POETIC
IST-2000-28027
Reconfigurable POEtic Tissue

Deliverable No: 10
Description of the structure and functionality of the O circuit
Covering period 1.3.2002-31.8.2002

Report Originator: Gianluca TEMPESTI, Yann THOMA
Report Version: 1.2
Report Preparation Date: 31/08/2002
Classification: Internal
Project start: September 1, 2001 **Duration:** 36 months.
Project Co-ordinator: Professor AM Tyrrell
Partners: University of York, Technical University of Catalunya, University of Glasgow, University of Lausanne, Swiss Federal Institute of Technology Lausanne.

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

1 Introduction

POEtic tissues are designed to implement a vast range of bio-inspired systems, covering all the major axes of research in the domain (Phylogenesis, Ontogenesis, and Epigenesis) [16, 18]. In this report, we will present the approaches we have selected to realize one of the three biological models that inspire circuit design: ontogenesis, that is, the development and growth of multi-cellular organisms, along with the fault tolerance that such structures imply. However, the final goal of the POEtic project has not been ignored, and the mechanisms presented in this report have been selected so as to be compatible with all three models and to take into account the conclusions of previous project workpackages (WP1, WP2, WP3) and deliverables (D1-D9).

In the next section, we will introduce the main features and the global architecture of POEtic systems as developed in our project, with particular emphasis on those aspects that are directly involved in the ontogenetic axis. We will then examine in some detail the three main areas of research currently under study for this part of the project, namely inter-cellular communication, development and fault tolerance, and present our solutions for their implementation in silicon.

2 POEtic Hardware

POEtic systems draw inspiration from the multi-cellular structure of complex biological organisms. As in biology, at the heart of our systems is a *hierarchical* structure (Fig. 1) ranging from the molecular to the population levels. In particular, the subjects of this report relate mainly to the cellular and organismic levels, with a brief foray into the molecular level for fault tolerance.

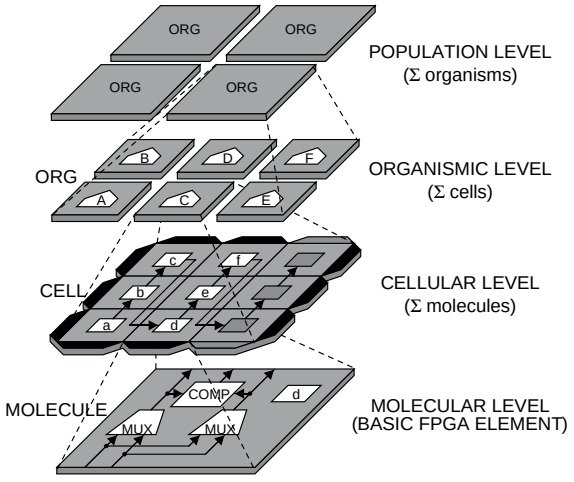

Fig. 1. The organizational layers of an electronic organism.

It is important to note, in this context, that the only hardware (and hence fixed) level of the POEtic systems is the molecular level, implemented as a novel digital FPGA. All other levels are *configurations* for this FPGA, and thus can be seen as *configware*, that is, entities that can be structurally programmed by the user. As a consequence, most of the mechanisms described in this report, and notably ontogenetic development and cellular fault tolerance, can be altered (or removed) to fit a particular application or a novel research approach.

The current (being configware, it could be modified in the future as new solutions are researched) architecture of the POEtic cells is described in Fig. 2. It consists of three logical layers (physically, of course, the three layers are "flattened" onto the molecular FPGA), each with a specific function within the domain of bio-inspired systems. In this report, we shall concentrate on the *mapping layer*, as, together with the *differentiation table*, it is the layer directly concerned by the ontogenetic development of our machines (fault tolerance, as we shall see, is involved in the system at all levels).

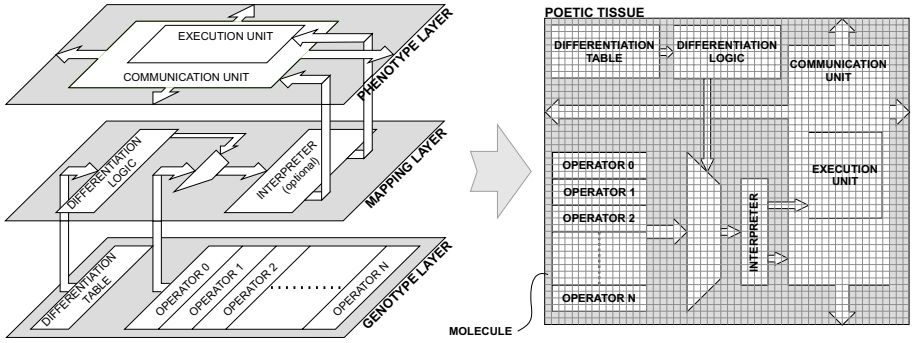

Fig. 2. The three logical layers of an electronic cell, mapped on the molecular tissue.

To illustrate with a practical example the structure of POEtic circuits, we shall use a very simple application: a *timer* (Fig. 3) capable of counting minutes and seconds. This architecture has the twin advantage of being conceptually very simple and of being easily broken down into a cellular structure.

