Published in *Proceedings of the NASA/DoD Conference on Evolvable Hardware 2003*, Chicago, IL, USA, 9-11 july 2003, which should be cited to refer to this work.
DOI: 10.1109/EH.2003.1217677

Developmental Processes in Silicon: An Engineering Perspective

Gianluca Tempesti, Daniel Mange, Enrico Petraglio, André Stauffer, Yann Thoma
Swiss Federal Institute of Technology in Lausanne (EPFL)
Logic Systems Laboratory
IN-N Ecublens, CH-1015 Lausanne, Switzerland
Gianluca.Tempesti@epfl.ch

Abstract

In this article, we try to analyze the requirements of developmental processes from the perspective of their implementation in digital hardware. After recalling the motivations for such an implementation, we concentrate separately on the two mechanisms (cellular division and cellular differentiation) that are exploited by biological systems to realize development. We then describe some of the current and projected solutions to implement such mechanisms in hardware, and conclude by analyzing the most interesting features of developmental approaches.

1 Introduction

The majority of living beings, with the exception of unicellular organisms like viruses and bacteria, share a common *multicellular organization*: the organism is divided into a finite number of cells, each realizing a single function (skin, neuron, muscle, etc.).

Outstanding examples of massive parallelism, these organisms are not designed in their final shape (as opposed to fully engineered electronic circuits), but rather go through a process, alternatively called development, ontogenesis, growth, or morphogenesis (terms often used interchangeably, even if differences do exist in biological terms) that continually alters their structure throughout their lifetime.

This process, which not only allows organisms to develop from a single initial cell (the zygote) to a fully-grown individual, but also provides mechanisms that let the organism survive considerable amounts of damage (illness, wounds, etc.), relies essentially on two mechanisms:

- *Cellular division* is the process through which each cell achieves its duplication. During this phase, a cell copies its genetic material (i.e. the genome) and splits into two identical daughter cells.
- *Cellular differentiation* defines which function a cell has to realize. This specialization, which essentially depends of the cell's position in the organism, is obtained through the expression of a part of the genome.

In this article, we shall try to justify the need for the implementation of development in electronic circuits, and to analyze, through some examples, the general requirements of developmental mechanisms in hardware, touching on their advantages/disadvantages. In particular, we shall consider development from the standpoint of its implementation in digital hardware, rather than biological plausibility. This approach has several consequences on the solutions proposed, as mechanisms that are biologically plausible are not necessarily well-adapted to electronic hardware: the developmental process in biology exploits heavily the chemistry and physics of carbon-based organisms, and a digital hardware implementation should exploit just as heavily the chemistry and physics of silicon-based systems.

2 Motivations

It is often difficult to justify the need for adding a developmental process to the design of digital circuits. Mainly, this difficulty is engendered by the important overhead imposed by the hardware that needs to be included to realize such a process. The overhead, in turn, is caused by the lack of versatility of silicon, which hampers all kinds of adaptive processes (in fact, this observation holds true for all kinds of bio-inspired processes, which rely heavily on adaptation). Also, the advantages of such a process for "conventional" applications are very slight, if they exist at all.

Nevertheless, research on development in silicon remains useful, as areas of application that can benefit from such mechanisms do exist. In this article, we shall analyze three areas in particular: adaptive systems, such as neural networks; molecular systems, i.e., systems that will exploit molecular-level technologies such as nanotechnologies; and reliable systems, i.e., systems that require fault tolerance.

2.1 Adaptive systems

Probably the most evident application of developmental approaches to machines lies within the domain of adaptive systems. This very general term is used to denominate all those systems, mechanical as well as electronic, that somehow alter their operation to fit their environment in way that cannot be predicted at design time.

Even if adaptation has been studied for *structural* modifications of a machine (e.g., for space exploration [13]), technological issues have steered interest towards the adaptation of the electronic components only. In particular, the most common field of application for adaptive approaches is in control systems, which have to react to an *a priori* unknown environments, normally embodied within a robot.

Typically implemented with neural networks (special cases of multi-cellular system where each cell is a neuron-like element), this kind of adaptive systems can exploit a developmental approach to address several issues:

- *Genotype-phenotype mapping*. It is common knowledge that the information stored in the genome is not sufficient to completely define the structure of the neural network of a biological organism. Current research suggests that the genotype codes *instructions* on how the neural network should grow, and that the environment intervenes to handle the details of its final structure. This concept has been applied with success to the development of artificial neural networks [24, 35, 41], particularly in view of applying evolutionary mechanisms to their design: the complex genotype-to-phenotype mappings allowed by these approaches imply a drastic reduction of the size of the genome, with a consequent increase in its evolvability.

- *Structural adaptation*. Networks that adapt or self-organize structurally to the environment (e.g., [16, 37, 39]) by adding and removing neurons and connections in the system exploit mechanisms that are similar to those used in the growth of an organism. A developmental mechanism implies the presence, for example, of a mechanism for cell creation, clearly a requirement for structural adaptation, and thus can greatly simplify the design of structure-adaptable systems.

- *Environmental adaptation*. An interesting, but little-exploited aspect of the developmental process of complex organisms is the impact that the environment has on the growth of an individual. Most adaptive systems, and indeed most developmental approaches, assume that the environment has an impact on the organism only in its adult form. Environment-directed development, almost ignored as a field of research [49, 50], represents one of our current research directions.

