cell;

2) the insertion of one or more spare columns;
3) the definition of the connections between the molecules, required for the propagation of the ribosomic genome `RG`.

The information needed for the molecular configuration (essentially, the height and width of the cell in number of molecules and the position of the spare columns) makes up the third and last part of our artificial genome: the *polymerase genome* `PG` (a terminology which will be justified in Subsection E).

Finally, it is imperative to be able to automatically detect the presence of faults at the molecular level and to relay this information to the cellular level. Moreover, if we consider that the death of a column of cells is quite expensive in terms of wasted resources, the ability to repair at least some of these faults at the molecular level (that is, without invoking the organismic self-repair mechanism) becomes highly desirable. The biological inspiration for this process derives from the DNA's double helix, the physical support of natural genomes, which provides complete redundancy of the genomic information though the presence of complementary bases in the opposing branches of the helix. By duplicating the material of each molecule (essentially the multiplexer `MUX`) and by continuously comparing the signals produced by each of the two copies (Fig. 2.7c), it is possible to detect a faulty molecule and to generate a signal `FAULTY=1`, realizing the *molecular fault detection* which will make possible cellular self-repair (described below in Subsection E).

E. The Cell's Properties: Cellular Self-Replication and Cellular Self-Repair

A consequence of the multimolecular organization and of the molecular configuration of the FPGA (Subsection D and Fig. 2.7b) is the ability, for any given cell, to propagate its polymerase genome `PG` and 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 (Fig. 2.8), thus implementing *cellular self-replication*. Cellular self-replication is a prerequisite for cellular division at the organismic level described above (Subsection B and Fig. 2.4), during which the operative genome is copied from the mother cell into the daughter cells. In living systems, a specific molecule, the *polymerase enzyme*, allows cellular replication through the duplication of the genome. It is by analogy to this enzyme that the third part of our artificial genome is called polymerase genome.

