*SNF Professorship Application – Research Plan*

# Architectures and design methodologies for bio-inspired computing machines

***Gianluca TEMPESTI***

## Table of contents

# 1. Summary of Research Plan

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).

Astounding examples of massively parallel processing, multi-cellular organisms are an extremely interesting source of inspiration for the design of electronic computing machines, particularly when considering how extremely complex *global behaviors* (the most remarkable example being, of course, intelligence) are obtained through the cooperation of numerous very simple elements (the cells). The implementation of bio-inspired systems in silicon, however, is quite difficult, due to the sheer number and complexity of the biological mechanisms involved. Conventional approaches exploit a very limited set of biologically-plausible mechanisms to solve a given problem, but often cannot be generalized because of the lack of a *methodology* in the design of bio-inspired computing machines. This lack is a consequence of the heterogeneity of the hardware solutions adopted, which is itself due to the lack of *architectures* capable of implementing a wide range of bio-inspired mechanisms.

Biological inspiration in the design of computing machines finds its source in essentially three biological models: *phylogenesis* (P), the history of the evolution of the species, *ontogenesis* (O), the development of an individual as directed by his genetic code, and *epigenesis* (E), the development of an individual through learning processes (nervous system, immune system) influenced both by their genetic code (the innate) and by the environment (the acquired). The architecture to be developed in the proposed project will be capable to support specialization and adaptation through the modification of the resources depending on the application and on the model of bio-inspiration to be implemented.

In fact, Nature is *change*: adaptation and specialization are key features of all biological organisms. As a consequence, no bio-inspired architecture can be rigid, but must rather be able to adapt and specialize depending on the desired application. This requirement, a major obstacle until recently, can today be satisfied by exploiting a technology known as *reconfigurable logic*. This technology is based on VLSI circuits that can be *programmed* to implement different hardware architectures. The circuits can then be reset and re-programmed at will, a feature that makes them a vital resource in the implementation of bio-inspired systems.

Integrating reprogrammability into an architecture, however, is far from trivial. The heterogeneity of the applications that can potentially be implemented poses considerable design problems: a novel kind of architecture needs to be developed to fully exploit the potential of reconfigurable logic for bio-inspired systems. The conception of such an architecture is the first step of the proposed project.

Defining an architecture for bio-inspired machines is not sufficient to achieve the goals of the project, which aims at providing an integrated environment for the design of such machines. Such an environment should include a *customizable design flow* allowing the user to concentrate on the desired approaches. The three models of bio-inspiration are well-suited to different aspects of an architecture, and the user should be able to exploit the models separately, as well as together.

The key feature of the proposed environment is then the *automatization* of a large part of the design flow. The intervention of the user will be required only to decide which bio-inspired models are required by the application and the environment will then automatically handle the implementation of the models into a reprogrammable platform.

The development of a proper design methodology for bio-inspired systems is the logical next phase for a domain that is finally reaching maturity. Systems inspired by biological mechanisms are carving themselves a niche in several areas of computation, but the solutions adopted often cannot be generalized for the lack of a universal architecture capable of integrating the different approaches. The goal of this project is then precisely *the definition of a universal architecture and of a design methodology for the conception of bio-inspired digital hardware*, in order to simplify the implementation of a *wide range of quasi-biological mechanisms*.

# 2. Research Plan

## 2.1. State of the art in the domain

Biological organisms grow, adapt, heal, and reproduce, all features that are not truly present in any existing computing system. The introduction of such features in artificial machines would allow a quantum leap in the performance of digital hardware, particularly for applications where mere speed of computation is not the only concern. It is therefore not surprising that, since the very dawn of computer science [50][54][57], researchers have been seeking inspiration in nature for the design of computing machines.

An analysis of the mechanisms that regulate the development of living systems in nature reveals the presence of three levels of organization (Figure 1) [43]: *phylogenesis* (P), the history of the evolution of the species, *ontogenesis* (O), the development of an individual as directed by his genetic code, and *epigenesis* (E), the development of an individual through learning processes (nervous system, immune system) influenced both by the genetic code (the innate) and by the environment (the acquired). In analogy to nature, the space of *bio-inspired systems*, that is, computing systems that seek inspiration from biology for the development of novel computational paradigms, can also be partitioned along the same three axes.

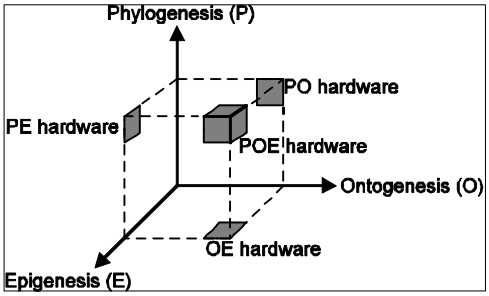


Figure 1: The three axes of the POE model for bio-inspired systems, and their convergence.

The ultimate goal of this project is the development of architectures and design methodologies that can be applied to the three models of bio-inspiration, either separately or together. The project is ambitious in two respects: first, because it is rare to find projects concerned with more than a single axis of biological inspiration, and second, because for the first time it will attempt to develop an automated approach to the design of bio-inspired machines.

In this section, we shall analyze the state of the art in the domain of bio-inspired *digital hardware* by examining the three axes of the POE model separately.

### 2.1.1. Phylogenetic systems

On the phylogenetic axis we find systems inspired by the processes involved in the evolution of a species through time. The process of evolution is based on alterations to the genetic information of a species (the genome), occurring through two basic mechanisms: crossover and mutation.

Crossover (or recombination) is directly related to the process of sexual reproduction. When two organisms of the same species reproduce, the offspring contains genetic material coming from both parents, and thus becomes a unique individual, different from either parent. Mutation consists of random alterations to the genome caused either by external phenomena (i.e., radiation or cosmic rays) or by chemical faults which occur when the two genomes merge. Often resulting in non-viable offspring (in fact, the great majority of mutations have no effect, as they act on unused parts of the genome) mutations are nevertheless vital for the process of evolution, allowing “leaps” in evolution which would be impossible to achieve by merging the genomes of individuals of the same species.

Both these mechanisms are, by nature, non-deterministic. This represents both their strength and their weakness, when applied to the world of electronics. It is a strength, because they are fundamentally different from traditional algorithms and thus are potentially capable of solving problems which are intractable by deterministic approaches. It is a weakness, because computers are inherently deterministic (it is very difficult, for example, to generate a truly random number, a basic requirement for non-deterministic computation, in a computer).

Even with this disadvantage, algorithms which exploit phylogenetic mechanisms are carving themselves a niche in the world of computing. These algorithms, commonly referred to as *e*volutionary algorithms (a label which regroups domains such as genetic algorithms [30], evolutionary programming [11], and genetic programming [22][23]), are usually applied to problems which are too ill-defined or intractable by deterministic approaches, and whose solutions can be represented as a finite string of symbols (the equivalent of the biological genome). An initial, random population of individuals (i.e., of genomes), each representing a possible solution to the problem, is iteratively "evolved" through the application of mutation (i.e., random alterations of a sequence) and crossover (i.e., random merges of two sequences). The resulting sequences are then evaluated on the basis of their efficiency in solving the problem (the fitness function) and the best (fittest) solutions are in turn evolved. This approach is not guaranteed to find the best possible solution to a given problem, but can often find an "acceptable" solution more efficiently than deterministic approaches.

The phylogenetic axis has already provided a considerable amount of inspiration to the development of computer systems. To date, however, its impact has been felt mostly in the development of software algorithms. The application of evolutionary approaches to hardware, in fact, could not begin until the introduction, in the early nineties, of *reconfigurable circuits (FPGAs)* [53], that is, of circuits that could be functionally modified at runtime. The arrival of such circuits, by providing the needed platform for experimentation, heralded the birth of the field called *evolvable hardware*.

Setting aside the application of evolutionary mechanisms to analog hardware (a very active field), Higuchi [18] pioneered the attempt to apply evolutionary strategies to the synthesis of digital electronic circuits. Since then, evolutionary approaches have been applied to many different designs: control circuits for prostheses [19] and autonomous robots [10][24], pattern recognition circuits [61], frequency discriminators [51], sorting networks [28], hash functions [4], etc. The Logic Systems Laboratory has contributed to the field notably with the development of Firefly [13], a machine capable of evolving a solution to the problem of synchronizing a one-dimensional cellular automaton, a relatively simple task that nevertheless represents the first example of hardware capable of evolving completely on-line, i.e. without the assistance of a computer.

While promising, the field of evolvable hardware has shown some important limitations, most notably where *scalability* [16][56] is concerned: it is quite simple (and useful for many applications) to evolve small circuits, but the algorithms used so far do not scale well to larger systems. The issue of scalability is the focus of the latest research efforts in the domain, and an important part of the research we are conducting for the POEtic project, described in the next section.

### 2.1.2. Ontogenetic systems

The phylogenetic and, as we will see, epigenetic axes of the POE model cover the great majority of existing bio-inspired systems. The *development* of a multi-cellular biological organism, however, involves a set of processes which do not belong to either of these two axes. These processes correspond to the *growth* of the organism, i.e., to the development of an organism from a mother cell (the *zygote*) to the adult phase. The zygote divides, each offspring containing a copy of the genome (*cellular division*). This process continues (each new cell divides, creating new offspring, and so on), and each newly formed cell acquires a functionality (i.e., liver cell, epidermal cell, etc.) depending on its surroundings, i.e., its position in relation to its neighbors (*cellular differentiation*).