To summarize, developmental approaches have been applied to neural networks for a long time (see, for example, [1, 12]), and their efficiency has been proven, particularly whenever evolutionary approaches have been exploited [7, 10, 23, 34, 51]. Moreover, the same mechanisms that are implied in a developmental process can greatly improve the performance of a neural network by allowing it to alter its own structure in response to external stimuli.

2.2 Molecular systems

It is not yet clear which technology shall replace silicon. That one such technology is required in the search for ever-increasing performance is more or less universally accepted, and the most promising candidate today seems to be the field known as nanotechnology [8]. Even should a different technology take the lead, some consensus seems to have been reached on at least some of the features of future hardware. These features imply two consequences:

- *Conventional design methodologies will no longer be adequate*. Circuits will be built at the molecular scale, implying a density of computational elements so great that it will become impossible to fully design a system (incidentally, hardware overhead will disappear as an issue for nanocircuits). Self-organization techniques similar to the development of biological organisms are of great interest in this context, as they can potentially allow the specification of highly complex phenotypes though simple genotypes. Coded genotype-to-phenotype mapping is then a key issue for the implementation of development in computing systems.

- *It will be impossible to design defect-free circuits*. The development of defect-tolerant architectures is rapidly becoming a focus of interest for nanotechnologies [19, 21, 33], and is indeed one of the main thrusts of our own research [45]. Developmental processes, with their gradual occupation of a circuit through growth, allow the system to explore the hardware before it is used and to detect and avoid eventual faults. Gradual expansion through growth is then an important feature for systems implemented on defective hardware.

2.3 Reliable systems

The lack of reliability of conventional computing systems is a well-known issue in the world of computer design. Not extremely important for day-to-day applications, reliability, or fault tolerance to use another common term, is a key factor in what are usually called ”mission-critical” systems, i.e., systems that cannot crash without major consequences (some well-known examples are automotive, airplane, and space control systems).

The increase in complexity of the electronic circuits used in these applications increases the probability of faults, an issue that is bringing to the forefront the problem not only of testing the correctness of circuits, but also of designing circuits resistant to faulty components. Biological systems are astounding examples of complex fault-tolerant systems, and can be of great interest in the development of novel self-test and self-repair approaches.

Self-test in biological organisms is a highly complex hierarchical system, which is inspiring research on fault detection at several levels (see, for example, [3, 4]). More importantly for the topic of this paper, self-repair in biological organisms is also a highly complex hierarchical system, based, to a large extent, on the replacement of dead cells with new, functionally identical ones: in a multicellular organism, each cell contains the whole of the organism's genetic material (i.e. the genome) and is therefore "universal", i.e. potentially capable of replacing any other cell (in reality, only a minority of cells, the so-called *stem cells* [36] retain this capability in full, but every organism retains a number of these cells throughout its lifetime). In presence of a physical degradation, each living organism is then potentially capable of self-repair (i.e., through cicatrization).

The replacement of non-functional units by identical spare units is far from novel in the world of fault-tolerant systems [32]. However, the vast majority of such systems, based on a centralized controller to oversee the reconfiguration process, are not capable of *preserving correctness through reconfiguration*, i.e., the process implies a re-start of the system, which might well not be acceptable in a mission-critical application, and definitely is not in an adaptive system, where the learned information would be lost.

We have shown in the past [28, 29] that the bio-inspired concept of storing a complete copy of the genome in each cell proves to be very useful when a reconfiguration mechanism has to be implemented (see figure 1). But another key observation to arise from our research is that, in artificial as in biological organisms, *fault tolerance exploits many of the same mechanisms as development*. For example, new cells required to replace dead ones are created in almost exactly the same way as new cells required for the growth of the organism. Moreover, a very pragmatic analogy can also be made between the biological mechanism of cell death (*apoptosis*) and the cell deactivation mechanism required for reconfiguration: even if not very biologically plausible (apoptosis is not related to cicatrization in any meaningful way in biology), these mechanisms share similar effects, as in both cases cells have to be removed from the organism.

This observation adds value to the conception of developmental approaches, which will then not only allow adaptability via structural dynamics and self-organization in defective milieus, but also provide mechanisms to automatically increase the reliability of computing systems.

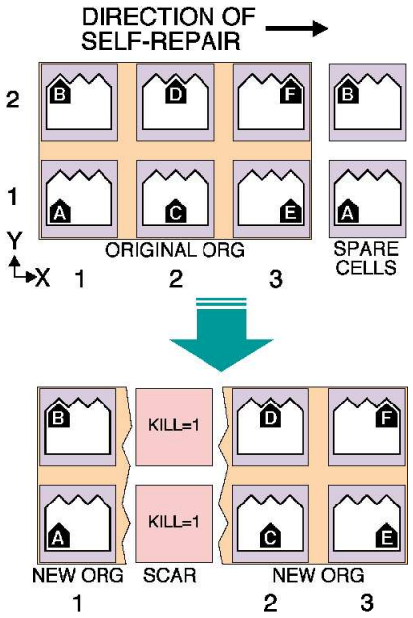

Figure 1. Self-repair through reconfiguration. Each cell contains the entire genome, but expresses only one gene (A to F) depending on its position (coordinates).

3 Cellular division

As mentioned in the introduction, the development process of biological organisms exploits essentially two mechanisms: cellular differentiation and cellular division.