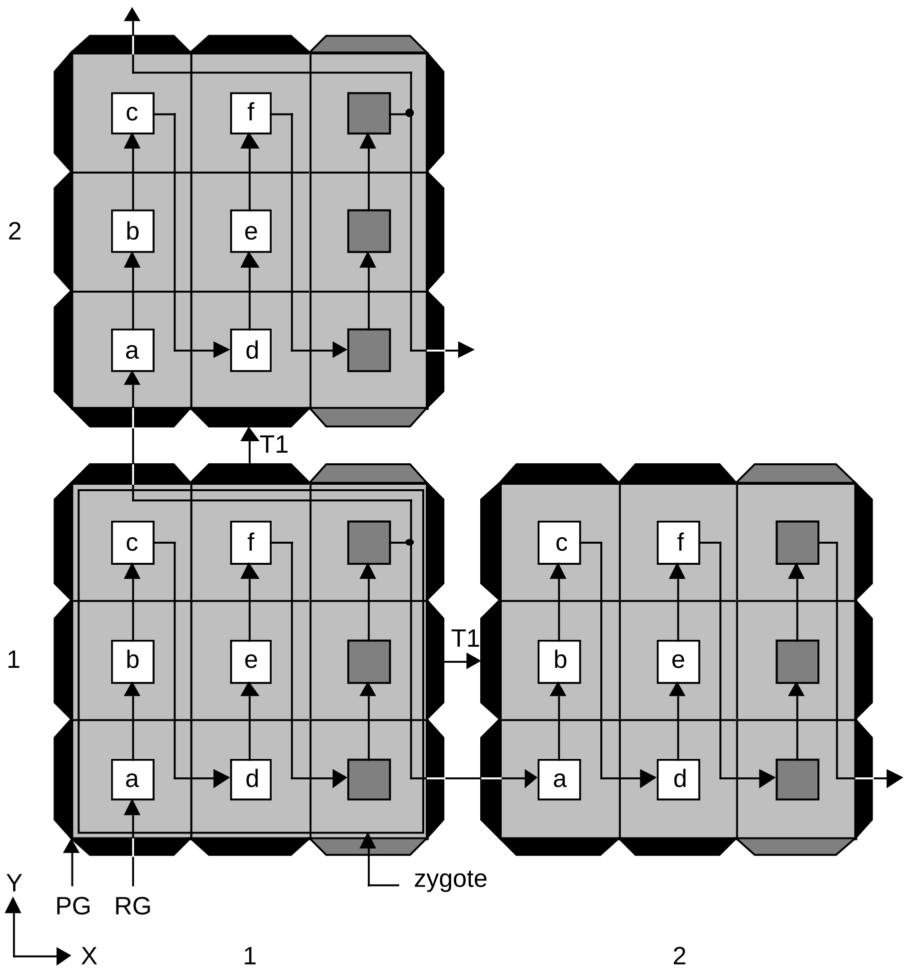
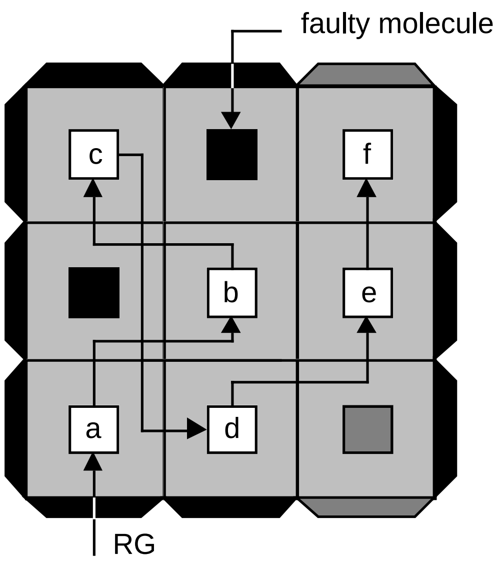
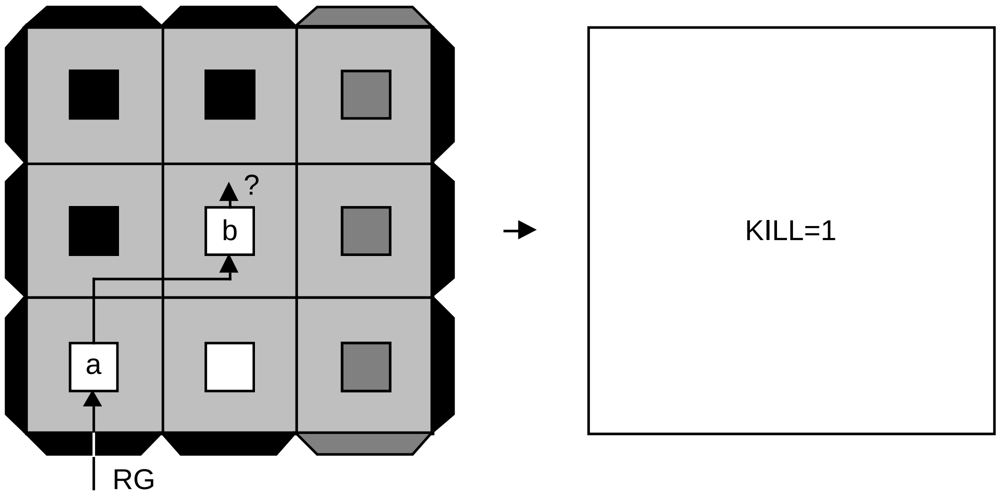


Fig. 2.8 Cellular self-replication; `T1`: one instance of cellular self-replication; `RG`: ribosomic genome; `PG`: polymerase genome.

The presence of spare columns, defined by the molecular configuration, and the automatic detection of faulty molecules (Subsection D, Figs. 2.7b and 2.7c) allow *cellular self-repair*: each faulty molecule is deactivated, isolated from the network, and replaced by a neighboring molecule, which will itself be replaced by a neighbor, and so on until a spare molecule (`SM`) is reached (Fig. 2.9a).



(a)



(b)

Fig. 2.9 Cellular self-repair. (a) Possible self-repair (at most one faulty molecule per row). (b) Impossible self-repair (more than one faulty molecule per row): `KILL=1` (self-repair at the organismic level).

The number of faulty molecules handled by the molecular self-repair mechanism is necessarily limited: in the example of Fig. 2.9a, we tolerate at most one faulty molecule per row. If more than one molecule is faulty in one or more rows (Fig. 2.9b), molecular self-repair is impossible, in which case a global signal `KILL=1` is generated to activate the organismic self-repair described above (Subsection C and Fig. 2.6).

