The BioWall: an Electronic Tissue for Prototyping Bio-Inspired Systems

Gianluca Tempesti, Daniel Mange, André Stauffer, Christof Teuscher
Logic Systems Laboratory, Swiss Federal Institute of Technology in Lausanne, Switzerland
E-mail: Firstname.Lastname@epfl.ch

Abstract

In this article, we present the BioWall, a giant reconfigurable computing tissue developed to implement machines according to the principles of our Embryonics (embryonic electronics) project. The BioWall's size and features are designed for public exhibition, but at the same time it represents an invaluable research tool, particularly since its complete programmability and cellular structure are extremely well adapted to the implementation of many different kinds of bio-inspired systems.

To illustrate these capabilities, we present a set of applications that range over many diverse sources of biological inspiration, from Embryonics' ontogenetic systems, through epigenetic artificial neural networks, to phylogenetic evolving hardware. All these applications have been fully implemented and tested in hardware on the BioWall.

1. Introduction

In our laboratory, we have been working on bio-inspired hardware for several years. In our research, we have covered most of the possible avenues for such inspiration [15], from *phylogenetic* systems, inspired by the evolution of biological species, through *ontogenetic* systems, inspired by the development and growth of multicellular organisms, to *epigenetic* systems, inspired by the adaptation of individuals to the environment.

Among all these research axes, the main effort in our lab has been concentrated on the ontogenetic axis, through the Embryonics (embryonic electronics) project [9][10], which aims at drawing inspiration from the development of multicellular individuals to obtain in digital hardware some of the features of biological organisms, and notably growth and fault tolerance.

Our activities have attracted a flattering amount of interest in the most varied and sometimes unexpected milieus. Among the most unexpected was undoubtedly Mrs. Jacqueline Reuge, who decided to fund the construction of a machine to display the principles of Embryonics to the public within a museum (the Villa Reuge [23]) built to honor the memory of her late husband. Her generous support has allowed us to maintain our tradition of always verifying in hardware the concepts developed for our project.

This serendipitous event allowed us to construct a machine that would otherwise have remained a dream. We named this machine *BioWall* [22], because of its biological inspiration on one side, and because of its size on the other. In fact, the main goal of the machine being as a platform to demonstrate the features of our Embryonics systems to the public through a visual and tactile interaction, the final implementation of the BioWall (Figure 1) weighs in at an impressive $5.3m \times 0.6m \times 0.5m = 3.68m^3$ (130 cubic feet).

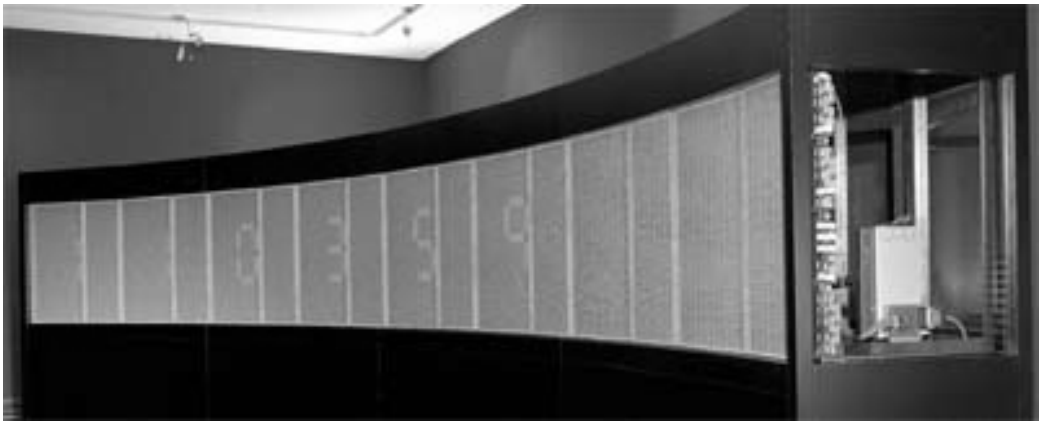

Figure 1: Frontal view of the BioWall.

Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware (EH'02)
0-7695-1718-8/02 $17.00 © 2002 IEEE

On this machine, which will be described in some technical detail in section 2, we implemented, for the first time in actual hardware, an organism endowed with all of the features of an Embryonics machine, as it has often been defined in the literature. The functionality of this organism (see subsection 3.1), the *BioWatch*, is to count hours, minutes, and seconds, and is used to demonstrate the growth and self-repair capabilities of our systems.

In a sense, the implementation of the BioWatch would by itself be sufficient to justify the effort that has gone into the construction of our BioWall (the realization of Embryonics systems was, after all, the goal of the machine). However, in developing our machine, we quickly realized that the capabilities of such a platform were not limited to a single application. In fact, as the description of the machine in section 2 should reveal, it is an ideal platform to prototype many different kinds of two-dimensional cellular systems, i.e. systems composed of an array of small, locally-connected elements.

The applications that correspond to this description are numerous, and particularly in the domain of bio-inspired systems. For example, cellular automata (CA) are a very common environment in bio-inspired research [4], from the classic Game of Life of John Conway [2] (subsection 3.2), through self-replicating loops as first developed by Chris Langton (subsection 3.3), to Von Neumann's universal constructor [20] (subsection 3.4), to name but a few (in growing order of complexity). And while the BioWall is ideally suited to the implementation of CAs, it is by no means limited to it. As examples of other possible bio-inspired systems, we will describe an implementation of a particular type of artificial neural networks, developed by Alan Turing [19] (subsection 3.5), and a two-dimensional realization of *Firefly* [13], a machine we designed and built to demonstrate the feasibility of online hardware evolution (subsection 3.6).