Of these two mechanisms, the most difficult to implement in silicon-based systems is undoubtedly cellular division. In biological organisms, in fact, this feature implies the formation of new physical entities (cells). Such manipulation of the material substrate is not possible in today's electronic circuits (though it might be in the future!), which cannot be physically altered after fabrication. Luckily for bio-inspired research, programmable logic devices (FPGAs) [42, 47] can be used to approximate this process by allowing the manipulation not of the physical substrate, but of the *logical structure* of a circuit.

Even within this limitation, cellular division is far from simple, and very few approaches have actually been implemented in hardware. Several cellular division mechanisms have been proposed throughout the history of computing, but almost without exception they are *implicit*: the mechanism is simply seen as the support that allows development, and it is assumed possible to create new cells in the organism without addressing the actual implementation of such a process, focusing rather on realizing cellular differentiation mechanisms.

Nevertheless, several approaches do address the issue of cellular division as separate from differentiation. Usually (but not always), these approaches deal directly with the phenomenon of *self-replication*, that is, the capability of a machine to produce a copy of itself, with development of neural networks, or else with the *morphogenesis* of an artificial organism, that is, with the development of its *form*, rather than that of its *function*. In the next subsections, we will present these approaches under three categories: *self-replicating*, *neural*, and *morphological* approaches, keeping in mind that only in very rare cases have these approaches been implemented in hardware.

We shall then present in a separate section one of our main research projects, *Embryonics*. This approach will be treated separately not only because it does not neatly fit into one of the two categories, but also because it is, to the best of our knowledge, the only project that has been trying to explicitly address the problem of implementing a cellular division mechanism in digital circuits using a bottom-up approach, that is, by trying to apply developmental principles to more or less conventional computing architectures.

3.1 Self-replicating approaches

Historically, the first research on self-replication of computing machines can be dated to von Neumann's design of his *Universal Constructor* [52]. This machine (figure 2), implemented as a cellular automaton [5], has many of the features now considered essential in the replication of cells (remember that von Neumann had no knowledge of the structure of DNA, discovered only years later).

Viewed as a uni-cellular organism, in fact, the constructor relies on the presence of a linear description of itself (the genome) which is first *interpreted* to build a copy of the machine, and then *read* to duplicate the genetic information.

Von Neumann's Universal constructor, however, is an extremely complex machine, too large for a hardware realization to be effective. This complexity is essentially due to the machine's ability to construct *any* other machine, if supplied with its description. It was in order to study self-replication as a separate, simpler process that Langton [26] designed his *self-replicating loop*, breathing new air into research on the self-replication of machines. In the years following Langton's seminal work, other researchers gradually modified the initial loop to compensate for its weaknesses (its size [40], its lack of functionality [44], its fragility, etc.) so that today we can safely say that the self-replication of computing machines in cellular automata is indeed possible.

While not strictly limited to cellular automata implementations, self-replicating approaches have for the most part exploited this environment. CAs, unfortunately, are extremely ill-adapted for a hardware implementation, unless they are conceived from the start with this goal in mind, as we are currently doing in our Embryonics project (see below). They are worth mentioning in the context of cellular division because the approaches we described set, within their limitations, cellular division as their main thrust (which is often not the case, for example, for several of the approaches described below). However, their translation into hardware is far from simple, and alternative approaches have proven more efficient from this point of view.

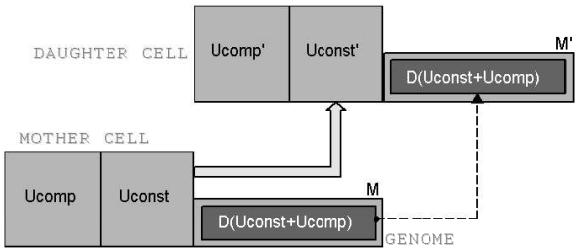

Figure 2. Self-replication of a universal computer in Von Neumann's approach.

3.2 Neural approaches

Among the most common approaches to the implementation of developmental mechanisms in neural networks are grammar-based systems. Often based on L-systems [27], initially proposed as a model for plant development, these approaches use sets of *production rules* iteratively applied to develop complex structures starting from an initial "seed". The main advantage of this kind of coding is, of course, the dramatic reduction of the size of the genotype, which has to encode the construction rules rather the final structure of the organism [18].

The applications of developmental systems to neural networks are too numerous to be comprehensively listed here. They are often applied to the development of the initial structure of neural networks [1, 7, 10, 34], and do not, strictly speaking, exploit differentiation as all the neural elements of the system are generally structurally identical.

Of particular interest [49, 50], in our opinion, are systems that couple a grammar-based developmental mechanism with an environmental influence. That is, the final structure of the organism results from a combination of genetic information and of environmental factors.

From our viewpoint, the main weakness of these approaches is the fact that the substrate in which they operate is often unspecified: the algorithms for development call for the creation and destruction of neural cells and of connections, but they rarely go beyond simulations into the details of the implementation of such systems in silicon.

3.3 Morphological approaches

Another interesting field of application for developmental approaches has traditionally been the development of what are generally called *animats*, that is, robots whose structure and behavior are inspired by those of biological organisms. In these systems, the morphology of the machine can have an important impact on its performance and some very interesting work has been done on machines that develop in *shape* as well as in function [2, 7, 11, 20, 22, 43].

As for neural approaches, morphological approaches tend to ignore the implementation issues, although they are perhaps more justified in doing so, considering the milieu in which morphological development occurs: whereas programmable logic has made possible the implementation of electronic circuits that are capable of adaptation to a certain degree, the transposition of such dynamics to building materials is not yet feasible.