The first step in the design of a POEtic application is in fact to identify a logical division into cells. For this example, the division is obvious: tens of minutes, units of minutes, tens of seconds, units of seconds, which incidentally also correspond to the four outputs of the system (while the only input is the 1Hz synchronization clock). In general, the difficulty of this step depends of course on the application, and is probably the main criterion to determine whether an application is amenable to being implemented in a POEtic circuit.

Next, it is necessary to identify, on the basis of the cellular structure of the application, the operators or functions required by the cells of the system. For the timer, a minimum of two operators is required: a counter by six (for the tens of seconds and minutes) and a counter by ten (for the units of seconds and

minutes). In addition, the communication pattern of the cells must also be fixed: for the timer of Fig. 3, the output of the counters is sent to the north, while the synchronization signals are sent out to the west.

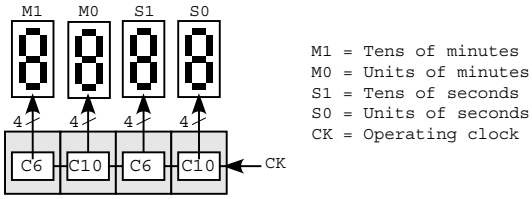

Fig. 3. Cellular implementation of a simple four-digit timer.

At this stage, the architecture of the cells can be defined. This step implies essentially a design choice to decide where to place the complexity of the system. In fact, in a typical example of hardware/software co-design, the complexity can reside in the genome and in its interpreter, Embryonics-style [10], or in the phenotype. For example, in the case of the timer, the first approach would require a genome implemented by a structured program, used to control a simple 4-bit register in the phenotype. Or, in the second approach, the genome could well be reduced to a single number, used as a parameter for a programmable counter in the phenotype layer. Probably a more efficient solution in this case, this second approach would practically eliminate the need for an interpreter of the genome, and shift all the (in this case limited) complexity to the phenotype layer.

Of course, the set of operators represents only part of the genetic information of the organism, the rest being made up by the differentiation table used to assign operators to the cells in the system. The definition of a mechanism to implement differentiation in a POEtic system is the subject of the next section.

3 Inter-Cellular Communication

The four-digit timer is a connection-poor circuit: the only signals are the synchronization signal passing from right to left and the outputs to the north of the values of the counters. This kind of architecture can easily be implemented using only local communication busses between cells. On the opposite end of the spectrum, an artificial neural network (ANN) requires a massive number of connections: a local network connecting a cell with its four or eight neighbors is not sufficient to adequately exploit the features of ANNs.

Since the implementation of neural network can be considered the main application for POEtic circuits, this limitation is not acceptable. We have therefore developed two novel mechanisms of cell communication: the first is based on *packet communication*, while the second implements a *dynamic routing algorithm*. Depending on the results of our analysis, the molecular level (the hardware) will be designed so as to directly support one or both of these mechanisms.

3.1 Packet Communication

The concept behind the packet communication approach is to find a way to implement a simple data-exchange algorithm within each cell. This task can be simplified by analyzing the communication requirements of our applications. Notably, in the case of neural networks, cells have many inputs, but a single output; in other words, they commonly use a one-to-many communication pattern.

On the basis of this observation, it would not be efficient for a cell to explicitly send a message to every other cell that needs its output. In the packet communication approach, the cell would send out a single message that propagates through the network in a ring (Fig. 4), letting each cell decide whether to read the signal off the ring or to ignore it, depending on the circuit's communication pattern (a pattern that can easily be altered at runtime, allowing on-line structural adaptation of the network).

A number of parameters are required to implement packet communication in a cellular array (Fig. 5). Notably, since the ring passes through every cell in the array, some sort of *unique identifier* is required to differentiate the cells. This identifier could (or could not, as we will see in the next section) be the same used to implement the development mechanism. In addition, each cell needs a (small) memory to store the identifiers of the cells it is connected to, so as to be able to decide whether to read the values passing on the ring.

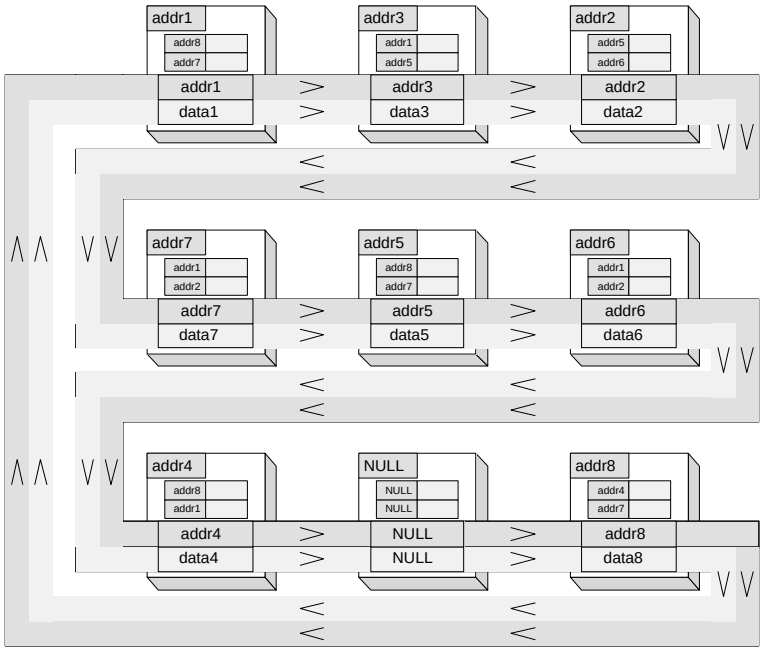

Fig. 4. The packet communication approach in a 9-cell organism.