But we have only begun to explore the possibilities of the BioWall as a research tool. In the conclusions of this article (section 4) we will define some of the future areas of research in which the machine will be used as a prototyping platform for bio-inspired systems.

2. The BioWall

The main idea behind the construction of the BioWall is the realization of Embryonic machines. The structure of such machines, described in detail elsewhere [9][10], is hierarchical: *organisms* (application-specific systems) are realized by the parallel operation of a number of *cells* (small processors), and each cell is implemented as an array of *molecules* (programmable logic elements). To implement this kind of machines, the BioWall is structured as a two-dimensional *tissue* composed of *units* (each unit corresponds to a molecule), where each unit (Figure 2a) consists of an input element (a touch-sensitive membrane), an output element (an array of 8x8=64 two-color LEDs), and a programmable computing element (a Spartan XCS10XL Xilinx FPGA [22]). The BioWall contains 3200 units, arranged as 20 rows of 160 units.

The tissue represents then an impressive amount of computational power (3200 FPGAs, some of which are shown in Figure2b), coupled with an I/O interface (the membranes and the LED arrays) that allows a large-scale visual and tactile interaction. The advantages of this solution are obvious: on one hand the size of the display allows an immediate interaction with applications that are normally limited to software simulation on a computer screen (some of these applications are described in the next section, others are mentioned in the conclusion), and on the other hand the computing power and programmability of the FPGAs allow the prototyping of new bio-inspired systems.

For the moment (more on the subject in the conclusion), the Xilinx FPGA can only be programmed with the same configuration, which limits the functionality of the units to the 10,000 equivalent logic

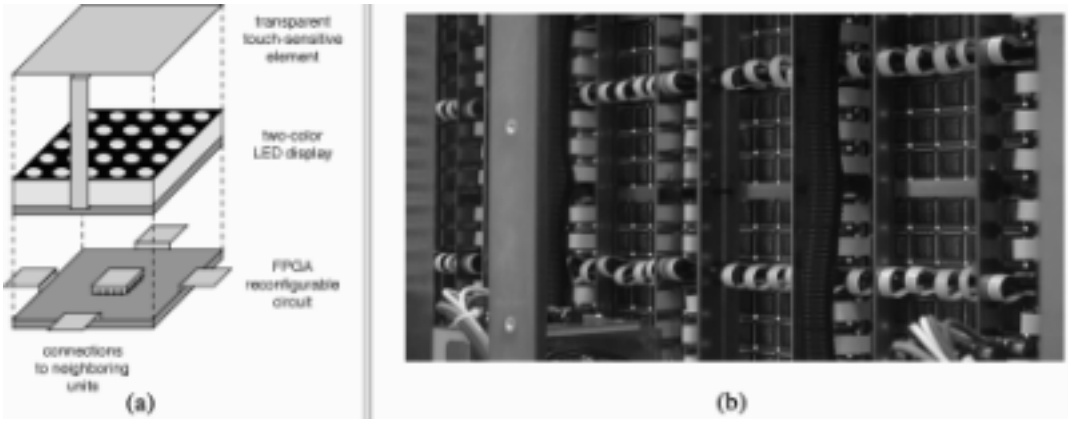

Figure 2: (a) Schematic outline of a BioWall unit. (b) Partial view of the Xilinx FPGAs.

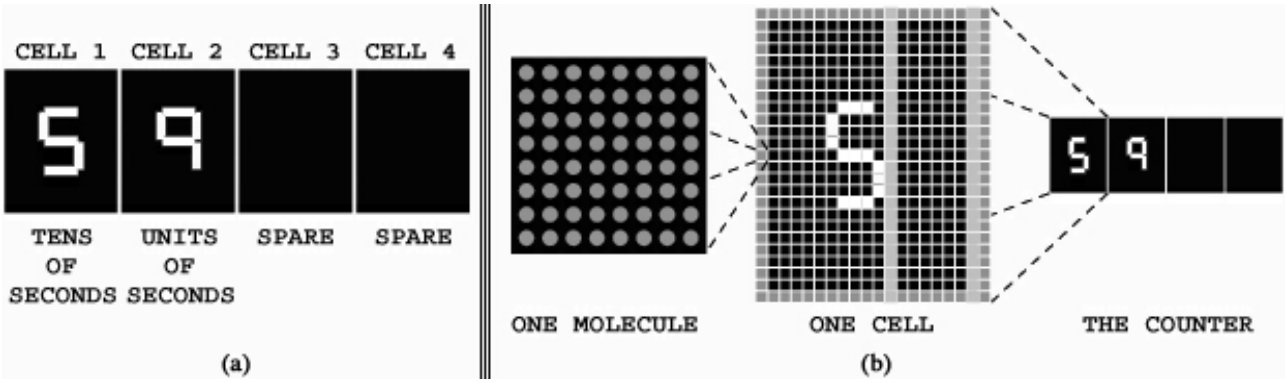

Figure 3: (a) The Counter organism. (b) Embryonics' hierarchy on the BioWall.

gates of the Spartans, while the considerable delays inherent in propagating a global signal over distances measured in meters limit the clock speed to a few hundred KHz (a speed that is more than adequate if coupled with the massive parallelism of the machine and considerably too fast for human interaction in many applications).

Besides the I/O capabilities of the membranes and of the LED displays mentioned above, a set of modules placed on the borders of the machine allow the tissue to be interfaced with standard logic, either via a PC or directly with user-defined modules (the modules, of course, allow access only to the borders of the array, but, if necessary, signal propagation logic can be programmed in the FPGAs).