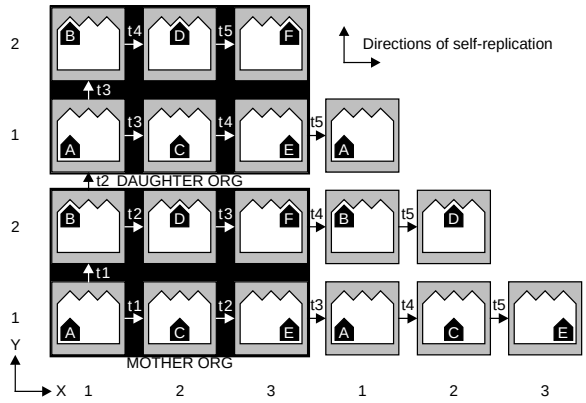

Figure 3. Self-replication of a 6-cell organism in a limited homogeneous array. Only the expressed gene is shown in each cell.

3.4 The Embryonics approach

The main goal of the Embryonics (embryonic electronics) project [29, 31, 45] is to implement, in an integrated circuit, a fault tolerant system inspired by the development of multi-cellular organisms. In a sense, it is the only effort (to our knowledge) to implement in hardware a cellular division mechanism with an emphasis not so much on biological plausibility, but rather on electronic efficiency. Some recently published work does introduce developmental hardware [9, 17], but without addressing cellular division, supposing that cells are available when needed.

In Embryonics systems (figure 3), an artificial organism (ORG) is realized a set of *cells*, distributed at the nodes of a regular two-dimensional array. Each cell contains a small processor coupled with a memory used to store the program (identical for all the cells) that represents the organism's genome. In the organism, each cell realizes a unique function, defined by a sub-program called the *gene*, which is a part of the genome. The gene to be executed in a cell is selected depending on the cell's position, defined by a set of X and Y coordinates. In figure 3, the genes are labelled A to F for coordinates $(X, Y) = (1, 1)$ to $(X, Y) = (3, 2)$.

Development in Embryonics occurs through a process of self-replication. In fact, there are two self-replication mechanisms in our systems. The first kind is that of the organism: an artificial organism can replicate itself if there is enough free space in the array (at least six cells in figure 3) to contain the new daughter organism: as each cell is configured with the same information (the genome), a cycle in the coordinate pattern (e.g., $Y = 1 \rightarrow 2 \rightarrow 1 \rightarrow 2$) causes the repetition of the same pattern of genes and thus, in a sufficiently large array, the self-replication of the organism for any number of specimens in the X and/or the Y axes.

In Embryonics, however, there is a second replication process, which corresponds to the cellular division process in biological entities, used to put in place the initial array of cells that will then be differentiated to obtain the organism.

The need to build cells of different size and structure depending on the application naturally led to the use of programmable logic. Practical considerations then led us to develop our own FPGA, incorporating features dedicated to the realization of cellular development. Each of the computational elements of our FPGA can then be seen as *molecules*, assembled in a precise configuration to form a cell. As all cells are identical, the development process is analogous to the replication of this configuration for as many times as there are cells in the organism.

Embryonics implements this process in two phases: a *structural* phase, where a "skeleton" is created in order to divide the physical space into a groups of molecules (empty cells), and a *configuration* phase, where the configuration is sent in parallel into all the empty cells.

The structural phase is implemented by a small cellular automaton (CA) [6], placed in the spaces between the molecules (figure 4), capable of transforming a one-dimensional string of states (a configuration bitstream) into a two-dimensional structure (the blocks that will host the cells). Given an appropriate sequence of states, the CA will partition the array into identical blocks of variable size, defining a "skeleton" that can be seen as the *membranes* of the artificial cells. Once the membrane is in place, the second part of the cellular self-replication begins: a bitstream containing the genome is sent to all the blocks in parallel (figure 4), automatically creating multiple copies of the same artificial cell. At the end of this phase, the coordinate-based differentiation mechanism (see next section) defines the structure of the organism.

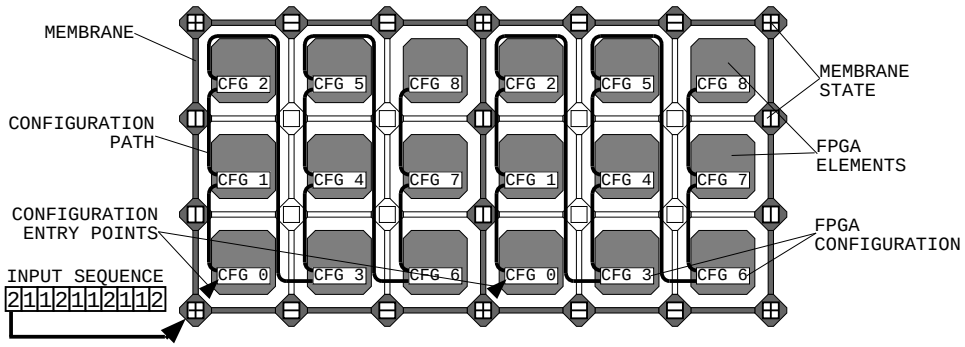

Figure 4. First, an input sequence is sent to the CA (represented by the diamond-shaped elements) to set up the cells' membrane. Then, the genome is sent in parallel to each cell in the organism.