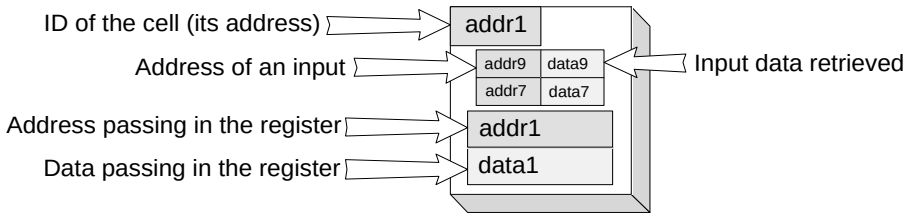

Fig. 5. Information required by each cell for packet communication.

Of course, the packet communication mechanism can be implemented as a configuration for any programmable digital FPGA. However, such an implementation would necessarily be complex and inefficient, unless the FPGA is specifically designed for the task. Considering that the POEtic circuits are meant to implement this kind of complex interconnection patterns, we are currently designing a molecule that can implement packet communication with minimal effort.

The ring will then be programmed onto molecules configured in a dedicated mode of operation, and implemented with a double shift register (i.e., two synchronous shift registers side by side). One register contains the address of the source, and the other one stores the associated data. The flow process of the entire system is very simple, based on two phases: one for the treatment, and one for communication. Each cell computes its output and places it on the ring (treatment phase), where it is propagated through the array by the shift register (communication phase). As the values are shifted, each cell keeps the values it requires as inputs for the next treatment phase, which starts as soon as the propagation is complete (i.e., the data have completed a full circle within the ring).

The principal limitation of this approach is, of course, speed, as a communication phase is associated to each treatment phase. Since it is implemented as a shift register, the time required by the communication phase through the array is proportional to the number of cells in the circuit.

From the point of view of scalability, on the other hand, there is no real problem, as the molecules will be able to use 8-bit, 16-bit, or 32-bit addressing. The user will simply have to decide at design time, depending on the requirements of the application, the size of the addresses to be used in the circuit, letting then the layout tools automatically insert the appropriate mechanisms in the configuration bitstream of the FPGA.

An advantage of this approach is that every cell can communicate with every other cell, a process that can be very difficult to realize with physical connections. Another advantage concerns fault tolerance: should a cell die, it can automatically use the ring to send a message warning of the presence of a fault, and a spare cell (which can be placed anywhere within the circuit, since packet communication uses explicit addressing) can take over its functionality, repairing the circuit.

3.2 Dynamic Routing

The second approach we are investigating is based on a dynamic routing algorithm developed by J.-M. Moreno and described in [11]. This algorithm allows a regular array of routing units to dynamically create paths between sources and targets (Fig. 6), and requires only to made aware of the position of the sources and of the targets. Each source can, in turn, be connected to as many targets as needed, if a sufficient number of channels are available. The automatic routing process requires, in a NxM-cell array, a time less than $2 * (N + M)$.

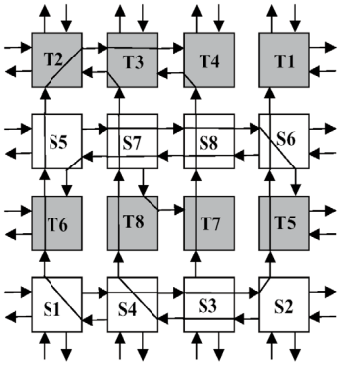

Fig. 6. Dynamically-created paths in a 4x4-cell array.

Applied to our cellular arrays, the routing process needs to be realized for every cell in the system. However, the speed issue is much less relevant than for packet communication, as the routing process needs to be executed only rarely, either after the end of the configuration and differentiation of the cells or after a reconfiguration of the circuit following self-repair.

In Moreno's approach, as described in his article, each routing unit has four neighbors. While efficient from the point of view of hardware utilization, simulations have shown that, for highly-connected systems, the risk of congestion with this kind of connectivity is high. As a consequence, we are developing a similar mechanism, based however on eight-neighbor connections (i.e., each routing unit is connected with its 8 nearest neighbors), so as to improve the overall connectivity of the circuit.

In practice, inter-cellular communication using the dynamic routing approach is realized by introducing a hardware *communication layer* (Fig. 7) superimposed onto the molecular layer. The layer consists of regularly-spaced routing units (one every 4 or 9 molecules), accessible to the molecules via dedicated I/Os. Only one of the 4 or 9 molecules underneath a routing unit can have access to the unit at one time (access permissions are handled at configuration time by the place/route software), so that for a cell composed of NxM molecules, we can theoretically have up to $\lfloor \frac{N}{2} \rfloor * \lfloor \frac{M}{2} \rfloor - 1$ inputs and one output.

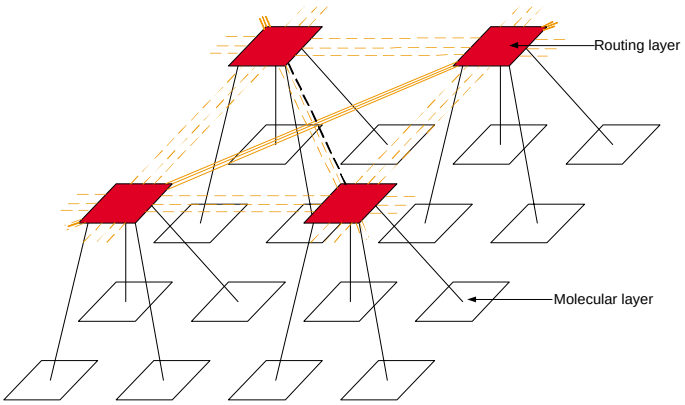

Fig. 7. Additional communication layer for dynamic routing.