Cellular division is therefore a key mechanism in the growth of multi-cellular organisms, impressive examples of massively parallel systems: the $6 \times 10^{13}$ cells of a human body, each a relatively simple element, work in parallel to accomplish extremely complex tasks (the most outstanding being, of course, intelligence). If we consider the difficulty of programming parallel computers (a difficulty which has led to a decline in the popularity of such systems for general computing tasks), biological inspiration could provide some relevant insights on how to handle massive parallelism in silicon.

A fundamental feature of biological organisms is that each cell contains the blueprint for the entire organism (the *genome*), and thus can potentially assume the functionality of any other cell: no single cell is indispensable to the organism. In fact, cells are ceaselessly being created and destroyed, a mechanism at the base of one of the most interesting properties of multi-cellular organisms: *healing*.

The mechanisms of ontogeny (cellular division and cellular differentiation), unlike those of epigenesis and phylogeny, are completely deterministic, and are thus, in theory, more easily adaptable to the world of digital circuits (which is by nature deterministic). In spite of this, the ontogenetic axis has been almost completely ignored by computer scientists, despite a promising start in the fifties with the work of John von Neumann, who developed a theoretical model of a universal constructor, a machine capable of constructing any other machine, given its description [57]. Given a description of itself, the constructor can self-replicate, a process analogous to cellular division.

Unfortunately, electronic circuits in the 1950s were too primitive to allow von Neumann's machine to be realized, and the concept of self-replicating machines was thus set aside. Probably the main obstacle to the development of self-replicating machines was the impossibility of physically creating self-replicating hardware. In fact, such machines require a means to transforming information (i.e. the description of a machine) into hardware, and such a means was definitely unpractical until the introduction of sufficiently powerful reconfigurable logic circuits in the early nineties.

In the meantime, another aspect of the development of living organisms has historically attracted a considerable amount of attention since the early days of computer science: *morphogenesis*, that is, the formation of the organism as a *structure*. Since Thompson's [52] and Turing's [42][54] classical works on modeling morphogenesis, several scientists have developed interesting mathematical models: Lindenmayer's L-systems [25], Kitano's grammar-based encoding [20], Gruau's cellular encoding [15], and Paun's membrane computing [37] are but a few of the most remarkable attempts at finding models capable of mimicking the structural development of biological organisms (it should be noted, however, that the *functional* aspect of an organism is usually ignored in these systems, a serious drawback when considering architectures for computing systems rather than for modeling biological processes).

More recently, the developmental process of biological organisms has been attracting a growing amount of attention as a solution to the above-mentioned problem of *scalability*: living beings are, after all, astounding examples of scalable cellular computing, where extremely complex global behavior is obtained through the cooperation of numerous simple processing elements. This aspect of bio-inspiration is the main focus of the Embryonics project, as well as one of the major research axes of the POEtic project, and as such will be described in detail in the following section.

### 2.1.3. Epigenetic systems

The human genome contains approximately $3x10^9$ bases. An adult human body contains something like $6x10^{13}$ cells, of which approximately $10^{10}$ are neurons, with $10^{14}$ connections. Obviously, the genome cannot contain enough information to completely describe all the cells and synaptic connections of an adult organism.

There must therefore exist a process which allows the organism to increase in complexity as it develops. Since we already know that the additional information cannot come from within the cell itself, the only possible source of additional information is the outside world: the growth and development of an organism must be influenced by the environment, a process which is most evident in the development of the nervous, immune, and endocrine systems. Following A. Danchin's terminology [5], we have labeled this process *epigenesis.*

Epigenetic mechanisms have already had considerable impact on computer science, and particularly on software design, notably through the concept of learning. The parallel between a computer and a human brain dates to the very earliest days of the development of computing machines, and led to the development of the field known as artificial intelligence [59].

Hardware systems based on epigenetic processes have also become common, in the form of artificial neural networks (ANNs) [1][17], two-dimensional arrays of processing elements (the neural cells) interconnected in a highly complex pattern. Each cell receives a set of input signals from its neighbors, processes them according to a given, implementation-dependent function, and propagates the results through the network. This process is a good approximation of the mechanisms exploited by biological neurons (for example, as in nature, some inputs signals have a greater impact, or weight, in the computation of a neuron's output value).

Three aspects of neural networks define their performance:

- The *processing elements* or *neurons* are the constituent parts of a neural network. They consist of a set of *synapses*, which receive and modify the inputs to the neuron depending on their *synaptic weight*; of an *adder*, to sum all the weighted inputs; and of an *activation function*, applied to the weighted sum of inputs, which both limits that output's amplitude and determines whether or not the neuron should *fire* (i.e., send out an output through the network). The activation functions can be either *deterministic* or *probabilistic*. In the latter case the neurons, referred to as *stochastic neurons*, more closely mimic the observed uncertainty of biological neuron activity, but generally result in networks that learn slowly and with difficulty.
- The *network topology* defines how the neurons are interconnected. Essentially, there are four main connectional methods used with neural networks: *feedforward*, feedback (*recurrent*), *lateral* (inhibition), and *self-excitatory*. Based on these methods, many parallel connectionist paradigms have been proposed: the *feedforward multi-layer perceptron* (MLP), the most widely used topology for hardware implementations; the *Hopfield network*, normally associated with

*Hebbian learning*; the *competitive network*, in which the neurons compete for given input patterns by using *lateral inhibitory* and *self-excitatory* connections, and its refinement the *self-organizing feature map* (SOM); the *radial basis function network* (RBF), in which each layer uses a different kind of neuron, therefore removing homogeneity from the network; the *adaptive resonance theory network* (ART), which combines neurobiological and psychological findings, introducing the idea of *short-term* and *long-term memories*; and many more.

- The *learning algorithm*, that is, in general, the method used by the network to optimize its ability to perform a given task. The majority of existing learning methods perform this optimization by changing the weights associated with individual neurons. Depending on the algorithm used, neural networks can either learn for a finite period of time and then perform the required task without further learning (*off-line* learning) or else continue to learn whilst performing the task (*on-line* learning). Among the most common learning approaches, we will mention: *supervised* learning, which involves the calculation and feedback of error signals, i.e., of the difference between the (known) desired and the actual response of the network to a given *training set* of inputs; *reinforcement* learning, a variant of supervised learning where a critic (*heuristic*) informs the network of whether it has been performing correctly or not, and where the network corrects itself until the desired input/output mapping has been achieved; *unsupervised* learning, in which the networks update their parameters without external guidance; *phylogenetic* learning, where evolutionary algorithms are applied to a population of neurons (this kind of learning represents in fact a first combination of two axes of bio-inspiration, phylogenesis and epigenesis).

When implemented in hardware, many designs use software running on a host computer, or external control circuitry, to carry out the learning procedure. This is referred to as *off-chip* learning, and is popular mostly because of its relative simplicity and versatility. In fact, when the learning is an integral part of the hardware design, as is the case for *on-chip* learning, the complexity and connectivity of each neuron (and hence of the network) are vastly increased, and it can be relatively difficult to alter the learning method once the circuit is fabricated. One of the main goals of this project is then to *simplify* the generation of large arrays of neurons capable of learning *on-chip* at hardware speeds, so as to increase the versatility and rapidity of neural hardware implementations.

The digital implementations of neural networks can be divided into two main categories: dedicated ASIC solutions and reconfigurable logic (FPGA) implementations. Even by limiting the discussion to the latter category, more interesting from the standpoint of the proposed project, the implementations described in the literature are too numerous to describe. We shall then limit ourselves to mentioning one implementation, the FAST (Flexible Adaptable-Size Topology) algorithm developed in the Logic Systems Laboratory by A. Perez-Uribe [39][40], as it is one of the very few attempts at merging the ontogenetic (development) and epigenetic axes into a single system.

FAST is a neural network with a dynamically reconfigurable structure. Traditional ANNs have a fixed interconnection structure, and only the weights associated with the connections can be modified. By implementing the network using reconfigurable logic, FAST achieves on-line learning capabilities and can exploit an adaptable topology (two features which are essential to obtain true learning systems, as opposed to “learned” ones). While FAST is not the first neural network based on an adaptable topology [12][31], it is unique in that it does not require intensive computation to reconfigure its structure, and can thus exist as a stand-alone machine which could be used, for example, in real-time control applications.

## 2.2. Past and ongoing research

My research, past and ongoing, in the field of bio-inspired computing machines has been conducted within two main research projects: the *Embryonics* (embryonic electronics) project, started in 1993 in the Logic Systems Laboratory and still in progress through funding from several sources, and the *Reconfigurable POEtic Tissue* (or *POEtic* for short) project, started in 2001 under the aegis of the Information Society Technologies (IST) program of the European Community and involving a collaboration between the Swiss Federal Institute of Technology in Lausanne (EPFL, Switzerland), the University of York (England), the Technical University of Catalunya (UPC, Spain), the University of Glasgow (Scotland), and the University of Lausanne (Switzerland).

This section presents an outline of these projects, which represent the main scientific and technological basis that will be exploited by the proposed project. Further information on the two projects can be found in the literature ([27][34][35][36][44][45][46][47] and particularly [26] for the Embryonics project, [8][41][48][49] and particularly [55] for the POEtic project, along with the project’s website at http://www.poetictissue.org).

### 2.2.1. The Embryonics (embryonic electronics) project

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 biological organisms, 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:

- *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 is copied into the daughter cell(s).
- *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) [38].