Of course, this replication process is quite different from real cellular division, even within the limitations imposed by silicon, as all cells are created in parallel. To ameliorate our approach, we are currently developing a set of mechanisms [30, 38], designed to be integrated with Embryonics but much closer to biological cellular division in the sense that the system will begin operating from a single cell, which will autonomously launch the self-replication process and create a copy of itself, which will then itself replicate, and so on until the complete organism is in place.

4 Cellular differentiation

Cellular differentiation is a mechanism very closely tied in, of course, with cellular division: in a biological organism, once the cells have divided, they specialize to realize a specific task (skin cell, liver cell, neurons, etc.). The differentiation mechanism in nature is then based on the creation of new, specialized cells, whose structure is adapted to their task, and thus implies cellular division. As we have seen, however, in electronics the creation of cells is not yet feasible, and most approaches to cellular division suppose that the cells that will compose the final organism are already physically present, but are inactive until required: if a surface of totipotent cells is provided, cellular differentiation can be implemented by selecting which part of the genetic information will be activated in each cell.

With this assumption, the developmental process can actually be seen as based essentially on differentiation, more or less independently of a "true" cellular division: the inactive stem cells are activated, and a differentiation mechanism selects their function, usually depending on their surroundings (their physical position among the other cells).

In general, differentiation can be implemented according to two kinds of approaches: *genetic approaches*, where the information required for the construction of the organism is fully contained in the genome, and *environmental approaches*, where development is at least partially influenced by the environment. Each of these approaches has its own strengths and weaknesses, as detailed in the next subsections. In general, however, differentiation requires some sort of positional information for the cell (even if the cell's identification need not be unique), so that it can identify its role within the organism and specialize accordingly.

Even within systems based on the activation/deactivation of cells (rather than on their creation and destruction), however, the dynamics of the system introduce considerable problems with respect to a hardware implementation, problems that do not appear in simulation. To cite only the most flagrant, whenever a new cell appears (and unless all interactions are local, in which case other serious problems are introduced) it has to be somehow connected to the rest of the organism. In programmable logic, connections are traditionally the most complex and fragile part of a system, and in fact the development of a truly dynamic connection network is extremely costly in conventional FPGAs.

Addressing this issue is one of the main goals of the circuit we are developing within the POEtic project (http://www.poetictissue.org), funded by the EC in cooperation with several research groups throughout Europe [46, 48]. The goal of the project is the fabrication of a programmable logic device dedicated to the implementation of digital bio-inspired systems. Together with features designed specifically for evolution and/or learning, it contains some circuitry dedicated to simplify the task of designing developmental systems. In particular, to address the connection issue, it contains a fully dynamic autonomous routing network [references will be available for the final version] that can automatically handle intercellular communication, without need to specify at design time the connection paths. Allowing newly created cells to be seamlessly integrated within an existing network, it will be invaluable for the implementation of developmental mechanisms.

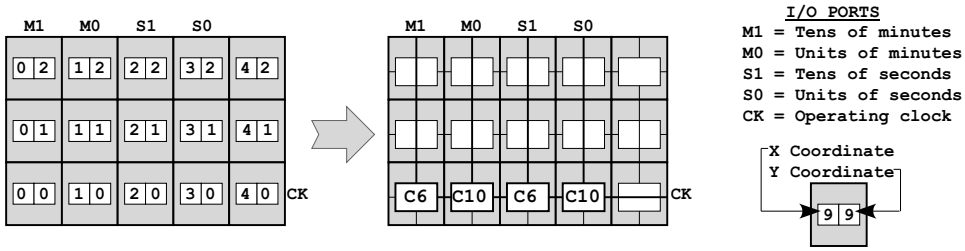

Figure 5. Example of a coordinate-based system that places four cells of two different types (C6 and C10) in a 5x3 array according to a predetermined differentiation pattern to implement a timer.

The self-routing mechanism, moreover, has the additional benefit of simplifying *other* processes, such as self-repair through reconfiguration (where a defective cell is killed and replaced by a spare stem cell) and structural adaptation (where connections between cells are dynamically altered during the lifetime of the organism). Considering how these mechanisms, even in biology, are similar to those used in development, this result is not surprising, but is nevertheless of great interest for the design of bio-inspired hardware.

4.1 Genome-based systems

Probably the simplest approach to the study of differentiation in artificial organisms is to code within the genome the information required for the construction of the organism, defining not only its general structure, but its internal composition as well. The basis of this kind of approach is the presence of different cell types and of an algorithm that selects the type of each newly created cell in the system.

Systems that rely on this approach [11, 10, 41] have been used to develop very interesting structures using some reasonably bio-plausible approaches. It should however be noted that this research has been carried out specifically with the goal of defining (and eventually evolving) complex structures in *simulation*. This observation is fundamental, because it could very well be argued that *a genome-based differentiation mechanism in hardware would be extremely inefficient*. The hardware overhead implicit in these approaches would in fact be considerable, and cannot really be justified: the algorithm could just as easily be used at design time, the hardware implementation being reserved for the adult organism only.

In a sense, our Embryonics systems are genome-based: once an algorithm has defined the differentiation pattern to be realized within the organism, the *most efficient* hardware realization of differentiation is a simple coordinate system (figure 5), assigning a function to each cell depending on its spatial position (see subsection 3.4). We argue that such a system suffices to realize in hardware any genome-based differentiation algorithm with a minimal overhead.

4.2 Environment-aware systems

The development of biological organisms, with few exceptions (such as the nematode worm *C.Elegans*), is influenced by the environment from its earliest stages. This impact has been recognized in developmental approaches applied to neural networks [49, 50, 51], but never, to our knowledge, to systems that implement differentiation mechanisms (the few possible exceptions, e.g. [25], do not deal with hardware implementations).