The dynamic routing approach presents several interesting features. First of all, once the routing is in place, after configuration, the routing channels behave as wires, and are therefore much faster than the connection ring of the packet communication approach. Secondly, the mechanism allows complex long-distance interconnection patterns, very useful for neural networks(the routing process can be seen as somewhat analogous to axon growth). Finally, the presence of the additional hardware layer for communication does not adversely impact the performance of the FPGA for applications that do not require such long-distance routing.

3.3 Implementation Issues

Both the approaches we presented can be implemented in our circuit without excessive overhead, if the molecules are designed so as to support the mechanisms directly in hardware. Adapting the molecular level to implement these approaches is necessary for dynamic routing, and packet communication would be too costly to realize without dedicated hardware.

Each communication systems has its strengths and weaknesses. In general, the dynamic routing algorithm probably presents the best tradeoff in terms of versatility and performance, but does introduce some limitations in terms of connectivity. On the other hand, the packet communication approach can implement fully connected networks, with a non-negligible cost in terms of speed.

As the design of our molecular level continues, we hope to introduce support for both communication systems. We shall undoubtedly design the additional communication layer for dynamic routing, the most efficient solution in the majority of cases, but we shall also try to add dedicated features within each molecule to support packet communication, so as to accelerate and compact its implementation.

4 Development in the POEtic Tissue

The modelization in electronic hardware of the developmental process of multi-cellular biological organisms is a challenging task, for reasons that should be obvious: on one hand, the sheer complexity of even the simplest multi-cellular biological organisms is orders of magnitude beyond the current capability of electronic circuits, and on the other hand the development of such organisms implies, if not the creation, at least the transformation of matter in ways that cannot yet be realized in digital or analog hardware.

And yet these same considerations make such a process extremely interesting for hardware design: a developmental approach could well provide a solution, via the concept of self-organization, to the design of electronic circuits of a complexity beyond current layout techniques, and introduce into electronics at least some of the astounding fault tolerance of multi-cellular organisms, while an approximation of the growth process of biological entities could be an invaluable tool to introduce adaptability and versatility to the world of electronics.

Setting aside, for the moment, the concept of fault tolerance, in this section we will describe two different approaches to the twin mechanisms of *cellular division* and *cellular differentiation*, mechanisms at the basis of biological development.

This section will introduce two solutions to the implementation of the second mechanism: cellular differentiation, that is, the mechanism that allows a cell to determine its function within the organism (in other words, the architecture of the mapping layer of our cell). As far as cellular division is concerned, we are currently working on the development of algorithms allowing our circuit to implement this mechanism (see the "Implementation Issues" subsection for more details).

4.1 Coordinate-Based Development

Probably the simplest approach to cellular differentiation in an architecture such as that of the POEtic circuits is to assign to each cell in the system a unique coordinate (in fact, for two-dimensional structures, a set of [X,Y] coordinates). The cell can then determine its function by selecting an operator depending on its (unique) coordinates.

This is the approach used in the Embryonics project, and has been described in much detail elsewhere [10, 17].

Applied to the timer example (Fig. 8), this approach would require three different kinds of functional cell (count by 6, count by 10, and pass-through), and the timer can be realized by four cells (out of the 15 that could fit in the circuit in this example). The computation of the coordinates is quite simple, implemented by incrementing the coordinate in each cell. It is then trivial to define a differentiation table to assign an operator to each set of coordinates.

It might be worthwhile to mention that in the Embryonics approach the complexity of the system resides mostly in the genome, implemented as an executable program. As a result of this design choice, the interpreter for the genome

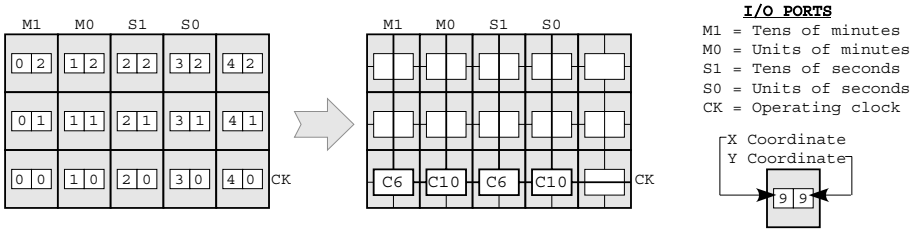

Fig. 8. A possible layout for the timer in a coordinate-based development system.

is the most complex part of the cell, while the phenotype layer is limited to a single register plus some rudimentary connection network. The POEtic approach, insofar as a coordinate-based development is concerned, will investigate a wide range of tradeoffs between the complexity of the genome and the complexity of the phenotype, as described above for the timer example.

4.2 Gradient-Based Development

The coordinate-based approach described above has a number of advantages, and notably its simplicity on the one hand, and the fact that it has been extensively used and tested in the Embryonics project over many years on the other. In a way, as long as the systems to be implemented are completely pre-determined (i.e., as long as the entire development of the organism is fully specified in the genome, as is the case, for example, of the *caenorhabditis elegans* [15]), it is also the most efficient and complete, and in fact any other approach can be reduced to a coordinate-based approach during development.