To our knowledge, the only research groups trying to explicitly draw inspiration from the ontogenetic and healing processes are the Swiss Federal Institute of Technology at Lausanne, Switzerland [26][27][45][46], and the University of York, UK [34][35][36], both involved in the Embryonics 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.

On the other hand, 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.

The morphogenetic approaches discussed in the previous section have, as we have mentioned, one important drawback with respect to the design of bio-inspired machines: they concentrate, as the

name says, on the definition of the *shape* of an organism, ignoring another, fundamental aspect, the *function*. The main goal of the Embryonics project, on the other hand, is the design of highly-robust integrated circuits, endowed with 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.

The final architecture of the Embryonics project is based on four hierarchical levels of organization which, described from the bottom up, are the following (Figure 2):

- The basic primitive of our system is the *molecule*, a programmable logic element of a novel FPGA, consisting essentially of a multiplexer associated with a programmable connection network. The multiplexer is duplicated to allow the detection of faults and the associated repair logic allows a first (molecular) level of *fault tolerance*. The logic function of each molecule is defined by its molecular code (a string of approximately 30 bits).
- A finite set of molecules makes up a *cell*, essentially a processor with the associated memory. Using a hardware subsystem present in each molecule, the *topology* of the cell (that is, its width, height, and the presence and positions of columns of spare molecules for molecular fault tolerance) can be defined at runtime depending on the application. The architecture of the cells includes mechanisms capable of automatically exploiting the presence of *spare cells* should a fault impair the operation of a cell, implementing a second (cellular) level of fault tolerance.
- A finite two-dimensional array of cells makes up an *organism*, an application-specific multiprocessor system. The function of the cells within the organism is determined through a set of [X,Y] coordinates (implementing cellular differentiation), relying on the presence of the entire genome of the organism within each cell.
- The organism can itself self-replicate, giving rise to a *population* of identical organisms, the highest level of the Embryonics hierarchy.

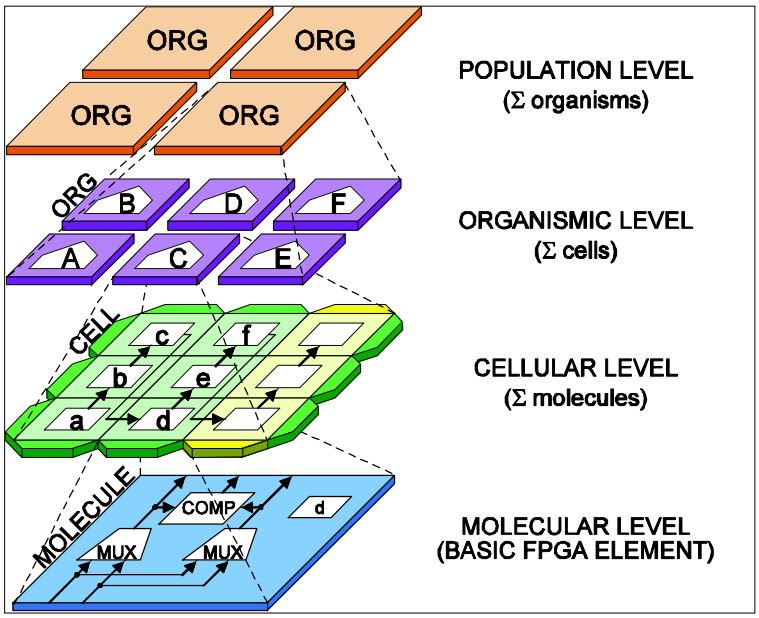


Figure 2: The hierarchical organization of Embryonics architectures.

The architectures developed in the Embryonics project have been validated through a series of prototypes, and finally completely realized in the *BioWall* [44][47], a large-scale reconfigurable computing tissue capable of interacting with its environment via a set of touch-sensitive elements coupled with LED displays. The final implementation has the respectable dimensions of approximately 5.5mx0.9m. The entire tissue contains 3200 molecules, each consisting of a four-color 8x8-pixel display, a transparent touch-sensitive membrane, and a reconfigurable circuit (a Xilinx Spartan XCS10XL FPGA). Each molecule is interconnected with its four cardinal neighbors. The BioWall consists of about 32 million reconfigurable logic gates, 3200 binary inputs, and more that 50 thousand LEDs, and has been used to implement several bio-inspired cellular architectures.

The Biowall, as a highly versatile prototyping platform for cellular systems, and the Embryionics project, as a source of inspiration for the development of novel architectures, will be an invaluable contribution to the success of the present project.

### 2.2.2. The Reconfigurable POEtic Tissue (POEtic) project

As we have seen in the preceding section, biological inspiration in the design of computing machines finds its source in essentially three biological models: phylogenesis (P), the history of the evolution of the species, ontogenesis (O), the development of an individual as directed by his genetic code, and epigenesis (E), the development of an individual through learning processes (nervous system, immune system) influenced both by their genetic code (the innate) and by the environment (the acquired). These three models share a common basis: a one-dimensional description of the organism, the *genome*.

While each of these models, taken separately, has to a greater or lesser extent been used as a source of inspiration for the development of computing machines, their amalgamation into a single artifact is a challenge yet to be met. The hypothesis behind the POEtic project is that such a machine could be created with the support of a *flexible computational substrate*. The goal of the POEtic project is then the development of such a substrate, inspired by the evolutionary, developmental and learning mechanisms of biological systems.

This final tissue will be the essential substrate for the creation of POE-based artifacts (POEtic machines), capable of evolution (P model), of growth, self-repair, and self-replication (O model), and of learning (E model). The tissue will be composed of digital logic elements whose functionality and interaction can be programmed at runtime to construct a tissue able to handle a given task. The overall structure of the circuit will thus be not unlike that of an FPGA (Field-Programmable Gate Array), but many improvements will be needed to adjust the architecture to the implementation of adaptive systems. For example, scalability must be guaranteed and a direct pin-to-pin connection between devices must be sufficient to augment the size of the tissue.

As indicated, the functional elements are the core of the POEtic tissue. They constitute flexible elementary processing elements that integrate evolutionary, self-repairing, self-replicating and adaptive features. The challenge is considerable: at the heart of all adaptive systems, and thus of living artifacts, is the concept of *chang*e, in the sense of functional and structural alterations occurring during the operational lifetime of an organism. While capable of off-line adaptation, conventional FPGAs are incapable of altering their own configuration at runtime (a partial exception to this rule is the Xilinx 6200 family, now discontinued). Because of its complexity and of its limited application to conventional circuit design, this capability, indispensable for the design of truly adaptive systems, is currently unavailable in off-the-shelf commercial FPGAs. A first feature required by the POEtic circuit will therefore be the capability of altering its own structure while operating.

Adaptation and specialization are key features of all biological organisms. As a consequence, no bio-inspired architecture can be rigid, but must rather be able to adapt and specialize depending on the desired application. The POEtic tissue will exploit reconfigurable logic to the fullest extent to allow for these characteristics: the bio-inspired machines will consist of two-dimensional arrays of *cell*s, where each cell is a small, fully reconfigurable processor. However, as in the Embryonics project, the cells will not represent the hardware of the tissue, which will be implemented by a *molecular substrate*: as organic cells are constituted by molecules, so artificial cells will be constituted by the programmable logic elements of the POEtic circuit.

The cellular architecture is of particular interest to the proposed project. In its current implementation, it is divided into three different layers (Figure 3): the genotype layer, the configuration or mapping layer, and the phenotype layer.

The *genotype layer* contains the entire genetic information of the organism, which consists essentially of:

- A *set of operator*s, which define the set of all possible functions for the cell. The quantity and functionality of the operators depends essentially on the application. For example, if the phenotype layer implements an array of neurons, one could imagine that the operators will consist of the internal parameters for the neuron, together with a set of initial connection weights. Instead, if the phenotype implements, say, a sound processing surface, the operators will consist of different filters and sound generation functions.
- A *differentiation tabl*e, used to determine which operators will be implemented in which cell. The table contains a compact, perhaps redundant, representation of the operators to be implemented by the cells. The main advantage of maintaining a separate table storing this information is its *size*: the table will represent an extremely compact representation of the structure of the system, and evolutionary mechanisms in hardware will be applied to the table, rather than to the operators themselves. The differentiation table also plays an important role in the mapping layer, as described below.

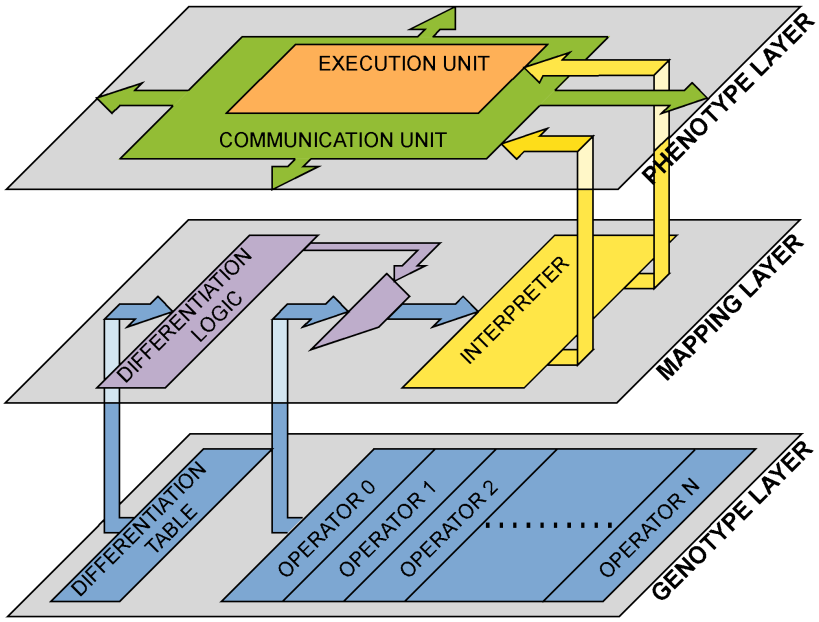


Figure 3: The three layers of POEtic cellular architectures.