Exploiting the features of the POEtic tissue, we are attempting to exploit the interaction between the environment and the developmental process to design systems that are capable of adapting not only their physical structure, but also their *functional structure* to their environment. For example, a "conventional" protein gradient approach could be made aware of the environment simply by placing the diffusers at the interface with the outside world, that is, on the I/O ports of the circuit (figure 6). In this example, the information provided by this mechanism allows the creation of paths leading from the functional cells (the four digits of the timer) to the I/O pins of the circuit, and from there to the external units of the system (clock generator, displays).

This very basic mechanism can be used to introduce non-determinism in the development of an artificial organism, a fundamental feature for biological plausibility (two individuals with the same genetic material will not be identical in their adult phase). This non-determinism implies that the circuit can adapt itself to the environment: for example, in figure 6 the same organism can structure itself differently depending on the placement of the I/O ports.

Environment-aware systems are then much more versatile (and interesting!) than genome-based approaches. This versatility, however, comes at a price, as the mechanisms involved are hardware-intensive and, in a way, difficult to exploit. In general, conventional applications do not justify such a price. This feature, however, can have a vital role in highly adaptive systems, such as, for example, *modular robotics*, where identical entities can realize different functions depending on their placement within the organism.

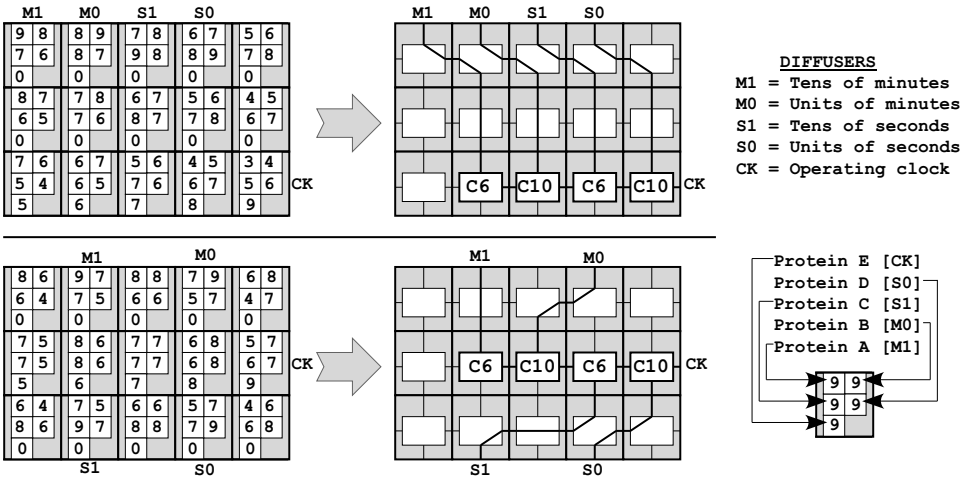

Figure 6. Example of a gradient-based system that places four cells of two different types (C6 and C10) in a 5x3 array according to an environment-directed differentiation pattern to implement a timer. Depending on the position of the diffusers (I/O ports), the position of the cells changes.

5 Analysis

The far from exhaustive analysis of developmental approaches in electronics contained in this articleshould be sufficient to outline the most important problem that this kind of research has to face: development is based on a on a process of physical growth, in which new cells are created and destroyed according to algorithm which is partly genetic and partly influenced by the environment. Moreover, the cells that are created or destroyed can have varying sizes and functions, depending on the organism's task.

Unfortunately, this kind of physical dynamics cannot be directly implemented in silicon, which is in fact a material very ill-suited to adaptive systems. Programmable logic can in part solve the problem, but the implementation of cell division and differentiation on this kind of substrate introduces several practical issues. Developmental approaches have thus been realized almost exclusively in simulation.

It is obvious that simulation allows a versatility not possible in hardware, and this versatility has been exploited to implement several *biologically-plausible* approaches. It remains in our opinion not proven that such mechanisms can indeed be useful for the design of electronic circuits: biological development and the mechanisms it exploits are deeply entwined with the physics and the chemistry of carbon-based entities, while electronics rely on very different rules. We believe that this crucial difference justifies a *bottom-up* approach that applies developmental mechanisms to "conventional" architectures, shifting the focus from biological plausibility (without, of course, ignoring it) to electronic efficiency. Once an efficient basis has been established, it can then be altered to integrate more complex models: this is the core of the Embryonics approach.

In searching for solutions to the problems of implementing development in a digital circuit, we came to the conclusion that conventional programmable logic cannot realize this kind of systems efficiently. We then resorted to designing our own programmable logic device, which integrates features dedicated exclusively to development. Even with this support, we could only realize systems that hardly resemble "true" cellular development.

To advance toward more powerful and versatile systems, we are currently integrating the Embryonics approach with more complex mechanisms. On one hand, we are working on *autonomous cellular replication*, opening the way to systems that consist not of identical cells, but of cells whose structure is determined during the growth of the organism. On the other hand, we are developing a *novel programmable logic substrate* (the POEtic tissue) with features that will allow us to create dynamic systems that can adapt much more efficiently than conventional circuits.