The software tools developed for the BioWall are rudimentary but complete. A simple interface on a PC allows the user to define a set of files that will be used to configure the tissue. Four kinds of files are currently defined (more can be added): the configuration file for the Xilinx FPGAs, and three different formats used to send user-defined data on the input pins at the borders of the tissue (used, for example, to provide an initial configuration for a cellular automaton). The values on the output pins at the borders of the tissue can be read by the PC and either stored on disk or used as required.

3. Applications

As we mentioned in the introduction, the BioWall was designed with a specific application in mind: the realization of ontogenetic machines as defined by the specifications of the Embryonics project. However, the capabilities of the BioWall are not limited to this application. Its cellular structure is well suited to the implementation of all sorts of bio-inspired applications. In this section, we will present a few such applications, to show how the BioWall can implement hardware inspired by all the three axes of the POE model of biological inspiration [15]: phylogenesis (P), ontogenesis (O), and epigenesis (E).

3.1. BioWatch

The principles of the Embryonics project, as well as the theory behind the BioWatch, have been described in detail in a number of publications [9][10][16]. To illustrate the implementation of the BioWatch application on the BioWall, we will introduce a slightly simplified example: whereas the complete BioWatch is an organism capable of counting hours, minutes, and seconds, the Counter application (Figure3a) only counts seconds. The principles of operation of the two machines are identical.

The Counter counts seconds, from 00 to 59. From left to right, the display shows tens of seconds (from 0 to 5), units of seconds (0 to 9) and a spare zone, which remains inactive during normal operation. The counter can be described as being divided into four cells: two active (indicating tens and units respectively) and two spare. Each unit of the BioWall is a *molecule* of the Embryonics hierarchy. A cell is then a mosaic of 20x25=500 molecules (Figure3b), and contains two repair columns (2x25=50 molecules).

The visitor has control over the "life" of each molecule. A fault can be inserted in any molecule simply by pressing on the corresponding unit's membrane. The fault detection mechanism included in the Embryonics molecular layer automatically detects the error and activates the molecular self-repair mechanism. A dead molecule is instantly replaced by the neighbor immediately to its right, and so on, until the nearest gray repair column (Figure 4a). The limits of this kind of self-repair imply that only a single molecule per line, between two repair columns, can be killed. If this constraint is respected, the cell survives any amount of faults, although the figure displayed is distorted. Each cell can thus tolerate up to two faults per line (one fault between each pair of yellow columns), i.e. 2x25=50 faults in total.

If the above rule is not respected, and several faults are inserted on the same line of the same cell between two repair columns, the molecules can no longer repair themselves and the cell dies. However, the death of a cell

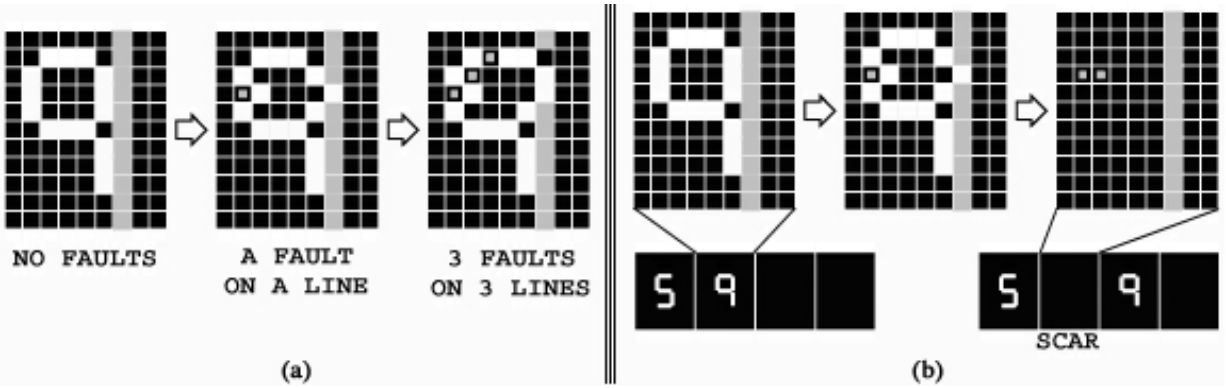

Figure 4: (a) Molecular self-repair. (b) Cellular self-repair (cicatrisation).

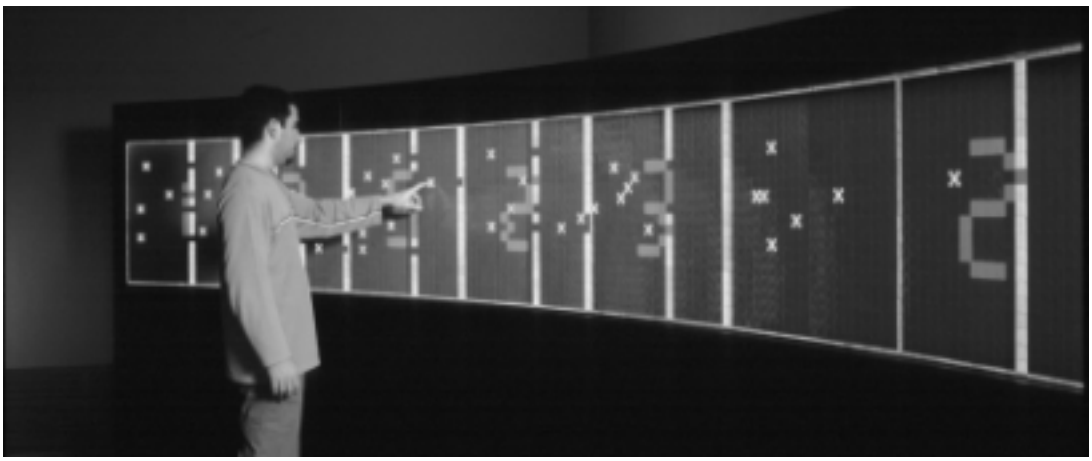

Figure 5: The complete BioWatch on the BioWall.