The key feature of the genotype layer lies in the presence of a set of *dedicated connection*s, used to provide user access to the differentiation tables of all the cells in the system. The advantage of this kind of approach lies in the presence of an *on-chip microcontrolle*r, whose function is to access the tables and to manipulate their contents through a set of user-defined evolutionary mechanisms.

The *configuration layer* of the POEtic systems is designed to implement the processes of cellular differentiation and cellular division (growth), the bases of the ontogenetic model (the fault-tolerance aspect of the model is also particularly present in this layer, but is in general distributed throughout the layers of the array). In its simplest conception, the function of the configuration layer is to select which operator will be implemented by the cell. Each cell must be able to identify its position within the array. A set of coordinates, incremented in each cell and transmitted along the horizontal and vertical axes as in Embryonics systems, is the simplest approach. Once the cell has established its position, it can use this information to select from the differentiation table which operator to express. The operator will then be interpreted to generate the appropriate control signals for the phenotype layer.

However, this approach is only a very basic solution to achieve cellular differentiation, and does not exploit many of the most interesting features of the development process. For example, it is limited to a *one-to-one mapping* between genotype and phenotype. From an evolutionary standpoint, this approach imposes considerable disadvantages in the application of phylogenetic mechanisms. More complex mappings, based on redundant coding, are being researched.

Also, the coordinate-based approach is not well suited to the implementation of processes that want to draw inspiration from the natural process of *growt*h, a key mechanism in the development of biological organisms. More complex differentiation mechanisms, based on L-systems [25] and on cell-signaling gradients [41], are currently under study.

The *phenotype layer* is probably the most application-dependent layer of the three. If the final application is a “conventional” neural network, the phenotype layer of the cell will simply consist of an artificial neuron. The architecture of the neuron is a choice left to the user. The POEtic project will concentrate on *spiking neurons* [8][29] but this choice does not imply that the tissue will be limited to such a model. However, since the POEtic tissue is meant to allow the implementation of bio-inspired systems that do not necessarily involve exclusively neural-like cells, a more general-purpose phenotype architecture will be required. The definition of such an architecture is one of the main targets of the proposed project.

The definition of three separate layers for the different models of bio-inspiration has the advantage of being able to select which of the models are to be used for a given problem. This feature is made possible by the complete programmability of the systems: all the three layers are implemented on the same *molecular substrate*, a programmable digital logic circuit that incorporates dedicated features for all three models. The implementation of the layers in programmable logic adds the possibility of adapting the architecture of the system to the application, a flexibility that will allow the definition of an *integrated design environment*, the objective of the proposed project, as detailed in the next section.

## 2.3. Detailed research plan

The ultimate goal of the proposed project is the realization of a unified framework for the realization of bio-inspired machines. The paradigm we propose to develop is based on two elements:

- A *universal architecture* for the implementation of bio-inspired systems, allowing the realization of machines inspired by any or all of the three models of bio-inspiration (evolution, development, and learning). Based on the structure defined in the POEtic project (Figure 3), the architecture will implement each model in a separate layer, thus allowing the user to choose which models are desirable for a given application.
- A *design environment* based on the above-mentioned architecture, to reduce the design and development times required for the hardware implementation of bio-inspired machines. The environment will provide a *customizable design flow* able to automatically generate the hardware required for the implementation of a set of basic bio-inspired mechanisms, allowing the user to concentrate on those mechanisms that need to be investigated.

The contribution of the Embryonics and POEtic projects will be invaluable, both from a scientific and from a practical point of view. The Embryonics project will have an impact notably where the development (ontogenetic) model is concerned, and the BioWall will be an ideal platform for the verification of the first prototypes. The POEtic project, on the other hand, will provide the hardware support for the architectures to be developed in the project, as well as have an impact on the selection of the most interesting bio-inspired approaches available for hardware implementation.

### 2.3.1. A novel architecture for bio-inspired systems

Nature is *change*. Adaptation and specialization are key features of all biological organisms. As a consequence, no bio-inspired architecture can be rigid, but must rather be able to adapt and specialize depending on the desired application. This requirement, a major obstacle until recently, can today be satisfied by exploiting *reconfigurable logic* [53], a technology based on VLSI circuits that can be *programmed* to implement different hardware architectures. The circuits can then be reset and re-programmed at will, a feature that makes them a vital resource in the implementation of bio-inspired systems (the POEtic tissue will unavoidably consist essentially of a reconfigurable logic array). The architecture we plan to develop exploits reconfigurable logic to the fullest extent: our electronic organisms will consist of two-dimensional arrays of *cells*, where each cell is a small, fully reconfigurable processing unit. The flexibility of programmable circuits will allow the cells to specialize depending on the application.

Integrating reprogrammability into an architecture, however, is far from trivial. The heterogeneity of the applications that can be implemented poses many design problems. A novel kind of architecture needs to be developed to fully exploit the potential of reconfigurable logic for bio-inspired systems.

The architecture we plan to develop is based, on the one hand, on the hierarchical structure of the Embryonics systems, where to each application corresponds an *organism* composed of a variable number of *cells*, themselves realized by a *molecular* level, the hardware substrate of the system, and on the other hand on the cellular structure of the POEtic project, based on three separate layers, which will then be mapped onto a single molecular tissue (Figure 4).

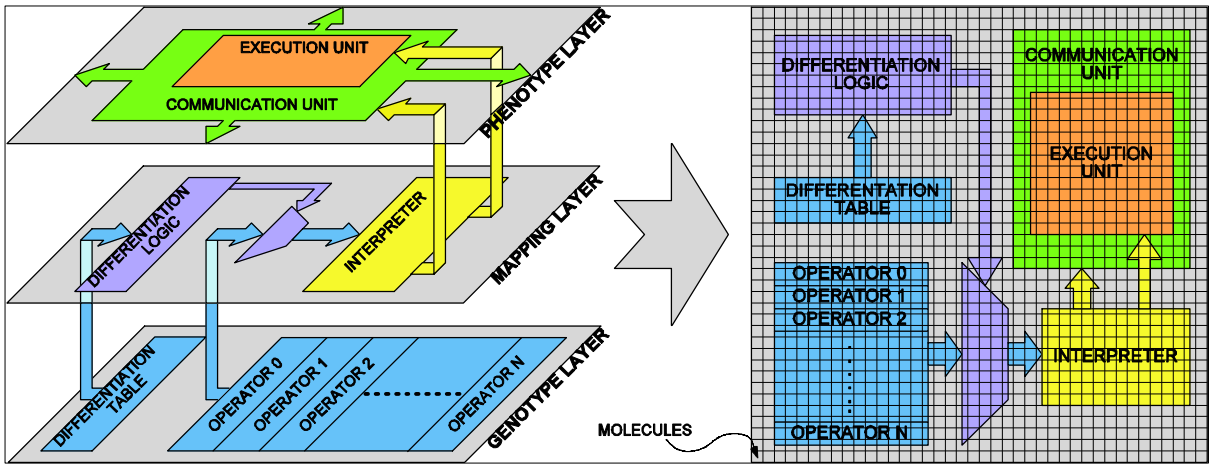


Figure 4: Mapping of the cells onto the molecular tissue.

The separation of the cells' architecture into three layers is not merely an abstract concept: each layer implements a different axis of bio-inspiration and their structure will need to reflect this separation by incorporating very different design mechanisms and interfaces. The challenge is considerable: not only must the final architecture be *universal*, in the sense of being capable of implementing all the three models, but it must also be *universally efficient*, in the sense of being capable of implementing the models together or separately (for those applications that do not require one or two of the models) without undue waste of resources. It should, for example, be able to implement a neural network without any evolutionary or ontogenetic features or a developmental approach independent of any learning, and to do so without excessive hardware overhead.

In the next subsections, we will briefly describe the salient features of a possible architecture for each of the three layers, keeping in mind of course that the exact definition of the architecture is one of the main *results* of the proposed project, and that therefore the ideas presented in this proposal are simply the starting point of the research (and as such are heavily influenced by the work being done in the Embryonics and POEtic projects).

*The genotype layer (evolution)*

The genotype layer is fundamental for bio-inspired systems, as it is charged of storing the genetic information (genome) of the organism. In true multi-cellular organisms, each cell contains the genome of the entire organism, a redundancy that allows the implementation of mechanisms such as cellular differentiation and healing (self-repair).

Often, when dealing with bio-inspired systems, the definition of genome is somewhat reductive: normally, it is defined simply as the information required by the phenotype layer to operate and is implemented as a look-up table or a simple random-access memory. In reality, the genome of biological systems is a highly complex and structured construct, with an active role that has little to do with a simple memory. A more accurate and useful electronic equivalent for the biological genome would be composed of several distinct subsets (Figure 5):

- A subset whose function is to determine the topology of the cell, that is, its width and height (as well as the eventual presence and position of spare molecules for self-repair). In analogy to the *polymerase enzyme*, a molecule that allows cellular replication in biological systems, we can call this part the *polymerase genome* (PG).
- A subset whose function is to assign the logic function of each molecule within the cell, that is, the configuration of the elements of the FPGA involved in the implementation of the cell. In each cell of every living being, the genome is translated sequentially by a chemical processor, the *ribosome*, to create the proteins needed for the organism's survival. The ribosome itself consists of molecules, whose description is a major part of the genome. By analogy, we can then call this part of the genetic information the *ribosomic genome* (RG).