The goal of our research is to realize *efficient* electronic devices that exploit development to implement some of the most interesting features of multicellular organisms:

- *devices that can self-organize*, to exploit the possibilities offered by molecular-scale electronics;
- *devices that can operate on defective substrates* by exploring the space where development will occur and by avoiding faulty areas of the circuit;
- *devices that are fault tolerant*, using spare stem cells to preserve functionality on faulty hardware;

- *devices that can adapt to the environment* to a much greater degree than currently possible, by using environmental information from the very start of their development, allowing circuits to structure themselves to best perform their function;
- *devices that can evolve* more easily than today's systems, by providing complex genotype to phenotype mappings that encode structural information as constructive algorithms, rather than as descriptions of the fully grown adult.

In our opinion, significant advances in all of these areas are required before many of the promises of bio-inspired systems (to mention but one, their use in space exploration [13, 15, 14]) can become realities. Development is a key factor for achieving these results, and should rightly be placed alongside evolution and adaptation as one of the main axes of bio-inspiration in the design of computing machines. One possible approach to the design of such systems is to investigate in depth the biological mechanisms involved, trying then to adapt them to silicon. In this article, we tried to show that a *second* approach exists: applying developmental mechanisms to existing, and proven, hardware architectures, so as to more directly exploit the properties of silicon-based technology.

Acknowledgements

This work was supported in part by the Swiss National Science Foundation under grant 20-63711.00, by the Leenaards Foundation, Lausanne, Switzerland, and by the Villa Reuge, Ste-Croix, Switzerland.

Some of the material presented was the outcome of research funded by the Future and Emerging Technologies programme(IST-FET) for the European Community, under grant IST-2000-28027 (POETIC).The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to this project are supported under grant 00.0529-1 by the Swiss government.

References

[1] R. K. Belew. Interposing an ontogenic model between genetic algorithms and neural networks. In *Advances in Neural Information Processing Systems (NIPS5)*, pages 99–106. Morgan Kaufmann, 1993.

[2] J. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proc, of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 829–836. Morgan Kaufmann, 2001.

[3] D. Bradley, C. Ortega, and A. Tyrrell. Embryonics + immunotronics: A bio-inspired approach to fault tolerance. In *Proc. 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 215–223. IEEE Computer Society Press, 2000.

[4] D. Bradley and A. Tyrrell. Multi-layered defence mechanisms: Architecture, implementation and demonstration of a hardware immune system. In *Proc. 4th Intl. Conf. on Evolvable Systems: From Biology to Hardware (ICES01)*, volume 2210 of *Lecture Notes in Computer Science*, pages 140–150. Springer Verlag, 2001.

[5] A. Burks, editor. *Essays on Cellular Automata*. University of Illinois Press, Urbana, IL, 1970.

[6] E. F. Codd. *Cellular Automata*. Academic Press, New York, NY, 1968.

[7] F. Dellaert and R. Beer. A developmental model for the evolution of complete autonomous agents. In *From Animals to Animats 4: Proc. 4th Intl. Conf. on Simulation of Adaptive Behavior (SAB96)*. MIT Press, 1996.

[8] K. E. Drexler. *Nanosystems: Molecular Machinery, Manufacturing and Computation*. John Wiley, New York, NY, 1992.

[9] R. Edwards. Circuit morphologies and ontogenies. In *Proc. 2002 NASA/DoD Conference on Evolvable Hardware (EH-2002)*, pages 251–260. IEEE Computer Society Press, 2002.

[10] P. Eggenberger. Creation of neural networks based on developmental and evolutionary principles. In *Proc. 7th Int. Conf. on Artificial Neural Networks (ICANN97)*, volume 1327 of *Lecture Notes in Computer Science*, pages 337–342. Springer Verlag, 1997.

[11] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proc. 4th European Conf. on Artificial Life (ECAL97)*, Complex Adaptive Systems Series. MIT Press, 1997.

[12] E. Fiesler. Comparative bibliography of ontogenic neural networks. In *Proc. 1994 Intl. Conf. on Artificial Neural Networks (ICANN94)*, volume 1, pages 793–796. Springer Verlag, 1994.

[13] R. A. Freitas Jr. A self-reproducing interstellar probe. *Journal of the British Interplanetary Society*, 33:251–264, 1980.

[14] R. A. Freitas Jr. Terraforming mars and venus using machine self-replicating systems. *Journal of the British Interplanetary Society*, 36:139–142, 1983.

[15] R. A. Freitas Jr., T. J. Healy, and J. E. Long. Advanced automation for space missions. In *Proc. 7th Intl. Joint Conf. on Artificial Intelligence (IJCAI81)*, pages 803–808. Morgan Kaufmann, 1981.

[16] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[17] T. G. W. Gordon and P. J. Bentley. Towards development in evolvable hardware. In *Proc. 2002 NASA/DoD Conference on Evolvable Hardware (EH-2002)*, pages 241–250. IEEE Computer Society Press, 2002.

[18] P. Haddow, G. Tufte, and P. Van Remortel. Shrinking the genotype: L-systems for ehw? In *Proc. 4th Intl. Conf. on Evolvable Systems: From Biology to Hardware (ICES01)*, volume 2210 of *Lecture Notes in Computer Science*, pages 128–139. Springer Verlag, 2001.

[19] J. Han and P. Jonker. A system architecture solution for unreliable nanoelectronic devices. *IEEE Transactions on Nanotechnology*, 1(4):201–208, 2002.

[20] F. Hara and R. Pfeifer. On the relation among morphology, material, and control in morpho-functonal machines. In *From animals to animats 6. Proc. 6th Int. Conf. on Simulation of Adaptive Behavior (SAB'2000)*, pages 33–42. MIT Press, 2000.