However, the growth and development of complex multi-cellular organisms in nature is influenced by the environment from the earliest stages. While a problem from the point of view of an electronic implementation, this feature can lead to extremely interesting systems capable of adapting to environments unknown at design time.

In order to illustrate how such adaptation could be applied to digital circuits, we shall exploit, for clarity's sake, the same example used above for the coordinate-based approach, that is, the timer. It should be noted, however, that such a simplistic example is far from the ideal candidate for an approach designed to address the issue of adaptation: a much more interesting field of application are neural networks, systems that are inherently capable of exploiting rich interactions with the environment.

To come back to the simple example of the 4-digit timer, let us then assume that the circuit has to operate in an *a priori* unknown environment. The environment of a timer is, of course, extremely limited, and could be seen, in the most complex case, as consisting of one input port for the operating frequency of the timer, and of four output ports for the four digits. An unknown environment could then be an environment where the exact position of these input and output ports is not known at design time (once again, this is an extremely

unlikely scenario in the design of a timer, but makes much more sense from the point of view of, for example, the control of a robot). A coordinate-based system is not capable of handling such a changing environment, since coordinates are defined as a function of the position of the cells *within* the circuit, independently of the external conditions. A more versatile approach is then required, capable of assigning a function to a cell depending not only on the internal configuration of the circuit, but also of the environment in which the circuit operates. Again drawing inspiration from biology, our choice has fallen on an approach based on the protein gradients that direct the development of organisms in nature.

The concept of protein gradients has been examined in detail in many publications, for biological [4, 14] as well as electronic [5, 6, 8] organisms. Essentially, a gradient-based system consists of a set of *diffusers* that release a given protein into the system. The concentration of the protein in the system is highest at the diffuser, and decreases with the distance. Cells are assigned a functionality depending on the proteins' concentration, and hence on the distance from the proteins' diffusers, implementing a cellular differentiation based on their position *relative* to the diffusers rather than on their coordinates within the system.

For the timer example, we can consider that each of the five I/O ports mentioned above is a diffuser for a different kind of protein (again, we chose this solution for the sake of clarity, as in reality there is a tradeoff between the number of different proteins and the complexity of the mapping circuit). It is then relatively simple to design a cell that selects its function depending on the concentration of the appropriate protein. For example (Fig. 9), a simple algorithm could place the four cells of the timer so that the rightmost cell is contiguous to the input port for the operating frequency and uses the other cells in the circuit to create paths from the four cells to the four output ports, following the corresponding gradients.

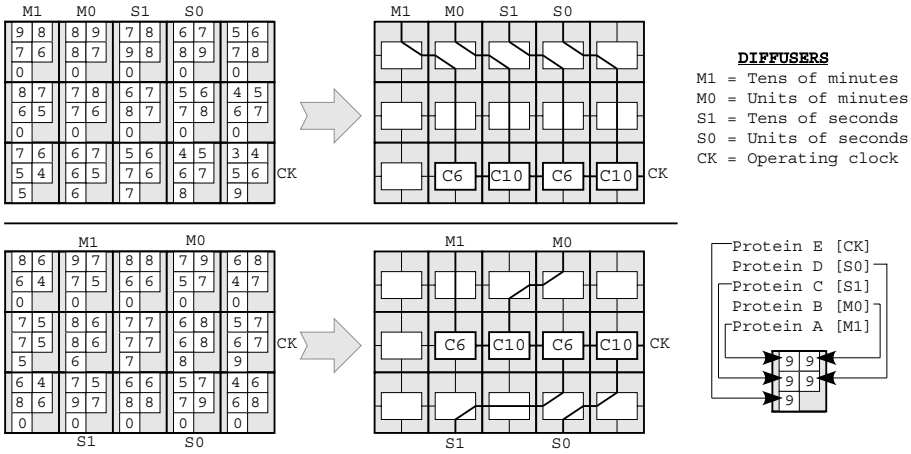

Fig. 9. Two possible layouts for the timer in a gradient-based development system.

Even in such a simplistic example, the versatility of a gradient-based system should be obvious. This kind of approach can also exploit the other axes of bio-inspiration to increase the adaptability of multi-cellular systems (aside from the obvious interest for adaptive systems such as neural networks, evolution can be used, for example, to define the position within the system of some emitters, allowing the genome to partially direct the development of the organism).

4.3 Implementation Issues

We are actively pursuing research using both of the developmental approaches described in this section. As we mentioned, for systems that can be completely defined in the genome at design time, a coordinate-based system would be more efficient. On the other hand, for adaptive systems where the development is influenced by the environment, a gradient-based approach provides a much more versatile tool.

In parallel, we are also investigating more complex, richer mappings between the genotype and the phenotype. The Embryonics approach, for example, requires a rigid one-to-one mapping from coordinates to function. A much more interesting approach exploits the possibility of using partially-defined differentiation tables that are accessed not through a one-to-one mapping, but rather through more complex algorithms (e.g., by computing the Hamming distance between the coordinates or the protein concentrations and the table entries).

The use of partially-defined differentiation tables, coupled with the possibility of environment-directed development (e.g., through protein diffusers), also opens the possibility of mechanisms that approximate the growth process in biological organisms. If the organism's structure is only partially defined in the genome, and if the environment is used to define the cells' function, then an informational (rather than physical, as in nature) growth process can be realized.