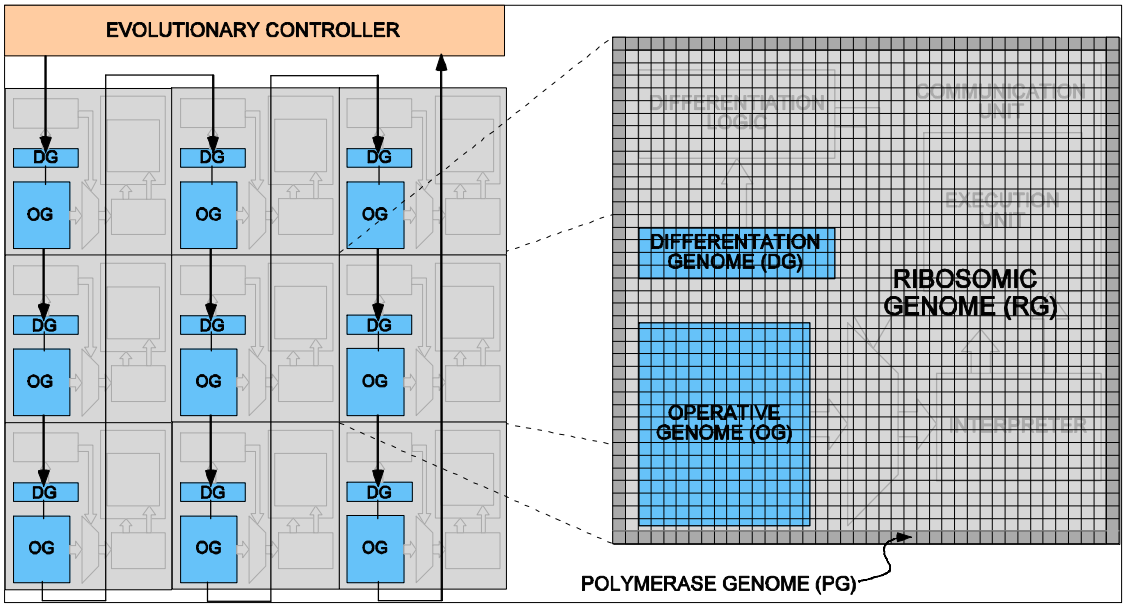


Figure 5: The genome subsets in the genotype layer of a cell and the evolutionary controller.

- A subset whose function is to control the development and differentiation process. This function is similar to that realized in biological systems by the *homeoboxes* or *HOX genes*, recently found to define the general architecture of living beings [58], and by the *switch genes*, which select which part of the genome will be executed depending on the cell's position within the organism. We shall call this subset the *differentiation genome* (DG).
- A subset whose function is to define the operation of the phenotype layer. This subset, which we can call the *operative genome* (OG) of the cell and which is often considered the *only* genetic information in artificial bio-inspired systems, can assume many different forms, ranging from a simple look-up table (storing, for example, the initial parameter of a neuron-like cell) to a fully structured program as in Embryonics systems (in which case, of course, the polymerase genome will have to include an interpreter for the appropriate language).

This separation of the genome's structure in several smaller parts has a number for interesting consequences for the design of bio-inspired machines, and notably for the implementation of evolutionary mechanisms. As mentioned in the section devoted to the state of the art (2.1.1), in fact, one of the main obstacles to the success of evolution-based systems is the problem of *scalability* [16][56]: given today's computational power, the evolutionary paradigms developed to date cannot handle the genome lengths required to describe a hardware system. By separating the genome into smaller subsets, each subset becomes a potential field of application for evolutionary algorithms.

Moreover, separately evolving each of the genome elements defined above has the advantage of being able to apply the most appropriate evolutionary approach to each element:

- The polymerase genome and the ribosomic genome determine the physical structure of the circuit within the tissue. Practical considerations based on their size, on the risk of short circuits, and on the time required for the configuration of the tissue make these subsets ill-adapted for on-line hardware evolution. It will still be possible to apply evolutionary algorithms to these parts of the genome, of course, but such an approach will probably be best applied in software via simulation (more on this subject in the next section).
- The differentiation genome is compact and does not directly impact the structure of the circuit. It does, on the other hand, determine the topology of the system at the cellular level, and as such has a strong impact on the development phase of an organism (indeed, evolutionary approaches applied to this part of the genome often mention ontogenesis [9][21][32][60]). Because of its features, this part of the genome can potentially be evolved directly in hardware [41], through the use of a dedicated *evolutionary controller* that will be placed next to the POEtic tissue and will be able to directly access the relevant configuration memory.
- The evolution of the operative genome depends, of course, on its application-specific structure. Should its size be small enough, hardware evolution could be applied. Otherwise, the software approach will be more appropriate.

*The mapping layer (development)*

The issue of development has been attracting an increasing amount of attention (see, for example, [7] and [14]) in the field of bio-inspired designs as a possible solution to the problem of scalability. In some respects, the complexity of biological organisms resides less in the genome itself than in the development process that leads to the formation of large multi-cellular structures. Long ignored for lack of an adequate technological support, the issue of development is coming to the forefront in the search for bio-inspired machines.

This research trend is not surprising, considering the advantages of a well-designed development mechanism: the redundancy introduced by such mechanisms (notably by the presence of a full genome in each cell) is compensated by the additional features that can be introduced, particularly when considering that technological progress is ceaselessly reducing the cost of redundancy.

Among the most interesting features made possible by a developmental approach, we will mention three in particular:

- The presence of a mapping layer that implements development will allow the introduction of *complex genotype/phenotype mappings*. In fact, while in its simplest form the mapping layer can consist of a simple set of [X,Y] coordinates to uniquely identify a cell within the organism (Figure 6), more complex mappings will permit a more versatile approach, better suited to the realization of adaptive and learning systems. For example, a development process could allow the implementation of systems whose structure is not defined at design time, but depends on the environment in which the machine has to operate (for example, the signals received through a robot's sensors could have an impact on the architecture of its control circuit).

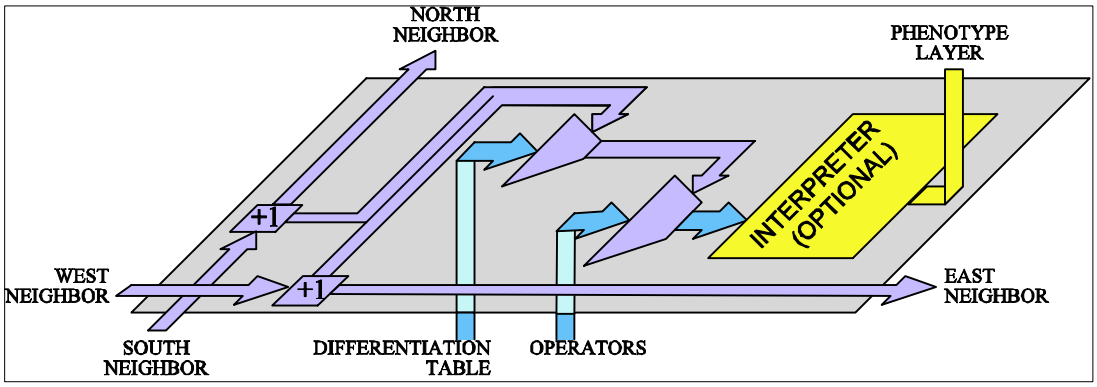


Figure 6: A simple coordinate-based architecture for the mapping layer.

- The implementation of mechanisms for cellular division and differentiation can simplify the design of large multi-cellular (multi-processor) systems: defining the configuration for the entire programmable logic surface (a time-consuming task, particularly for multi-chip systems) will no longer be necessary, as the configuration of a single cell will automatically propagate to fill the available hardware.
- The presence of the genome in each cell and the capability to differentiate can be invaluable tools for another extremely interesting feature of biological organisms: healing. Not only the redundant information can open the way for simple reconfiguration-based approaches [33][34][45], inspired by the biological concept of *stem cells* [38], but the identical structure of the cells in the system can be exploited to test the correct operation of the system at a higher level, using mechanisms inspired by the immune system of complex biological organisms [2][3].

All these advantages notwithstanding, research on developmental approaches is still in its early stages: only the latest generations of FPGAs (and soon the POEtic tissue) provide a sufficient amount of programmable logic to make such approaches viable. The architecture we propose to implement, together with the design environment described in the next section, will allow researchers to easily develop, verify, and compare different developmental mechanisms, giving a decisive contribution to this very promising research area.

*The phenotype layer (learning)*

The phenotype of a bio-inspired architecture is undoubtedly the most application-dependent layer. Even when limited to neuron-like cells, the variety of implementations of neural networks presents a considerable challenge for the development of a universal architecture. Of course, our system is emphatically *not* limited to neurons, as the user should be able to implement models of bio-inspiration that do not involve learning. We shall however use neurons as a benchmark for the performance of our system, as they present complex architectural challenges that make their implementation particularly challenging.