does not imply the death of the organism: it is instantly replaced by a spare cell to its right (Figure4b), while the dead cell is switched off and becomes a scar. It should be noted that, thorough this self-repair process, the Counter continues to work without fault: the tissue remembers its state and recovers the correct time after repair. Moreover, we are currently implement an "unkill" mechanism that, should a sufficient number of faults be removed (by pressing the membrane of a dead molecule), will automatically re-activate a dead cell, which will recover its functionality (and its state) within the organism).

The self-repair capabilities of the Embryonics machines are based on a general principle of life - *cell differentiation* [16]. Each organism is a collection of cells, each containing a full copy of the genetic program, the *genome*. This structure makes the whole organism extremely robust, since each cell contains the complete plan and can therefore replace any other defective cell. Nevertheless, like all artificial and natural organisms, the death of a sufficiently large number of cells cannot be repaired, causing the death of the organism. Of course, the advantage of the "controlled" environment in which the machine operates is that the death of the organism causes a general reset of the system, the obliteration of all injected faults, and the "birth" of a new, perfectly functioning machine.

The complete implementation of the BioWatch on the BioWall (Figure 5) uses 8 cells of 20x20 molecules each, with two spare columns of molecules in each cell. Six of the 8 cells are active during normal operation, while two are spares, ready to replace a dead cell. All the features of the Embryonics project have been tested and verified in hardware through this implementation.

3.2. Game of Life

Life is complexity. The way a spider weaves its web or an ant colony builds its nest suggests that these creatures are intelligent. They are nothing of the sort. Biologists have demonstrated that, by blindly following basic rules that have been gradually developed through natural selection, every animal behaves in ways which are sometimes extremely complex.

John Conway's game of *Life* [2] is a striking example of this kind of *emergent behavior*. The game is realized with a very simple two-dimensional CA, in which

each element represents an individual. Each individual has only two possible states: alive or dead. The next state of each individual depends on the current state of the cell itself and that of its eight nearest neighbors, according to the following rules:

- if the number of living neighbors is too small (zero or one), the individual dies of isolation and its future state is "dead";
- if an individual has exactly two living neighbors, it conserves its current state;
- if an individual has exactly three living neighbors, its future state is "alive";
- if there are too many living neighbors (four or more), it dies of overpopulation and its future state is "dead".

From these simple, local rules some astoundingly complex global behaviors have been seen and developed (Figure 6). A strikingly visual application, the game of Life is ideally suited for an implementation on the surface of the BioWall, where the touch-sensitive membranes are used to override the rules of the game and to give life to the elements. We developed two different architectures:

- In *Life1*, each of the 3200 units of the BioWall represents a single individual of the game of Life. The surface is toroidal and the touch-sensitive membranes toggle the state of the individuals.
- In *Life16*, each unit represents an array of 4×4=16 individuals (a total of 51,200 individuals on the BioWall), and the touch-sensitive membranes insert a glider (one of the stable configurations of the game of Life, which moves diagonally across the space).
- This application, while not of direct interest for research (it is mostly aimed at providing direct interaction for the general public), is nevertheless a good example of a bio-inspired application ideally suited to illustrate the features (display capabilities, interactivity, etc.) of the BioWall.

3.3. Self-Replicating Loops

The study of self-replicating machines, initiated by von Neumann over fifty years ago [20], has produced a plethora of results over the years [14]. Much of this work is motivated by the desire to understand the fundamental information-processing principles and algorithms involved in self-replication, independently of their physical realization [12]. The construction of artificial self-replicating machines can have diverse applications, ranging from nanotechnology [4], through space exploration [5], to reconfigurable computing tissues.

A major milestone in the history of artificial self-

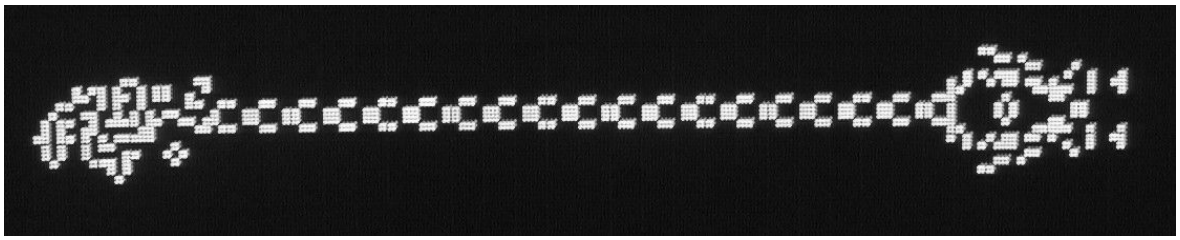

Figure 6: A global configuration on the *Life16* application, the *Dragon*.

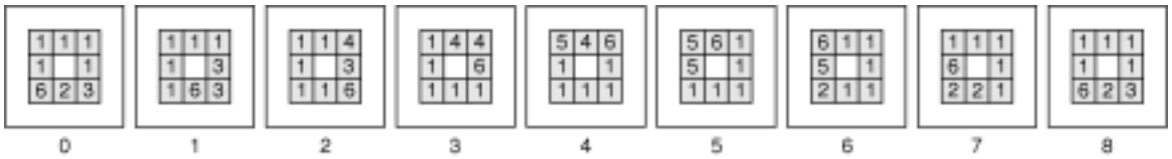

Figure 7: The idle cycle of a 3x3 self-replicating loop.