5 Fault Tolerance in the POEtic Tissue

Even if the complexity of digital hardware remains orders of magnitude behind that of the more developed biological organisms, some of the issues that nature had to confront during the millions of years required for the evolution of such organisms are gaining interest for electronics. For example, as we mentioned earlier, silicon might well be replaced, in a not-so-distant future,by molecular-size components with densities beyond what could be handled through current layout techniques. Among the many challenges represented by such novel hardware, one in particular stands out: these incredibly small components will never be "perfect". In other words, circuits will inevitably contain faulty elements.

Nature, faced with this problem, has developed extremely efficient solutions. Fault tolerance is definitely one domain where nature is vastly more advanced that electronics, and one of our goals is to draw inspiration from biological healing and repair mechanisms to achieve stronger fault tolerance in digital hardware.

From a "practical" standpoint, it is important to note that fault tolerance implies in fact two different processes: self-test, to detect that a fault has occurred within the circuit, and self-repair, to somehow allow the circuit to keep operating in the presence of one or more faults. The two processes are in fact very distinct, both conceptually and in practice. In particular, self-test implies that the circuit be capable of detecting incorrect operation and, since the fault will eventually have to be neutralized, the exact site where the incorrect operation occurs. Self-repair, on the other hand, implies that all parts of the circuit must be replaceable, physically or at least logically, via some sort of reconfiguration.

There exist, for both processes, many "conventional" approaches [1, 9, 12], but no really efficient off-the-shelf solution. Drawing once again inspiration from nature, where "fault tolerance" is achieved through not one, but a set of mechanisms ranging from molecular repair to complex organism-level immune systems, we are developing a *hierarchical* approach to fault tolerance, spanning all the levels (Fig. 1) of our POEtic systems.

5.1 Molecular-Level Fault Tolerance

Molecular-level fault tolerance is probably the most sensitive area of our hierarchical approach, as the molecules (and hence their test and repair mechanisms) represent the hardware of our system, and thus will necessarily have to be fixed at design time (rather than being modifiable by configware as the other layers).

However, the hardware layer of the POEtic tissue has not yet been fully designed (it represents the main research effort under way at the moment within the project) and, by their nature, logic level BIST (Built-In Self-Test) mechanisms require a completed layout to be efficiently implemented.

In general, the fault tolerance mechanisms implemented for the POEtic tissue will bear a loose resemblance to those used for Embryonics designs [10, 17], at least in that they will rely on a hybrid approach mixing mechanisms such as duplication (a mechanism found in nature in the DNA's double helix and in its error-correcting processes) and memory testing [9].

5.2 Cellular-Level Fault Tolerance

The coarser grain of cells with respect to molecules can be exploited to introduce more complex test and repair patterns. Fault tolerance at the processor level (cells can be seen as small processing units) is usually implemented using techniques very different from the logic level, and more complex reconfiguration can be realized by exploiting the development mechanisms of our circuits.

For example, testing at the processor level can take advantage of the presence of multi-bit busses for techniques such as parity-checking. As far as self-repair is concerned, reconfiguration mechanisms can exploit the features of the POEtic architecture to simplify the rearrangement of tasks within the array: the presence of *the full genome in each cell*, coupled with the developmental mechanism that assigns the cells' functions depending on local conditions, can for example simplify the implementation of on-line self-repair via reconfiguration (Fig. 10).

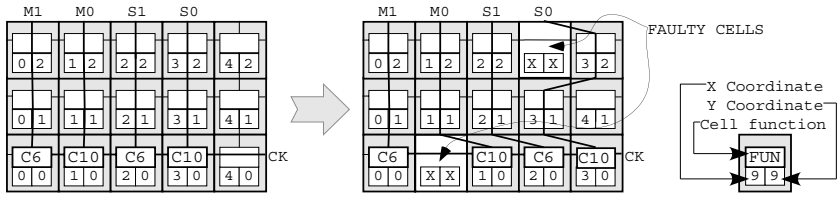

Fig. 10. Self-repair through reconfiguration in a coordinate-based system.

Of course, it should also be noted that many of the systems that POEtic architectures are meant for, and notably neural networks, are *inherently* fault tolerant at the cellular level: the death of a neuron within the system will force the learning algorithms to automatically avoid the faulty neuron and to find a solution that can perform the desired computational task in a reduced network.

5.3 Organismic-Level Fault Tolerance

A simplistic vision of organismic-level fault tolerance consists of exploiting the developmental approach of POEtic systems to create multiple redundant copies of an organism during configuration. This approach, used in the Embryonics systems [10], introduces improved fault tolerance through a simple comparison of the outputs of multiple identical circuits (the biological analogy would reside in the survival of a population even if individuals die).

This form of fault-tolerance, while relatively simple to realize (through cycles in the coordinates, as in Fig. 11, or by using multiple diffusers of the same kind in a gradient-based system), requires material *outside* of the POEtic circuits to identify (and possibly kill) faulty organisms.