Referring back to the section on the state of the art (2.1.3), we can observe that neural networks consist of three parts: the neurons, the topology of the network, and the learning algorithm. While the latter does not have an immediate impact on the hardware architecture of the network (aside from its impact on the first two parts), both the neurons and the topology need to be implemented directly in hardware. The task is far from trivial: many neuron architectures are extremely computation-intensive (in fact, many are ill-adapted for a hardware implementation), involving complex operations on the incoming data, and the topologies of neural networks are invariably characterized by very dense connectivity patterns. These features are a major obstacle to hardware implementations, but on the other hand they make neural networks a very good benchmark for our universal architecture.

To reduce the overall size of the system, the project will investigate the possibility of exploiting *on-line arithmetic* [6] for the implementation of both the computational cores (the most common arithmetic operations have all been successfully implemented with this kind of approach) and the communication interface (the use of serial communication will be required by the sheer number of connections in a neural network and it could be very interesting, for example, to incorporate on-line multipliers to the connections to perform the weight computation as the data is being transmitted).

The subdivision of the phenotype into a computational core and a communication interface is a universal feature for cellular systems. We will therefore structure our phenotype architecture into two sub-circuits (Figure 7):

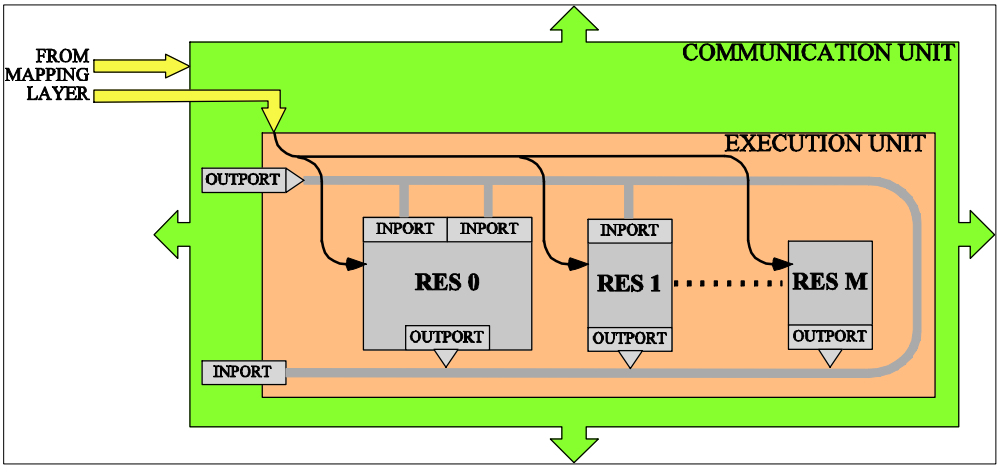


Figure 7: A possible architecture for the phenotype layer.

- An execution unit, implementing the computational core of the cell. In order to make the cell capable of implementing different applications, the architecture cannot be specific, but must rather be able to adapt to the requirements of the task. The approach that we are planning to develop relies on the presence within the core of a set of application-dependent *resources* (e.g., adders, counters, etc.). The resources are defined by the user at design time, and are accessed through a set of input and output ports, interconnected by a system bus. This kind of architecture has two main advantages, in the context of the proposed project: it can support specialization and adaptation through the modification of the resources depending on the application and, as we shall see, it can be integrated into a design environment.
- A communication unit, implementing the data-exchange interface between cells as well as with the environment. The complexity of this unit will, of course, be extremely application-dependent. In view of the massive amounts of connectivity of bio-inspired systems, the communication unit will undoubtedly have to rely on *serial communication*, and will be seen by the execution unit as another resource, with standard input and output ports (an approach that will simplify the implementation and control of the data exchange algorithm) A major issue in the implementation of neural networks in hardware, the interconnect topology will exploit a set of *dedicated intercellular communication channels* provided by the POEtic tissue.

### 2.3.2. An integrated design environment for bio-inspired systems

Defining an architecture for bio-inspired machines is only the first step of the proposed project, which aims at providing an integrated environment for their design. Such an environment should include a *customizable design flow* allowing the user to concentrate on the desired approaches. As we have seen, the three models of bio-inspiration are well-suited to different architectural layers, and the user should be able to exploit the models separately, as well as together:

- The phylogenetic model acts on the genetic material of a cell. Typically, in the architecture defined above, it could be used to find and select the genes of the cells for the *genotype layer*.
- The ontogenetic model concerns the development of the individual. It should act mostly on the *mapping layer* of the cell, implementing cellular differentiation and growth. In addition, ontogenesis will have an impact on the overall architecture of the cells where self-repair (healing) is concerned.
- The epigenetic model modifies the behavior of the organism during its operation, and is therefore best applied to the *phenotype layer*.

Defining separate layers for the different models has a number of advantages, notably in that it allows the user to decide whether to implement any or all models to solve a particular problem: a researcher wishing to study aspects of the developmental process might well not be interested in either evolution or learning. A design environment should take into account this requirement, and allow the *efficient* implementation of machines inspired by each model of bio-inspiration.

The key feature of the proposed environment is the *automatization* of a large part of the design flow. The intervention of the user will be required only to decide which bio-inspired models are required by the application. The environment then will automatically take care of the implementation of the models. This automatization is a necessary ingredient to reduce development time, a crucial consideration when dealing with hardware systems.

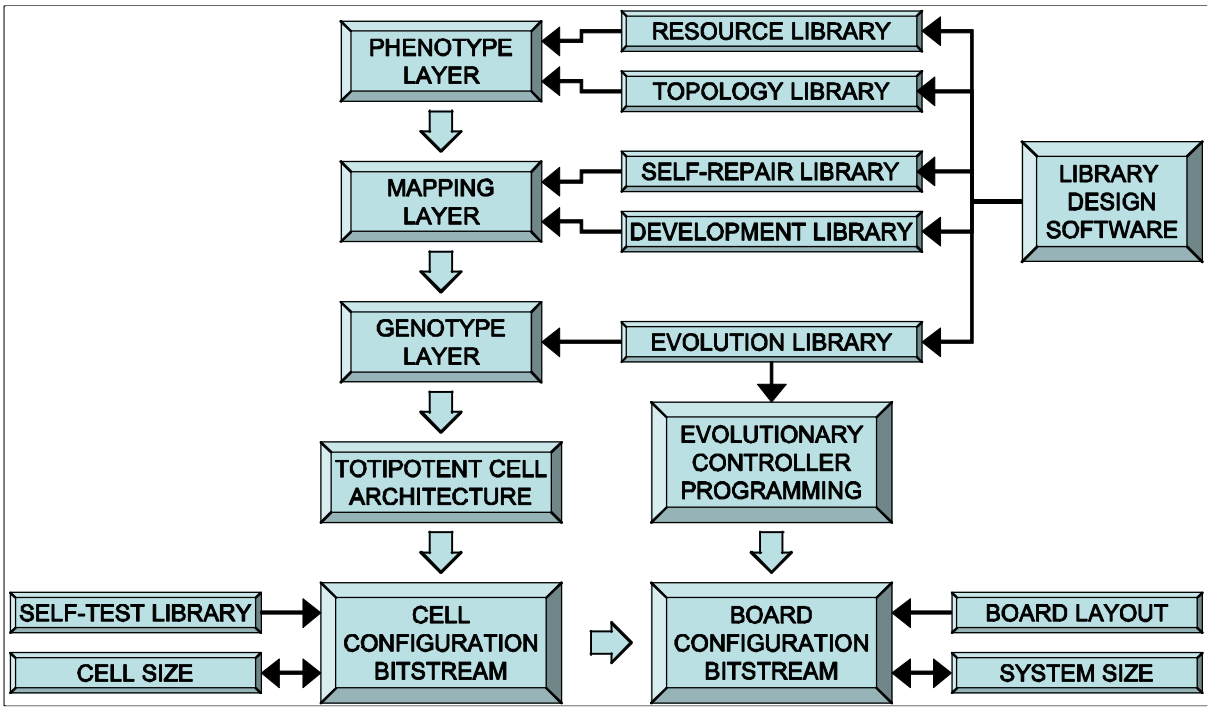


Figure 8: Design flow for the development of bio-inspired systems.

A typical *application-driven design flow* for a bio-inspired system implemented on the kind of architecture we have presented could then be (Figure 8):

- Based on the application, the user defines the functionality of the phenotype layer (the layer that actually realizes the application) by accessing a *resource library*. The library contains a set of standard modules that correspond to the resources mentioned in the previous section, e.g. adders, multipliers, shifters, etc. In general, the user should be aware of the general requirements of an application, at least where the basic operations are involved. It should noted that the user does not, at this stage, need to specify the *organization* of the system, that is, where (in which cells) the specific resources will be used within the array. The communication topology is defined similarly, by accessing a *topology library*, containing a set of possible communication patterns for the array. Once again, at this stage, depending on the application, the user might not wish to define the exact topology, in which case the maximum possible interconnection network should be specified.
- Once the components of the phenotype layer have been specified, the user can advance to the mapping layer. By accessing a *development library*, the system can implement different differentiation mechanisms. Should development not be an issue, a simple coordinate-based mechanism can be used. At this stage, unless an evolutionary approach is to be implemented, the user decides which resources are implemented by which cells and what topology the array should implement, thus defining the parameters for cellular differentiation. In addition, a *self-repair* library allows the user to decide whether a reconfiguration mechanism should be put in place to provide fault tolerance, and, if so, what kind of mechanism should be used.
- After the mapping layer is defined, the user must configure the genotype layer, and more particularly the operative and differentiation genomes. If no evolution is planned for the system, then the complexity of the operative genome depends essentially on the resources defined for the phenotype layer. Some resources will need a simple set of parameters to operate, in which case the operative genome will be implemented by a simple look-up table, while more complex resources might require a structured program, implying a larger operative genome and the presence of an interpreter for the genome language (typically, this will be a tradeoff of *hardware/software codesign*). The differentiation genome, on the other hand, will be limited to a small look-up table to determine which resource corresponds to a cell's coordinates. On the other hand, if an on-line hardware evolutionary mechanism is to be implemented, the user will access an *evolution library* containing a set of such mechanisms. In the most extreme case, depending on the evolutionary approach selected, it will be up to the evolutionary mechanism to determine on-line the topology and the organization of the system.