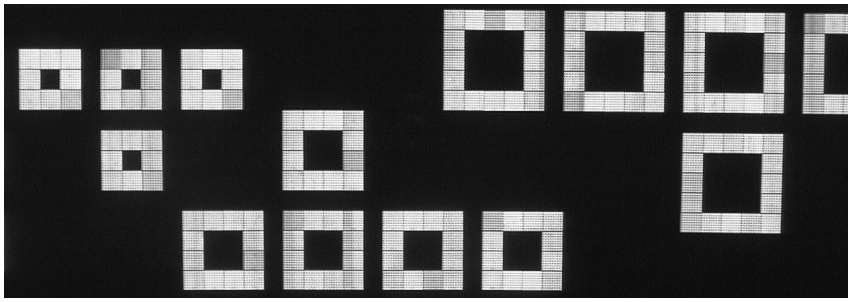

Figure 8: Self-replicating loops of different sizes.

replication is Langton's design of the first self-replicating loop [8]. His 86-cell loop is embedded in a two-dimensional, 8-state, 5-neighbor cellular space; one of the eight states is used for so-called *core* cells and another state is used to implement a sheath surrounding the replicating structure. Reggia et al. [12] proposed simplified versions of Langton's loop, the smallest being sheath-less and comprising five cells.

All self-replicating loops presented to date are essentially worlds unto themselves: once the initial loop configuration is embedded within the cellular automaton (CA) universe (at time-step 0), no further user interaction occurs, and the CA chugs along in total oblivion of the observing user. To render more interactive and more visible the self-replication process, we implemented self-replicating loops, initially of size 2x2 [17], and then of variable size (e.g., Figure 7), on the BioWall. In this implementation, every unit of the BioWall is one cell of the CA, and pressing on the membrane of a unit belonging to a loop causes the latter to replicate in one of the four cardinal directions (Figure 8).

3.4. Von Neumann's Constructor

The field of bio-inspired digital hardware was pioneered by John von Neumann. A gifted mathematician and one of the leading figures in the development of the field of computer engineering, von Neumann dedicated the final years of his life on what he called the theory of automata [20]. This research, which was unfortunately interrupted by his untimely death in 1957, was inspired by the parallel between artificial automata, of which the paramount example are computers, and natural automata such as the nervous system, evolving organisms, etc.

To find a physical realization for his theory of automata, von Neumann developed a model, known as *von Neumann's Universal Constructor*, a cellular automaton that became the basis for the greater part of the research on self-replicating machines for decades following its conception.

In von Neumann's work, self-replication is always presented as a special case of universal construction (Figure 9a): his machine `Uconst` is capable of building

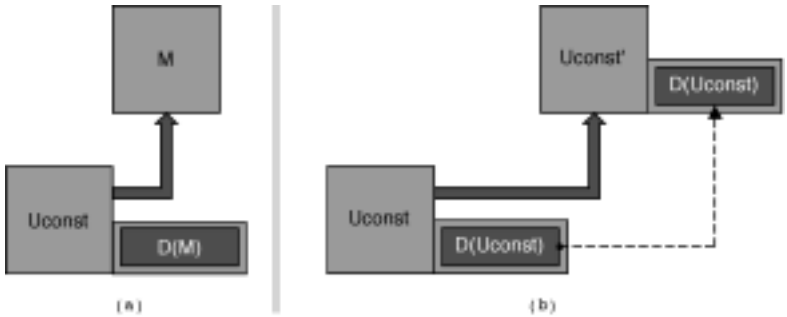

Figure 9: Von Neumann's universal constructor. (a) Basic operation. (b) Self-replication.

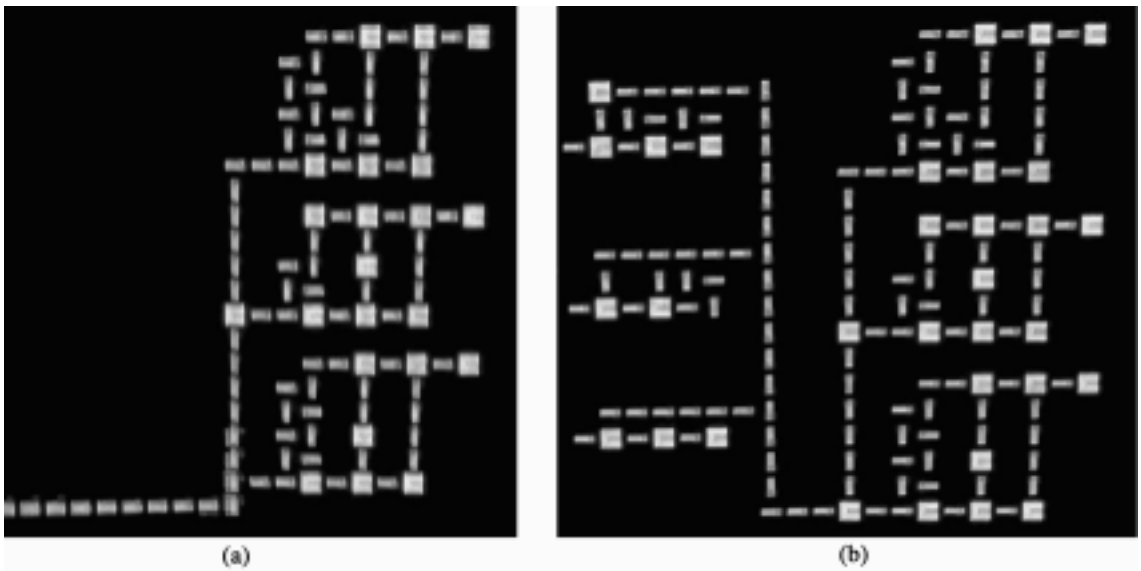

Figure 10: An organ of von Neumann's constructor. (a) Construction. (b) Final state.