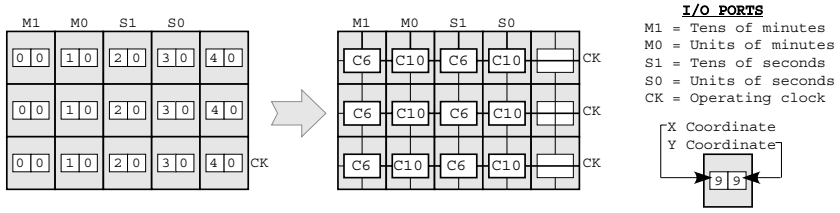

Fig. 11. Multiple redundant copies of an organism through coordinate cycling.

A more interesting approach (and one of the research axes of the POEtic project) tries to exploit mechanisms inspired by biological immune systems [2, 3]. In higher organisms, these systems rely on a multi-layered, distributed approach that is robust and capable of identifying numerous pathogens and other causes of illness, and is undoubtedly an interesting source of inspiration for the development of reliability mechanisms in silicon.

The approach relies on a negative selection algorithm [7] coupled with a more "conventional" configuration mechanism that could be based on roving self-test areas (STAR) [1]. In practice, the approach merges a fixed testing mechanism (the *innate* part of the immune system) with a *learning* negative selection mechanism (the *acquired* part), working at a system or subsystem level to monitor the state of the circuit, so as to be able to identify and kill faulty cells.

5.4 Implementation Issues

The final goal of the mechanisms we introduced is to illustrate how a hierarchical approach to fault tolerance, along the same general lines as the one present in complex biological organisms, is a very efficient solution from the point of view of assuring the correct operation of the circuit in the presence of faults. By adopting a hierarchy of mechanisms, we can exploit the best features of each level of our systems, and thus limit the overhead required to obtain an "acceptable" level of reliability for the system as a whole.

In reality, the only relevant overhead generated by introducing fault tolerance in our POEtic circuits resides in the molecular level: since all other levels are implemented in configware, reliability (and its associated overhead) becomes an option, and can be removed should the designer decide that fault tolerance is not a priority.

The important implementation issues for fault tolerance thus concern essentially the test and repair mechanisms at the molecular level. As we mentioned, however, the design of the molecule is currently under way, and all the important parameters required to introduce reliability to the molecular FPGA (connection network, degree of homogeneity, configuration mechanism, etc.) have yet to be fixed. A complete description of the implementation of fault tolerance in the POEtic tissue will thus be the subject of a future report.

6 Conclusions

The concepts presented in this report represent an attempt at finding a useful set of mechanisms to allow the implementation in digital hardware of a bio-inspired developmental process with a reasonable overhead. The modelization of multi-cellular organisms in silicon is an approach that could soon become extremely useful for the realization of highly complex systems, where concepts such as self-organization and fault tolerance are becoming key issues.

Much of our effort is currently focused on finding an adequate approach to implement development in the POEtic tissue. We are examining processes that are at the same time efficient from the point of view of the required hardware resources and scalable across one or even multiple chips. Notably, we hope to develop a mechanism versatile enough to allow on-line addition of new chips (i.e., it will be possible to add hardware to the system without disrupting its operation), achieving something closer to the *physical* growth of biological organisms.

The two developmental models described in this report fit remarkably well our requirements (scalability, robustness, efficiency, etc.), and complement each other remarkably well: the coordinate-based approach is most efficient for "conventional" systems where the layout of the circuit is known in advance, while the gradient-based approach provides an increased versatility invaluable for the realization of adaptive systems such as neural networks. Moreover, the architecture of our POEtic machines places the developmental mechanism where it can be modified by configware, allowing in the future the implementation of different approaches.

It is definitely too soon to draw any conclusions on the topic of fault tolerance in the POEtic tissue: until the hardware layer is fixed, the application of self-test and self-repair techniques is impossible. One aspect that is already apparent, however, is that our research is focusing on a *hierarchical* set of mechanisms meant to work together to achieve a level of reliability that would not be possible for a single mechanism.

Finally, we wish to reiterate the *reconfigurable* aspects of the architecture we described: the molecular tissue is meant to be a *universal* tissue for the implementation of bio-inspired systems. The architectures and mechanisms described in this report represent only some of the ideas that will be pursued in the POEtic project. The circuit, once developed, will be made publicly available and should prove a useful tool for research in the domain beyond the participants to the project.

7 Glossary

In this section, we shall briefly correlate the terms used in the project proposal with the terms and mechanisms presented in this report.

Cell. An entity composed of molecules. A cell is an element of an organism, for instance a neuron in a network. It can communicate with other cells to realize the function of the entire organism. In our approach, the architecture of a cell is defined via standard layout tools as a configuration for our molecular FPGA. As such, the architecture of the cell depends on the application. In general, for POEtic applications, the cell has the three-layer structure defined in Section 2, and each cell contains the genetic material (genome) of the entire organism within the genotype layer.

Cellular Differentiation. The behavior of an organism is the behavior of the cells that compose it. During the development of an organism (see Development, Growth, Morphogenesis, Ontogenesis), the cells of an organisms are first "created" (see Cellular Division), and then specialize so as to realize the function required by the application. This specialization is our equivalent to cellular differentiation of biological organisms, and is implemented by the development mechanism.

Cellular Division. The mechanism that generates multiple cells starting from the description (the molecular FPGA's configuration in the case of our POEtic circuits) of a single cell. The end result of the cellular division process is a two-dimensional array of identical universal (in the sense of being totipotent, that is, of being able to implement any of the functions required by the organism) undifferentiated cells. Once cellular division is finished, cellular differentiation begins.