Once the three layers have been defined and the bio-inspired mechanisms selected, the environment will automatically continue the design process, with little or no user intervention.

- The three layers will be merged together to define the architecture of a *totipotent cell*, that is, of a cell capable of implementing any of the functions required by the application. This cell will then contain a copy of each resource, a connection network to implement any cell topology, and the operative genome of all the cells in the organism.
- Accessing the description of each of the elements selected by the user (and of course the architecture of the FPGA that will implement the array), the environment will then generate the *configuration bitstream* for a *single* totipotent cell. In other word, it will generate the ribosomic genome of the cell. From the ribosomic genome, the *size* of the cell (i.e., its height and width in terms of reconfigurable elements of the FPGA) can be determined, and from this information the environment can define the last part of the genome, the polymerase genome. At this stage, if required, the environment can activate the self-test features provided by the reconfigurable platform (in the case of Embryonics, for example, the molecular self-test features can be exploited by the cellular level).
- If an evolutionary approach has been selected for the genotype layer, the environment will also generate the program that will be executed by the evolutionary controller to implement the desired mechanism. This program, separate from the genome of the organism, represents the algorithm to be used to evolve a population of organisms on a single chip or board. It will contain the information required for the evolution of the system, as well as the instructions necessary to test the fitness of each individual.
- From the bitstream of the cell and the program of the evolutionary controller, the environment can then generate the configuration bitstream for the prototyping board used to implement the system. Depending on the surface of reconfigurable logic available, on the features of the evolutionary controller (e.g., the amount of memory at its disposal), the environment will determine the final size of the system (typically, how many cells will be implemented) and make this information available to the user. At this stage, of course, the user will be able to decide to modify the choices made for the cellular layers, should the system not meet the expectations (e.g., if a neural network is too small for the desired application). A functional description of the system will also be provided, allowing the user to simulate the behavior of the circuit.

The goal of the project being to provide an environment both fast and easy to use, the design flow will of course be implemented through a *graphical interface*, allowing even users not familiar with hardware design to access the main features of the architecture and to test the performance of bio-inspired systems on the application of interest.

Naturally, research on bio-inspired systems is not limited to architectures known *a priori*. In fact, the most interesting aspect of this kind of research is the conception and design of novel approaches, to be tested and verified by experimentation. In other words, an environment meant for research cannot be limited to closed libraries of known functions and mechanisms, but must allow researchers to develop novel approaches and to incorporate them into the design flow by modifying the component libraries.

To allow this kind of access, *library design software* will be developed, allowing users familiar with basic hardware design to add new elements to the libraries. The design process will of course depend on the kind of element to be added: elements affecting the phenotype layer would typically be designed using standard logic-level tools, and transformed into compatible macros within the library; elements affecting the mapping layer would probably be best described as algorithms, automatically transformed into the corresponding hardware circuit by the design software; finally, elements affecting the genotype layer should have a minimal hardware component (in general, all operative genomes are stored in standard memories) but will need to include the program to be executed by the evolutionary controller in order to implement the algorithm of choice. Another interesting feature of the library design software will be the possibility of *evolving*, using standard software approaches, those parts of the genome that are too complex for hardware evolution.

In conclusion, the development for a proper design methodology for bio-inspired systems is the logical next phase for a domain that is finally reaching maturity. Systems inspired by biological mechanisms are carving themselves a niche in several areas of computation, but the solutions adopted often cannot be generalized for the lack of a universal architecture capable of integrating the different approaches. Through the development of such an architecture and of a user-friendly design environment capable of drastically reducing the design time of bio-inspired hardware, this project should have a considerable impact on the development of the field.

## 2.4. Project timetable

- **First year**
  - Analysis of the requirements of evolutionary mechanisms for the genotype layer.
  - Analysis of the requirements of development mechanisms for the mapping layer.
  - Analysis of the requirements of learning mechanisms for the phenotype layer.
  - Analysis of the final implementation of the POEtic tissue to identify its features and the mechanisms used access such features.
- **Second year**
  - Development of a universal architecture for the genotype layer.
  - Development of a universal architecture for the mapping layer.
  - Development of a universal architecture for the phenotype layer.
  - Design of a prototyping board to validate the architecture.
- **Third year**
  - Verification of the universal architecture on known real-world applications.
  - Design of a component library for the three architectures, together with the tools necessary to access and integrate the components into a single design.
  - Design of the graphical user interface to allow the integration of the components into the system.
- **Fourth year**
  - Design of the low-level tools required to transform the graphical representation of the system into configurations for the prototyping board.
  - Design of the graphical user interface allowing on-line access to the prototyping board, so as to permit the implementation of real-time adaptive systems.
  - Verification of the design environment on real-world applications.

## 2.5. Importance of the planned research

The relevance of the proposed project to the scientific community can be evaluated from several different points of view:

- The project represents the first attempt ever made at developing an architecture capable of efficiently implementing in hardware the salient features of all the main models used in the design of bio-inspired machines: evolution, development, and learning. Separating the architectural layers to which each model is applied will allow researchers to reduce the number of parameters to be modified, so as to be able to more easily evaluate the impact of each model on the performance of the system for a given task.
- By providing an integrated design environment and by allowing users to fully exploit the features of the tissue developed within the POEtic project, the project will let hardware to be modified as easily as software, a long-awaited breakthrough in a field where the performance of software is often a bottleneck, but where hardware development times are usually too restrictive. A graphical user interface will allow researchers to easily control the environment in which the bio-inspired systems will operate, simplifying repetitive tasks such as the fitness evaluation for evolvable machines or the training and learning phases for neural architectures.
- The outcome of the project, a universal architecture coupled with a customizable design flow, will be an invaluable research and prototyping tool not only for our group, but for all scientists interested in the development of bio-inspired systems. The ability to quickly and automatically generate all the accessory mechanisms will allow researchers to concentrate on the features they wish to investigate. The presence of dedicated hardware to simplify the access to the features of the circuit will increase its versatility and simplify the verification process for novel bio-inspired approaches.
- Last, but not least, the project will allow the EPFL to place itself at the forefront in the field of bio-inspired hardware: the new group will integrate a team that is ready to achieve the critical size necessary to become world leaders in the domain. The experience that I have accumulated in years of research on ontogenetic machines, coupled with the excellent synergy between Embryonics, the POEtic project, and the proposed project, provide an outstanding opportunity for research and place me in an ideal position to direct such a project.

# Bibliography


[1] Arbib, M.A., ed. *Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[2] Bradley, D.W., Tyrrell, A. "Multi-Layered Defence Mechanisms: Architecture, Implementation, and Demonstration of a Hardware Immune System". *Proc. 4th Int. Conf. on Evolvable Systems (ICES2001)*, LNCS 2210 (2001), 140-150.

[3] Canham, R.O., Tyrrell, A.M. "A Multilayered Immune System for Hardware Fault Tolerance within an Embryonic Array". *Proc. 1st Int. Conf. on Artificial Immune Systems (ICARIS 2002)*, Canterbury, UK, September 2002.

[4] Damiani, E., Tettamanzi, A., Liberali, V. "On-Line Evolution of FPGA-Based Circuits: A Case Study on Hash Functions". *Proc. 1st NASA/DoD Workshop on Evolvable Hardware (EH1999)*, 1999, 26–33.

[5] Danchin, A. "Stabilisation fonctionnelle et épigénèse: une approche biologique de la genèse de l'identité individuelle". In *L'identité*, Grasset, 1977, 185-221.

[6] Denyer, P., Renshaw, D. *VLSI Signal Processing: A Bit-Serial Approach*. Addison-Wesley, 1985.

[7] Edwards, R.T. "Circuit Morphologies and Ontogenies". *Proc. 2002 NASA/DoD Conf. on Evolvable Hardware*, IEEE Computer Society Press (2002), 251-260.

[8] Eriksson, J., Torres, O., Mitchell, A. Tucker, G., Lindsay, K., Halliday, D., Rosenberg, J., Moreno, J.M., Villa, A. "Spiking Neural Networks for Reconfigurable POEtic Tissue". *From Biology to Hardware: Proc. 5th Int. Conf. on Evolvable Hardware (ICES03)*, Trondheim, Norway, March 2003. Submitted.

[9] Fiesler, E. "Comparative Bibliography of Ontogenic Neural Networks". *Proc. Int. Conf. on Artificial Neural Networks (ICANN94)*, 1994.

[10] Floreano, D., Urzelai, J. "Evolutionary Robots with On-Line Self-Organization and Behavioral Fitness". *Neural Networks* 13, 431-443.

[11] Fogel, D.B. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.

[12] Fritzke, B. "Growing Cell Structures – a Self-Organizing Network in k Dimensions". *Artificial Neural Networks 2*, 1992.

[13] Goeke, M., Sipper, M., Mange, D., Stauffer, A., Sanchez, E., Tomassini, M. "Online Autonomous Evolware". *Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES96),* LNCS 1259, Springer-Verlag, Berlin, 1997, 96-106.