any other machine M, provided it can access its description D(M). This approach was maintained in the design of his cellular automaton, which is therefore much more than a self-replicating machine. The complexity of its purpose is reflected in the complexity of its structure, based on three separate components:

- A memory tape, containing the description (a one-dimensional string of elements) of the machine to be built. In the special case of self-replication, the memory contains a description of the universal constructor itself (Figure9b).
- The constructor itself, a machine capable of reading the memory tape and interpreting its contents.
- A constructing arm, directed by the constructor, used to build the offspring (the machine described in the memory tape). The arm moves in space and sets the state of the elements of the offspring to the appropriate state.

Beyond this already considerable complexity, von Neumann postulated the presence of a universal computer Ucomp (in practice, a universal Turing machine [6], an automaton capable of performing any finite computation) that is attached to, and replicated with, the universal constructor.

The implementation as a cellular automaton is correspondingly complex. Each cell has 29 possible states, and thus, since the next state of a cell depends on its current state and that of its four cardinal neighbors, 29^5=20,511,149 transition rules are required to exhaustively define its behavior. If we consider that the size of von Neumann's constructor is of the order of 100,000 elements, we can easily understand why the automaton has not even been fully simulated as of today.

In fact, as part of the Embryonics project, we did realize a hardware implementation of a set of elements of von Neumann's automaton [1]. By carefully designing the hardware structure of each element, we were able to considerably reduce the amount of memory required to host the transition rules. Using this same technique, we were able to encode one cell of von Neumann's automaton in each element of the BioWall, providing us with a surface of 3200 elements that, while not sufficient by far to fully implement the universal constructor, represents a sufficiently large surface to implement many of the significant portions (*organs*, in von Neumann's terminology) of the machine (Figure 10).

3.5. Turing Neural Networks

It was in 1948 that Alan Turing wrote a little-known report entitled "Intelligent Machinery" [19]. At that time, he was employed at the National Physical Laboratory (NPL) in London where he worked on the design of an electronic computer - the *Automatic Computing Engine* (ACE). Turing never had great interest in publicizing his ideas, so the paper went unpublished until 1968, 14 years after his death.

Few people know that the "Intelligent Machinery" paper contains a fascinating investigation of different connectionist models that would today be called *neural networks*. It is amazing that his employer at the National Physical Laboratory, Sir Charles Darwin, grandson of the well-known English naturalist, dismissed the manuscript as a "schoolboy essay". In describing randomly connected networks of artificial neurons, Turing has written one of the first manifests of the field of *artificial intelligence* (although he did not use this term). Turing's neural networks have recently been investigated in detail in a book [18].

Turing himself called his networks *unorganized machines*. He basically proposed three types of machines: A-type, B-type, and P-type unorganized machines. A-type and B-type machines are Boolean networks made up of extremely simple, randomly interconnected NAND gates (i.e., neurons), each having exactly two inputs (i.e., synapses) from other neurons. The neurons are synchronized with a global clock signal. Unlike A-type networks, Turing's B-type networks have modifiable interconnections (basically a switch) and thus an external agent can "organize" these machines (by enabling and disabling connections) to perform a required job. The idea behind the introduction of B-type networks was to open

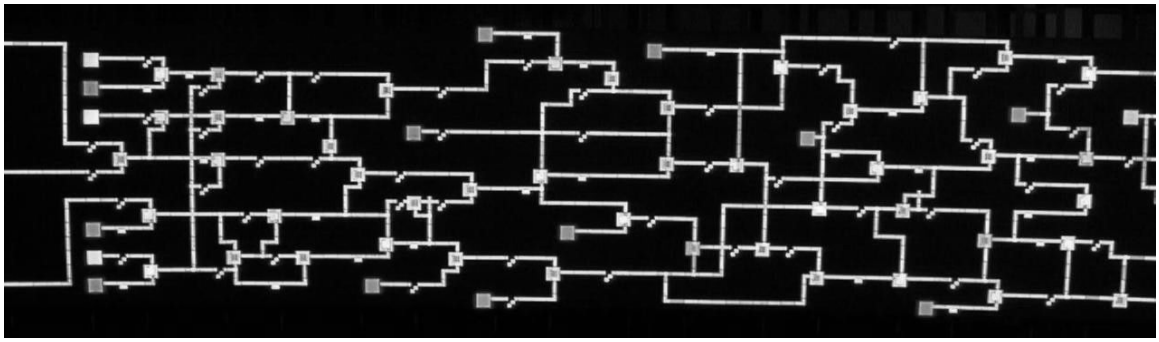

Figure 11: A B-type Turing neural network on the BioWall.