F. The Embryonics Landscape

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. 2.1):

1) The basic primitive of our system is the *molecule*, the element of our new FPGA, consisting essentially of a multiplexer associated with a programmable connection network. The multiplexer is duplicated to allow the detection of faults. The logic function of each molecule is defined by its molecular code or `MOLCODE`.
2) 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 polymerase genome `PG` defines the topology of the cell, that is, its width, height, and the presence and positions of columns of spare molecules. In a second step, the ribosomic genome `RG` defines the logic function of each molecule by assigning its molecular code or `MOLCODE`.
3) A finite set of cells makes up an *organism*, an application-specific multiprocessor system. In a third and last programming step, the operative genome `OG` is copied into the memory of each cell to define the particular application executed by the organism (electronic watch, random number generator, and a Turing machine being examples shown by us to date)
4) The organism can itself self-replicate, giving rise to a *population* of identical organisms, the highest level of our hierarchy.

IV. Genotype-Phenotype Mapping

To the best of our knowledge, the Embryonics project is the only approach that implements quasi-biological growth processes in real hardware. Recently, the BioWatch, a large-scale implementation of a fault-tolerant and self-repairable watch has been presented [31], demonstrating that the systems described above are indeed feasible as a hardware implementation. A weak point of the Embryonics project, however, is that there exists no methodology and no tools allowing the automatic construction of artificial organisms. Thus, all organisms are coded by hand so far. Furthermore, the model is not flexible enough to allow for a direct implementation of learning and evolutionary mechanisms.

The main drawback of the Embryonics approach lies in the genotype-phenotype mapping, that is, in the way the application (the phenotype) is encoded by what is finally a sequence of electronic bits (the genotype). In Embryonics, the genotype is quite complex, consisting of three separate parts (the polymerase, ribosomic, and operative genomes) that need to be defined more or less separately. Even if we consider that evolutionary mechanisms in POEtic machines are to be applied mostly to the operative genome and that learning mechanisms will concern exclusively the ribosomic genome (very strong assumptions, not necessarily true), the model remains extremely cumbersome and the lack of a strict methodology will hinder further development.

The genotype-phenotype mapping is a thorny subject, and its determination will be one of the crucial aspects of the entire POEtic project. In reality, the ontogenetic axis is probably not the part of the project most concerned with this mapping, which becomes much more important when adaptation and (particularly) evolution are examined (without an efficient mapping, evolution becomes impossible). As a consequence, it is a bit early in the project to define a precise mapping, and the ontogenetic axis, through WP2 and WP4, should probably concentrate more on a general methodology for the design of cell-based machines, rather than on a specific mapping.

V. Conclusion: Towards POEtic Machines

As we have seen, the Embryonics project is without doubt the approach that most closely resembles the ontogenetic development of multi-cellular organisms in silicon. Its applicability to electronic circuits has been proven through the realization of many prototypes, and is therefore the ideal starting point for the conception of more advanced ontogenetic machines to be used within the POEtic project.

The Embryonics project is rather well advanced along the ontogenetic axis. However, its development never really took into consideration the *other* axes of the POE model (phylogenesis and epigenesis). Merging these three axes is the goal of the current project, and it will surely be necessary to alter or completely eliminate some of the features of the Embryonic machines. It is therefore vital to determine which of the features of the Embryonics project can be considered *requirements* in order to obtain machines that can be called “ontogenetic”.

Let us examine these features in the order in which they have been described above for the Embryonics project..

To begin with, for the reasons outlined in the introduction, the use of a programmable logic layer is a requirement to even loosely model processes such as growth and development. Moreover, if biological inspiration is respected, each *cell* in an ontogenetic system, whatever its size or shape, must contain the blueprint of the entire organism, the genome. Leaving aside such a feature would invalidate most of the more interesting features of ontogenesis (such as, for example, cicatrisation). A brief analysis of the resources required by such a system reveals that it would be unlikely to be able to obtain these features without a hierarchical approach similar to the one used in Embryonics (i.e., organisms divided into cells divided into molecules, the latter being the elements of a programmable logic circuit).

As far as the organismic features are concerned, the Embryonics project can provide some guidelines, but some important modifications would be desirable in view of the implementation of POEtic machines. For example, cellular division in Embryonics is a one-time process only, occurring once during configuration. It would be much more interesting, to more closely approach the phenomenon of growth, to be able to alter the size of the organism on-line, for example by adding new physical components to the system. Obviously, any ontogenetic system will have to realize, one way or another, something resembling cellular division. Embryonics does it through self-replication at the molecular level, where any number of cells can be “created” in parallel in the silicon space.