[14] Gordon, T.G.W., Bentley, P.J. "Towards Development in Evolvable Hardware". *Proc. 2002 NASA/DoD Conf. on Evolvable Hardware*, IEEE Computer Society Press (2002), 241-250.

[15] Gruau, F. "Genetic Systems of Boolean Neural Networks with a Cell Rewriting Developmental Process". In D. Whitley and S. D. Schaffer, eds., *Combination of Genetic Algorithms and Neural Networks*. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[16] Haddow, P. C., Tufte, G. "Bridging the Genotype-Phenotype Mapping for Digital FPGAs". *Proc. 3rd NASA/DoD Workshop on Evolvable Hardware (EH2001)*, 2001, 109–123.

[17] Hassoun, M.H. *Fundamentals of Artificial Neural Networks*. The MIT Press, Cambridge, MA, 1995.

[18] Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H., Furuya, T. "Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine". *Proc. 2nd Int. Conf. on Simulation of Adaptive Behavior (SAB92)*, MIT Press, 1993, 417-424.

[19] Kajitani, I., Hoshino, T., Nishikawa D., Yokoi, H., Nakaya, S., Yamauchi, T., Inuo, T., Kajihara, N., Iwata, M., Keymeulen, D., Higuchi, T. "A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI". *Proc. 2nd Int. Conf. on Evolvable Systems (ICES 98)*, LNCS 1478, Springer-Verlag, Berlin, 1998, 1-12.

[20] Kitano, H. "Building Complex Systems using Developmental Process: An Engineering Approach". *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98)*, 1998, 218-229.

[21] Kitano, H. "Designing Neural Networks using Genetic Algorithms with Graph Generation System". *Complex Systems*, 4:461-476, 1990.

[22] Koza, J.R. *Genetic Programming II*. The MIT Press, Cambridge, MA,1994.

[23] Koza, J.R. *Genetic Programming*. The MIT Press, Cambridge, MA, 1992.

[24] Krohling, R., Zhou, Y., Tyrrell, A.M. "Evolving FPGA-Based Robot Controllers using an Evolutionary Algorithm". *Proc. 1st Int. Conf. on Artificial Immune Systems (ICARIS 2002)*, Canterbury, UK, Sept. 2002.

[25] Lindenmayer, A. "Mathematical Models for Cellular Interaction in Development, Parts I and II". *Journal of Theoretical Biology*, 18:280-315, 1968.

[26] Mange, D., Sipper, M., Stauffer, A., Tempesti, G.. "Towards Robust Integrated Circuits: The Embryonics Approach". *Proceedings of the IEEE*, vol. 88, no. 4, April 2000, 516-541.

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

[28] Masner, J., Cavalieri, J., Frenzel, J., Foster, J.A. "Representation and Robustness for Evolved Sorting Networks". *Proc. 1st NASA/DoD Workshop on Evolvable Hardware (EH1999)*, 1999, 255–261.

[29] Maya, S., Reynoso, R., Torres, C., Arias-Estrada, M. "Compact Spiking Neural Network Implementation in FPGA". *Field Programmable Logic Conference (FPL2000)*, Austria, 2000, 270-276.

[30] Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[31] Moreno, J.M. *VLSI Architectures for Evolutive Neural Models*. Ph.D. Thesis, Universitat Politecnica de Catalunya, Barcelona, 1994.
[32] Murakawa, M., Yoshizawa, S., Kajitani, I., Higuchi, T. "Evolvable Hardware for Generalized Neural Networks". *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI97)*, MorganKauffman, 1997, 1146-1151.
[33] Negrini, R., Sami, M.G., Stefanelli, R. *Fault Tolerance through Reconfiguration in VLSI and WSI Arrays*. MIT Press, Cambridge, MA, 1989.
[34] Ortega, C., Tyrrell A.M. "Reliability Analysis in Self-Repairing Embryonic Systems". *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*, Pasadena, CA, July 1999, 120-128.
[35] Ortega, C., Tyrrell, A. "MUXTREE revisited: Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays". LNCS 1478, Springer-Verlag, Berlin (1998), 206 -217.
[36] Ortega, C., Tyrrell, A.M. "Self-Repairing Multicellular Hardware: A Reliability Analysis". In D. Floreano, J.-D. Nicoud, F. Mondada, eds., *Advances in Artificial Life*, LNAI 1674, Springer-Verlag, Berlin, 1999, 442-446.
[37] Paun, G. "Computing with Membranes". *Journal of Computer and System Sciences*, 61(1):108-143, 2000.
[38] Pearson, H. "The regeneration gap". *Nature*, vol. 414, 22 November 2001, 388-390.
[39] Pérez-Uribe, A. *Structure-Adaptable Digital Neural Networks*. Ph.D. Thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, 1999.
[40] Pérez-Uribe, A., Sanchez, E. "Structure-Adaptable Neurocontrollers: A Hardware-Friendly Approach". *Proc. Int. Work-Conference on Artificial and Natural Neural Networks (IWANN97)*, 1997.
[41] Roggen, D., Floreano, D., Mattiussi, C. "A Morphogenetic System as the Phylogenetic Mechanism of the POEtic Tissue". *From Biology to Hardware: Proc. 5th Int. Conf. on Evolvable Hardware (ICES03)*, Trondheim, Norway, March 2003. Submitted.
[42] Saunders, P. T., ed. *Collected Works of A. M. Turing: Morphogenesis*. North-Holland, Amsterdam, 1992.
[43] Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Pérez-Uribe, A., and Stauffer, A. "A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems". *IEEE Transactions on Evolutionary Computation*, 1:1 (1997), 83-97.
[44] Stauffer, A., Mange, D., Tempesti, G., Teuscher, C. "BioWatch: A Giant Electronic Bio-Inspired Watch". In D. Keymeulen, A. Stoica, J. Lohn, and R. Zebulum, eds., *Proc. 3rd NASA/DoD Workshop on Evolvable Hardware (EH-2001)*, IEEE Computer Society Press, Los Alamitos, CA, 2001, 185-192.
[45] Tempesti, G. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes*. Thesis No.1827 (1998), Swiss Federal Institute of Technology, Lausanne.
[46] Tempesti, G., Mange, D., Stauffer, A. "A Robust Multiplexer-Based FPGA Inspired by Biological Systems". *Journal of Systems Architecture: Special Issue on Dependable Parallel Computer Systems*, 43(10), 1997.
[47] Tempesti, G., Mange, D., Stauffer, A., Teuscher, C. "The BioWall: an Electronic Tissue for Prototyping Bio-Inspired Systems". *Proc. 2002 NASA/DoD Conference on Evolvable Hardware*, IEEE Computer Society Press, Los Alamitos, CA, 221-230.
[48] Tempesti, G., Roggen, D., Sanchez, E., Thoma, Y., Canham, R., Tyrrell, A.M. "Ontogenetic Development and Fault Tolerance in the POEtic Tissue". *From Biology to Hardware: Proc. 5th Int. Conf. on Evolvable Hardware (ICES03)*, Trondheim, Norway, March 2003. Submitted.
[49] Tempesti, G., Roggen, D., Sanchez, E., Thoma, Y., Canham, R., Tyrrell, A., Moreno, J.-M. "A POEtic Architecture for Bio-Inspired Systems". *Proc. 8th Int. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life VIII)*, Sydney, Australia, December 2002.
[50] Teuscher, C. *Turing's Connectionism. An Investigation of Neural Network Architectures*. Springer-Verlag London. September 2001.
[51] Thompson, A. "An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics". Evolvable Systems: From Biology to Hardware". *Proc. of the 1st Int. Conf. on Evolvable Systems (ICES 98)*, 1998, 390–405.
[52] Thompson, D.W. *On Growth and Form*. Cambridge University Press, Cambridge, UK, 1961.
[53] Trimberger, S., ed. *Field-Programmable Gate Array Technology*. Kluwer Academic, Boston, 1994.
[54] Turing, A.M. "The Chemical Basis of Morphogenesis". *Philosophical Transactions of the Royal Society of London*, B 237:37-72, 1952.
[55] Tyrrell, A.M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J.M., Rosenberg, J., Villa, A. "POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware". *From Biology to Hardware: Proc. 5th Int. Conf. on Evolvable Hardware (ICES03)*, Trondheim, Norway, March 2003. Submitted.
[56] Vassilev, V.K., Miller, J.F. "Scalability Problems of Digital Circuit Evolution". *Proc. 2nd NASA/DoD Workshop on Evolvable Hardware (EH2000)*, 2000, 55–64.
[57] Von Neumann, J. *The Theory of Self-Reproducing Automata*. A. W. Burks, ed. University of Illinois Press, Urbana, IL, 1966.
[58] Watson, J.D., Hopkins, N.H., Roberts, J.W., Argetsinger Steitz, J., Weiner, A.M. *Molecular Biology of the Gene*. Benjamin Cummings, Menlo Park, CA, 4th ed., 1987.
[59] Winston, P.H. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 3rd ed., 1992.
[60] Yao, X. "Evolving Artificial Neural Networks". *Proceedings of the IEEE*, 87(9):1423-1447, Sept. 1999.
[61] Yasunaga, M., Nakamura, T., Kim, J.H., Yoshihara, I. "Kernel-Based Pattern Recognition Hardware: Its Design Methodology using Evolved Truth Tables". *Proc. 2nd NASA/DoD Workshop on Evolvable Hardware (EH2000)*, 2000, 253–262.