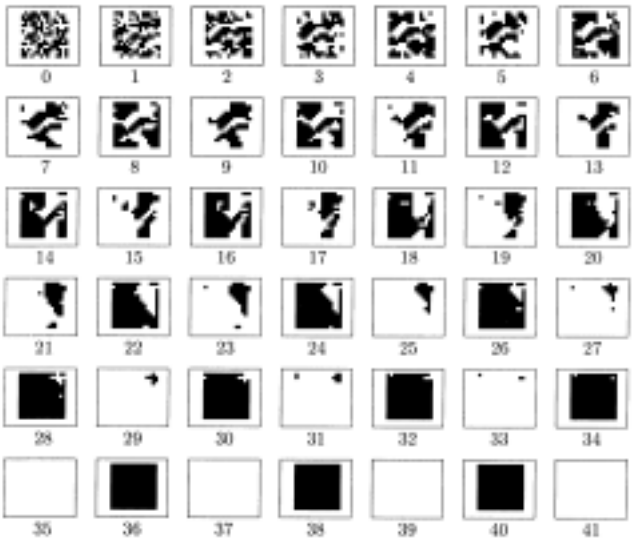

Figure 12: 2D synchronization task: a co-evolved, non-uniform, 2-state, 5-neighbor CA.

the possibility of reinforcing successful and useful links and of cutting useless ones. His deeper motivation was to build structures that can learn. On the other hand, the idea of organizing an initially random network of neurons and connections is undoubtedly one of the most significant aspects of Turing's paper.

Recently, Turing's neural networks have been implemented on the BioWall's reconfigurable tissue. Each of the 3200 units of the machine can be interactively configured by choosing one out of five possible functions: (1) empty cell, (2) neuron, (3) connection, (4) synapse, or (5) input cell. Figure 11 shows a possible configuration. The user (the external supervisor) is invited to discover and affect the behavior of the unorganized B-type machine by opening and closing synapses (i.e., "organizing" the machine) and by modifying the network's inputs. All modifications occur by simply pressing on the respective touch-sensitive membranes. This application is first and foremost a demonstration of Turing's neural networks on reconfigurable hardware (to the best of our knowledge, the first one). However, it also exemplifies the fusion of the ontogenetic and epigenetic axes in a single artificial tissue.

3.5. 2D Firefly

In 1997, the Logic Systems Laboratory presented an evolving hardware system called *Firefly* [13], based on a cellular programming approach, in which parallel cellular machines evolve to solve computational tasks. The computational task studied and successfully solved is known as synchronization: given any initial configuration, the non-uniform CA must reach, within `M` time steps, a final configuration in which all cells oscillate synchronously between all 0s and all 1s on successive time steps. The novelty of Firefly is that it operates with no reference to an external device (such as a computer that carries out genetic operators) thereby exhibiting *online autonomous evolution*.

Whereas the original Firefly machine was able to find a solution for a one-dimensional CA, we were able to evolve, on the BioWall's 3200 FPGAs, a CA that solves the synchronization task in two dimensions (Figure 12). The theoretical bases of the extension from 1D to 2D are presented in [13].

The implementation on the BioWall consists of a two-state, non-uniform CA, in which each cell (i.e., each FPGA of the BioWall) may contain a different rule. The cells' rule tables are encoded as a bit-string, known as the genome, that has a length of 2^5=32 bits for our 2D CA (the binary CA has a neighborhood of 5). Rather than employ a population of evolving CAs, our algorithm evolves a single, non-uniform CA of the size of the entire BioWall (one cell of the CA in each unit of the BioWall, that is, 3200 cells), whose rules are initialized at random. Initial configurations are then randomly generated and for each configuration the CA is run for `M` time steps. Each cell's fitness is accumulated over `C` initial configurations: a single run's score is 1 if the cell is in the correct state after `M+4` iterations, and 0 otherwise. The (local) fitness score for the synchronization task is assigned to each cell by considering the last four time steps (`M+1` to `M+4`): if the sequence of states over these steps is precisely 0-1-0-1, the cell's fitness score is 1, otherwise this score is 0. After every `C` configurations the rules are evolved through crossover and mutation. This evolutionary process is performed locally, that is, genetic operators are applied only between directly-connected cells.

Unlike standard genetic algorithms, where a population of independent problem solutions globally evolves, our approach involves a grid of rules that co-evolves locally. The CA implemented on the BioWall performs computations in a completely local manner, each cell having access only to its immediate neighbors' states. In addition, the evolutionary process is also completely local, since the application of genetic operators as well as the fitness assignment occurs locally.

Using the above-described cellular programming approach on the BioWall, we have shown that a non-uniform CA of radius 1 can be evolved to successfully solve the synchronization task. In addition, after having found a set of successful rules, our machine allows the state of each CA cell to be changed by pressing on its membrane. The user can then observe how the machine synchronizes the 3200 cells.

4. Conclusions and future research

The applications we presented are just a small sample of the capabilities of the BioWall, capabilities that we are still discovering. The cellular structure of the machine make it an ideal platform for the prototyping of bio-inspired systems, which often exploit this kind of structure, very common in nature at all levels. Its size and structure impose a certain number of limitations (e.g., clock speed), but its complete programmability provides an outstanding versatility (the different applications we mentioned should be a sufficient, if incomplete, example) and the visual and interactive component of the system are invaluable tools both for the dissemination of ideas and for the verification of research concepts that are often limited to software simulations.

Among some the other bio-inspired applications that we have implemented or plan to implement on our machine we will mention, for example, L-systems [8], ant simulations, predator-prey environments, other kinds of CAs, and more "conventional" artificial neural networks. But one application (or rather, group of applications) merits a separate mention.

As we repeatedly mentioned, biological inspiration in the design of computing machines finds its source in essentially three biological models [15]: *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.

In September 2001, our laboratory, in collaboration with the University of York, England, the Technical University of Catalunya (UPC), Spain, the University of Glasgow, Scotland, and the University of Lausanne, Switzerland, has launched, under the aegis of the Information Society Technologies (IST) program of the European Community, a three-year research project called *Reconfigurable POEtic Tissue* [24], which aims at the development of a computational substrate optimized for the implementation of digital systems inspired by all of the three above-mentioned models.