This parallel implementation, while not quite close to actual biological development, has the advantage of being efficient both from the standpoint of performance and from that of cost, and could be adapted to provide online growth. On the other hand, it does impose that all cells be identical, which could be a disadvantage for cellular differentiation, as we shall see. It will be interesting to see if the application of some of the morphogenetic approaches described above (L-systems, cellular encoding, P-systems, etc.) to an Embryonics-based system could improve the versatility of its cellular division (i.e., development) process.

As far as cellular differentiation is concerned, for the moment it has been studied relatively little within the Embryonics project. The cells are all identical in structure, and only their state can differentiate them, It would be much more interesting to be able to actually fit the *structure* of the cell to its function, rather than only its state. Or at least, if this approach should prove too expensive, increase the adaptation of the cells to the application (that is, design cells whose structure reflects more closely the function of the organism). In other words, a more flexible and more immediate mapping is required between the phenotype (the structure of the cells) and the genotype (their description at the genome level). As far as the coordinate system is concerned, it likely that some form of spatial information will always be required to obtain differentiation. The system used in Embryonics (propagation from the neighbors and increment) is one possibility among many, with the advantage of being simple to implement.

The self-replication of organisms could be an interesting feature, particularly for the Phylogenetic axis, which deals with populations of individuals. Embryonics implements it in a fairly “un-interesting” way, by allowing multiple exact copies of an organism to exist side by side. A much more interesting approach would be to allow the existence of multiple *different* organisms within the same silicon space. It is definitely a research direction, but does not concern directly the ontogenetic axis (which deals with the development of a *single* individual).

Finally, self-repair is one of the main *raison-d’être* of an ontogenetic system. If the system is set up in a structure fairly close to biology (i.e., with a complete genome in every cell), self-repair, or cicatrisation, becomes fairly simple, at least at the organismic level (subsection II.C). The main issue with self-repair is that the actual cicatrisation process is only part of the whole: a fault-detection mechanism is required, and is normally very complex to put into place. Embryonics provides a two-level fault-tolerance mechanism, a solution that provides good fault coverage with reasonable overhead. However, this technique cannot easily be generalized, depending both on the molecular and on the cellular layer. A less flexible, but perhaps simpler approach could be to limit self-repair to the cellular level, with a reasonable set of constraints. It should be noted, however, that even this approach would not eliminate the need to act on the molecular level, at least as far as self-test is concerned (it is not reasonable to assume that self-test could be limited to the cellular layer). Since the self-test logic depends essentially on the circuit itself, it is difficult to estimate at this stage the applicability of the Embryonics techniques to our final POEtic circuit. Nevertheless, the importance of fault tolerance in a complex system such as a POEtic machine is such that some mechanisms and solutions will probably have to be integrated in whatever substrate will be developed.

In conclusion, the Embryonics project can provide a few guidelines for the realization of ontogenetic systems, but is not likely to be applicable “as is” to the development of POEtic machines. One of the positive aspects of the project is that it is relatively independent of the technology used, that is, its basic concepts (e.g., self-replication, self-repair, multi-cellular and multi-molecular organization) do not make particular impositions on the architecture of the elements of the system, and cover most of the fundamental aspects of ontogenesis. In other words, these concepts can be applied to almost any architecture, both at the molecular and at the cellular level, a consideration that should prove extremely useful when merging the three axes of the POE model to develop our POEtic machines.

References

1. L. Wolpert. *The Triumph of the Embryo*. Oxford University Press, New York, 1991.
2. M. Nicolaidis. "Future Trends in Online Testing: a New VLSI Design Paradigm?". *IEEE Design and Test of Computers*, 15(4), 1998, p. 15.
3. D. Mange, M. Tomassini, eds. *Bio-inspired Computing Machines: Towards Novel Computational Architectures*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.
4. J. von Neumann. *The Theory of Self-Reproducing Automata*. A. W. Burks, ed. University of Illinois Press, Urbana, IL, 1966.
5. 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*, 43(10), 1997.
6. D. Mange, M. Sipper, A. Stauffer, G. Tempesti. "Towards Robust Integrated Circuits: The Embryonics Approach". *Proceedings of the IEEE*, vol. 88, no. 4, April 2000, pp. 516-541.
7. C. Ortega, A. Tyrrell. "Reliability Analysis in Self-Repairing Embryonic Systems". Proc. of the First NASA/DOD Workshop on Evolvable Hardware, Pasadena, CA, July 1999, pp. 120-128.
8. C. Ortega, A. Tyrrell. "Self-Repairing Multicellular Hardware: A Reliability Analysis". In D. Floreano, J.-D. Nicoud, F. Mondada, eds., Advances in Artificial

Life, Lecture Notes in Artificial Intelligence, Vol. 1674, Springer-Verlag, Berlin, 1999, pp. 442-446.