Cicatrization. The self-repair process of the circuit, if implemented using spare cells and spare molecules, deactivates the areas (molecules or cells) that contain the fault. Thus, it leaves some "blank" areas that have no functionality, a bit like scar tissue after a wound is healed.

Development. The growth process of an arganism within a POEtic circuit. An initially empty array of cells is placed on the circuit. Each cell contains the entire genetic information of the organism, but is inert. Then, a differentiation mechanism (coordinate-based or gradient-based, depending on the approach used) defines the function of each cell within the organism. The function is selected on the basis of a differentiation table, accessed on the basis of the physical (absolute or relative, depending on the mechanism) position of the cell within the organism.

Growth. The process whereby the multi-cellular structure is programmed on the chip. The growth process can ideally continue throughout the "life" of an organism (i.e., if the physical structure of the circuit changes, for example by adding another chip to the system, new cells are created on the new material). It is implemented, essentially, through the developmental mechanism and the cellular division process.

Healing. The biological process that allows an organism to survive minor wounds and illnesses. It operates on severla levels of complexity, from molecular to organism-wide. It is implemented in hardware by fault detection and self-repair.

Molecule. The hardware processing unit of POEtic circuits. It is implemented by a simple FPGA element, connected to its nearest neighbors via a local connection network and to faraway elements via a long-distance connection network. In the POEtic circuit, all molecules are structurally identical, but can be configured to realize different functions. The layout of the POEtic molecule is currently under way, and will include dedicated hardware for the implementation of bio-inspired mechanisms such as cellular differentiation, inter-cellular communication, and fault tolerance, while remaining universal (in the sense of being able to implement any given digital circuit).

Morphogenesis. The process that defines the shape of an organism and of its cells (in the simplest case, if the organism a two-dimensional array of identical cells, it simply defines the number of cells in each dimension) on the basis of the information contained in its genome. In the approaches described in this report, the structure of our organisms depends on the developmental approach and, in practice, corresponds to the information stored in the differentiation table.

Ontogenesis. The development of a single organism. It relies partially on the information stored in the genome and on the physical position (coordinates) of the cells in the organism, and partially on the interaction between the organism and the environment during growth. On the basis of this information and of this interaction, cells divide and differentiate.

Organism. An entity composed of cells. An organism is the whole system (i.e. a neuron network), executing a given application.

Reconfiguration. The capability of a system to change its functionality and topology. On the same physical reconfigurable substrate (the transistors and metal lines on a chip), many functionally-different designs can be implemented. Also, by extension, the capability of the circuit to use spare parts to replace faulty areas, altering the structure of the configured architectures, but preserving their functionality.

Self-Repair. Self-repair is the electronic implementation of healing in our organisms. It rreserves the functionality of a circuit in the presence of one or more faults. It is realized by exploiting *spare cells* and by reconfiguring the circuit around faulty areas. It can operate at different hierarchical levels (molecule, cell, organism), and requires the presence of a fault detection mechanism.

Self-Replication. The ability of an artificial organism to create an identical copy of itself, giving rise to multiple redundant machines that can either be compared to provide error detection or differentiate into a population of organisms via interaction with the environment.

References

1. Abramovici, M., Stroud, C.: Complete Testing and Diagnosis of FPGA Logic Blocks. IEEE Trans. on VLSI Systems **9:1** (2001).
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, 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. Coen, E.: The Art of Genes. Oxford University Press (1999), New York.
5. Edwards, R.T.: Circuit Morphologies and Ontogenies. Proc. 2002 NASA/DoD Conf. on Evolvable Hardware, IEEE Computer Society Press (2002), 251–260.
6. Eggenberger, P.: Cell Interactions as a Control Tool of Developmental Processes for Evolutionary Robotics. From Animals to Animats 4: Proc. 4th Int. Conf. on Simulation of Adaptive Behavior, MIT Press - Bradford Books (1996), 440–448.
7. Forrest, A., Perelson, A.S., Allen, L., Cherukuri, R.: Self-Nonself Discrimination in a Computer. Proc. 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1994).
8. 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.
9. Lala, P.K.: Digital Circuit Testing and Testability. Academic Press (1997).
10. Mange, D., Sipper, M., Stauffer, A., Tempesti, G.: Towards Robust Integrated Circuits: The Embryonics Approach. Proceedings of the IEEE **88:4** (2000), 516–541.
11. Moreno Arostegui, J. M., Sanchez, E., Cabestany, J.: An In-System Routing Strategy for Evolvable Hardware Programmable Platforms. Proc. 3rd NASA/DoD Workshop on Evolvable Hardware, IEEE Computer Society Press (2001), 157–166.
12. Negrini, R., Sami, M.G., Stefanelli, R.: Fault Tolerance through Reconfiguration in VLSI and WSI Arrays. The MIT Press, Cambridge, MA (1989).
13. Ortega, C., Tyrrell, A.: MUXTREE revisited: Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays. LNCS **1478**, Springer-Verlag, Berlin (1998), 206–217.
14. Raven, P., Johnson, G.: Biology. McGraw-Hill (2001), 6th edition.
15. Riddle, D.L., Blumenthal, T., Meyer, B.J., Priess, J.R., eds.: C. Elegans II. Cold Spring Harbor Laboratory Press (1997).
16. 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.
17. Tempesti, G.: A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes. Thesis No. **1827** (1998), Swiss Federal Institute of Technology, Lausanne.
18. 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.