The POEtic tissue will be a cellular surface composed of a variable number of elements, or cells. Each cell will have the ability to communicate with the environment (through sensors and actuators) and with neighboring cells (through bi-directional channels), and accordingly executing a function. Each cell of the tissue will have the same basic structure, but will be able to acquire different functionalities, like *totipotent* or *stem cells* in living organisms [12]. This flexibility will be given by an organization in three layers: a genotype plane, a configuration plane, and a phenotype plane. The genotype plane of each cell will contain a full description of the organism in the form of a digital genome. The configuration plane will transform the genome into a configuration string directly controlling the processing unit of the phenotype plane. Through this cellular process, the tissue will be organized into a massively parallel multi-cellular electronic structure. Within such structure, groups of cells will be able to co-operate to realize a given task, giving rise to substructures not unlike organs in living beings.

While the POEtic project will eventually go beyond even the capabilities of the BioWall, it will be necessary, in the early stages of the project, to be able to quickly prototype the tissues developed for the project, in order to analyze not only their strengths and weaknesses, but also their feasibility in hardware (a consideration often ignored in software simulations). With its versatility, the BioWall will be an indispensable tool for the success of the POEtic project.

The BioWall, in the configuration described in this article, is currently on display at the Villa Reuge museum [23], and will remain accessible to the public for the foreseeable future. Thanks to the generosity of Mrs. Reuge, however, we were able to keep a smaller (but still sizeable, at 2000 elements) version of the machine in our lab, to serve as a research tool [22]. As a consequence, we are able not only to develop new applications, but also to keep ameliorating the features of the tissue (something that would not be possible with the museum's version, which cannot have downtimes). Notably, planned upgrades

to the system include an improved I/O interface and a more important upgrade that will allow each Xilinx FPGA to be programmed independently of the others, increasing even further the versatility of the tissue (with all the advantages and drawbacks inherent in having to handle a huge surface of programmable logic).

To conclude, we would like to invite all of you to come and "play" with the machine either at the Villa Reuge museum or in our laboratory. And, on a more "serious" note, we would be extremely interested in putting our machine at the disposal of other research groups, who could be interested in a hardware realization of their ideas and concepts.

Acknowledgements

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

We thank Moad Brahami and Martin Duvanel for von Neumann's constructor, Julien Pilet and Maciej Kupiec Gavillet for the implementation of Turing's neural networks, and Hans Jaeckle for 2D Firefly.

We also thank André Badertscher for some of the photos used in the article.

Finally, we wish to thank the European Community in general and the IST program in particular for providing the means to pursue the development of bio-inspired hardware in the next years.

References

[1] J.-L. Beuchat, J.-O. Haenni. "Von Neumann's 29-State Cellular Automaton: A Hardware Implementation". *IEEE Trans. on Education*, Vol. 43, No 3, pp. 300-308, Aug. 2000.

[2] E.R. Berlekamp, J.H. Conway, R.K. Guy. *Winning Ways for your Mathematical Plays. Vol.2: Games in Particular*. Academic Press, London, 1985.

[3] A. Burks (ed.) *Essays on Cellular Automata*. University of Illinois Press, Urbana IL, 1970.

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

[5] R.A. Freitas and W.P. Gilbreath (eds.). *Advanced automation for space missions: Proceedings of the 1980 NASA/ASEE summer study*. NASA, Scientific and Technical Information Branch (available from U.S.G.P.O., Publication 2255), Washington D.C., 1980.

[6] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley, Redwood City, CA, 1979.

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

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

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

[10] D. Mange and M. Tomassini (eds.) *Bio-Inspired Computing Machines*, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998

[11] H. Pearson. "The Regeneration Gap". *Nature*, vol. 414, 22, p. 388-390, November 2001.

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

[13] M. Sipper. *The Evolution of Cellular Automata: The Cellular Programming Approach*. Springer-Verlag, Berlin 1997.

[14] M. Sipper. "Fifty years of research on self-replication: An overview". *Artificial Life*, 4:237-257, 1998.

[15] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. "A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems". *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 83-97, April 1997.

[16] A. Stauffer, D. Mange, G. Tempesti, and C. Teuscher. "BioWatch:A giant electronic bio-inspired watch". In D. Keymeulen, A. Stoica, J. Lohn and R.S. Zebulum (eds.), *Proceedings of the Third NASA/DOD Workshop on Evolvable Hardware (EH-2001)*, pp.185-192, IEEE Computer Society, Pasadena CA, 2001.

[17] A. Stauffer and M. Sipper. "Externally controllable and destructible self-replicating loops". In J. Kelemen and P. Sosik (eds.), *Advances in Artificial Life: Proceedings of the 6th European Conference on Artificial Life (ECAL 2001)*, Lecture Notes in Artificial Intelligence, 2159:282-291, Springer-Verlag, Heidelberg, 2001.

[18] C. Teuscher. *Turing's Connectionism. An Investigation of Neural Network Architectures*. Springer-Verlag, London, 2001.

[19] A.M. Turing. "Intelligent Machinery". In B. Meltzer and D. Michie (eds.), *Machine Intelligence*, volume 5, pages 3-23. Edinburgh University Press, Edinburgh, 1969.

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

[21] Xilinx Corp. *Spartan/XL Families FPGAs Data Sheet*. Available online at http://www.xilinx.com

[22] http://lslwww.epfl.ch/biowall/

[23] http://www.villareuge.ch/

[24] http://www.poetictissue.org