[21] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280(5370):1716–1721, 1998.

[22] H. Kitano. Designing neural networks using genetic algorithms with graph generation system". *Complex Systems*, 4:461–476, 1990.

[23] H. Kitano. Morphogenesis for evolvable systems. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 99–117. Springer Verlag, Heidelberg, DE, 1996.

[24] H. Kitano. Building complex systems using developmental process: An engineering approach. In *Proc. 2nd Intl. Conf. on Evolvable Systems: From Biology to Hardware (ICES98)*, volume 1478 of *Lecture Notes in Computer Science*, pages 218–229. Springer Verlag, 1998.

[25] S. Kumar and B. P. J. Biologically inspired evolutionary development. In *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES2003)*, volume 2606 of *Lecture Notes in Computer Science*, pages 57–68. Springer Verlag, 2003.

[26] C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.

[27] A. Lindenmayer. Mathemetical models for cellular interaction in development, parts i and ii. *J. Theor. Biol.*, 18:280–315, 1968.

[28] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward self-repairing and self-replicating hardware: The embryonics approach. In *Proc. 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 205–214. IEEE Computer Society Press, 2000.

[29] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Towards robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–541, 2000.

[30] D. Mange, A. Stauffer, E. Petraglio, and G. Tempesti. Artificial cell division. In *Proc. Int. Conf. on Information Processing in Cells and Tissues (IPCAT03) - Submitted*, 2003.

[31] P. Marchal, C. Piguet, D. Mange, A. Stauffer, and S. Durand. Embryological development on silicon. In *Artificial Life IV*, pages 365–370, 1994.

[32] R. Negrini, M. Sami, and R. Stefanelli. *Fault Tolerance through Reconfiguration in VLSI and WSI Arrays*. MIT Press, Cambridge, MA, 1989.

[33] K. Nikolic, A. Sadek, and M. Forshaw. Architectures for reliable computing with unreliable nanodevices. In *Proc. 1st IEEE Conf. on Nanotechnology (IEEE-NANO2001)*, pages 254–259. IEEE Press, 2001.

[34] S. Nolfi and D. Parisi. Evolving artificial neural networks that develop in time. In *Advances in Artificial Life: Proc. 3rd European Conf. on Artificial Life (ECAL95)*, volume 929 of *Lecture Notes in Computer Science*, pages 353–367. Springer Verlag, 1995.

[35] S. Nolfi and D. Parisi. Genotypes for neural networks. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA, 1995.

[36] H. Pearson. The regeneration gap. *Nature*, (414):388, 2001.

[37] A. Perez-Uribe. *Structure-Adaptable Digital Neural Networks*. Ph.d., Swiss Federal Institute of Technology (EPFL), 1999.

[38] E. Petraglio, D. Mange, A. Stauffer, and G. Tempesti. Autonomous cell division by self-inspection. In *Proc. European Conf. on Artificial Life (ECAL03) - Submitted.*, 2003.

[39] P. Quinlan. Structural change and development in real and artificial neural networks. *Neural Networks*, 11:577–599, 1998.

[40] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng. Simple systems that exhibit self-directed replication. *Science*, 259:1282–1287, 1993.

[41] T. Reil. Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In *Advances in Artificial Life: Proc. 5th European Conf. on Artificial Life (ECAL99)*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pages 457–466. Springer Verlag, 1999.

[42] E. Sanchez. Field-programmable gate array (fpga) circuits. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 1–18. Springer Verlag, Berlin, DE, 1996.

[43] K. Sims. Evolving virtual creatures. In *Computer Graphics (Proc. SIGGRAPH'94)*, pages 15–22, 1994.

[44] G. Tempesti. A new self-reproducing cellular automaton capable of construction and computation. In *Advances in Artificial Life: Proc. 3rd European Conf. on Artificial Life (ECAL95)*, volume 929 of *Lecture Notes in Computer Science*, pages 555–563. Srpinger Verlag, 1995.

[45] G. Tempesti, D. Mange, and A. Stauffer. A robust multiplexer-based fpga inspired by biological systems. *Journal of Systems Architecture*, 43(10):719–733, 1997.

[46] G. Tempesti, D. Roggen, E. Sanchez, Y. Thoma, R. Canham, and A. Tyrrell. Ontogenetic development and fault tolerance in the poetic tissue. In *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES2003)*, volume 2606 of *Lecture Notes in Computer Science*, pages 141–152. Springer Verlag, 2003.

[47] S. M. Trimberger, editor. *Field-Programmable Gate Array Technology*. Kluwer Academic, Boston, MA, 1994.

[48] A. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.-M. Moreno, J. Rosenberg, and A. Villa. Poetic tissue: An integrated architecture for bio-inspired hardware. In *Proc. 5th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES2003)*, volume 2606 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 2003.

[49] J. Vaario. From evolutionary computation to computational evolution. *Informatica*, 1994.

[50] J. Vaario, N. Ogata, and K. Shimohara. Synthesis of environment directed and genetic growth. In *Artificial Life V*. MIT Press, 1996.

[51] J. Vaario, A. Onitsuka, and K. Shimohara. Formation of neural structures. In *Proc, 4th European Conference on Articial Life (ECAL97)*, pages 214–223. MIT Press, 1997.

[52] J. Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, 1966. Edited and completed by A. W. Burks.