9. R. C. Merkle. "Making Smaller, Faster, Cheaper Computers". *Proceedings of the IEEE,* Vol. 86, No. 11, November 1998, pp. 2384-2386.
10. "A D&T Roundtable: Online Test". IEEE Design & Test of Computers, Vol. 16, No. 1, January-March 1999, pp. 80-86.
11. Y. Zorian. "Testing the Monster Chip". IEEE Spectrum, Vol. 36, No. 7, July 1999, pp. 54-60.
12. G. D. Watkins. "Novel Electronic Circuitry", Predictive Paper Reprint. Proceedings of the IEEE, Vol. 86, No. 11, November 1998, p. 2383.
13. P. Kuekes. "Molecular Manufacturing: Beyond Moore’s Law". Invited Talk. Proc. Field-Programmable Custom Computing Machines (FCCM’99), Napa, CA, April 1999.
14. 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.
15. R. F. Service. "Organic Molecule Rewires Chip Design". Science, Vol. 285, No. 5426, 16 July 1999, pp. 313-315.
16. 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.
17. H. Kitano. "Building complex systems using developmental process: An engineering approach". In M. Sipper, D. Mange, and A. Perez-Uribe, editors, Proceedings of the Second International Conference on Evolvable Systems (ICES'98), volume 1478 of Lecture Notes in Computer Science, pages 218-229, Berlin, Germany, September 1998. Springer-Verlag.
18. D. W. Thompson. *On Growth and Form.* Cambridge University Press, Cambridge, UK, 1961.
19. A. M. Turing. "The chemical basis of morphogenesis". Philosophical Transactions of the Royal Society of London, B 237:37-72, 1952.
20. P. T. Saunders, editor. *Collected Works of A. M. Turing: Morphogenesis*. North-Holland, Amsterdam, 1992.

21. A. Lindenmayer. Mathematical models for cellular interaction in development, parts I and II. Journal of Theoretical Biology, 18:280-315, 1968.

22. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

23. H. Kitano. "Designing neural networks using genetic algorithms with graph generation system". Complex Systems, 4:461-476, 1990.

24. F. Gruau. "Genetic systems of boolean neural networks with a cell rewriting developmental process". In D. Whitley and S. D. Schaffer, editors, *Combination of Genetic Algorithms and Neural Networks*. IEEE Computer Society Press, Los Alamitos, CA, 1992.

25. F. Gruau and D. Whitley. *The cellular developmental of neural networks: The interaction of learning and and evolution*. Technical Report 93-04, Ecole Normale Superieure de Lyon, Institut IMAG, January 1993.

26. F. Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, 1994.

27. G. M. Odell, G. Oster, P. Albrech, and B. Burnside. "The mechanical basis of morphogenesis I: Epithelial folding and invagination. Developmental Biology}, 85:446-462, 1981.

28. M. Weliky and G. Oster. "The mechanical basis of cell rearrangement. 1: Epithelial morphogenesis during fundulus epiboly". Development, 109:373-386, 1990.

29. K. W. Fleischer. *A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures*. PhD thesis, California Institute of Technology, Pasadena, CA, 1995.

30. G. Paun. "Computing with membranes". Journal of Computer and System Sciences, 61(1):108-143, 2000. First published in a TUCS Research Report, No 208, November 1998, http://www.tucs.fi.

31. A. Stauffer, D. Mange, G. Tempesti, and C. Teuscher. "BioWatch: A giant electronic bio-inspired watch". In D. Keymeulen, A. Stoica, J. Lohn, and R. Zebulum, editors, Proceedings of the Third NASA/DoD Workshop on Evolvable Hardware, EH-2001, pages 185-192. IEEE Computer Society, Los Alamitos, CA, 2001.