Ontogenetic hardware[1]

Moshe Sipper *, Daniel Mange, André Stauffer

Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland

Abstract

Ontogeny is the process by which a single mother cell, the zygote, gives rise, through successive divisions, to a complete organism, possibly containing trillions of cells (e.g. in humans). This paper describes research whose inspiration is drawn from the process of ontogenetic development. By adopting certain features of cellular organization, and by transposing them to the world of integrated circuits on silicon, we show that certain properties unique to the living world, such as self-replication, self-repair, and growth, can also be attained in artificial objects (integrated circuits). Specifically, we identify and describe three classes of ontogenetic hardware: (1) self-replicating hardware; (2) embryonic hardware; and (3) L-systems based hardware, dubbed L-hardware. For each class we present an example of a hardware realization, along with a discussion of possible applications. Continued research on ontogenetic hardware may yield novel bio-inspired systems endowed with replicative, growth, and regenerative capabilities. © 1997 Elsevier Science Ireland Ltd.

Keywords: Ontogeny; Self-replication; Self-repair; Embryonics; L-systems

1. Introduction

A human being consists of approximately 60 trillion (60×10^{12}) cells. At every moment, in each of these cells, the genome, a string of 3 billion characters, is decoded to produce the proteins needed for the survival of the organism. This genome contains the individual's genetic inheritance, 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 throughout the individual's lifetime. Faults are rare and, in the majority of cases, are successfully detected and repaired.

This process is remarkable for its complexity and its precision. Moreover, it relies on completely discrete processes: the chemical structure of DNA is a sequence of four bases (adenine,

* Corresponding author. Tel.: +41 21 6932658; fax: +41 21 6933705; e-mail: moshe.sipper@di.epfl.ch

cytosine, guanine, and thymine). Each three-base group (known as a codon) is decoded in the cell to produce a particular amino acid, a future constituent of the final protein (except for a few codons which act as start and stop signals to control protein synthesis (Watson et al., 1987)).

The process by which such complex organisms come to be is known as ontogeny. It involves the successive divisions of a mother cell, the zygote, with each newly formed cell possessing a copy of the original genome, followed by a specialization of the daughter cells in accordance with their surroundings, i.e. their position within the ensemble. This latter phase is known as cellular differentiation. Ontogeny is thus the developmental process of a multicellular organism. Note that the process is essentially deterministic: an error in a single base within the genome can provoke an ontogenetic sequence which results in notable, possibly lethal, malformations.

This paper describes research whose inspiration is drawn from the basic processes of molecular biology, and in particular from ontogenetic development (Ransom, 1981). By adopting certain features of cellular organization, and by transposing them to the world of integrated circuits on silicon, we show that certain properties unique to the living world, such as self-replication, self-repair, and growth, can also be attained in artificial objects (integrated circuits).

Ontogenetic hardware represents one axis of the recently introduced POE model of bio-inspired systems (Sanchez et al., 1997; Sipper et al., 1997). The model is based on the observation that if one considers life on Earth since its very beginning, three levels of organization can be distinguished: the phylogenetic level concerns the temporal evolution of the genetic programs within individuals and species; the ontogenetic level concerns the developmental process of a single multicellular organism; and the epigenetic level concerns the learning processes during an individual organism's lifetime. In analogy to nature, the space of bio-inspired hardware systems can be partitioned along these three axes, phylogeny, ontogeny, and epigenesis, giving rise to the POE model (Fig. 1, Sanchez et al., 1997; Sipper et al., 1997). The distinction between the axes cannot be easily

drawn where nature is concerned, indeed the definitions themselves may be subject to discussion. Sipper et al. (1997) therefore defined each of the above axes within the framework of the POE model as follows: the phylogenetic axis involves evolution; the ontogenetic axis involves the development of a single individual from its own genetic material, essentially without environmental interactions; and the epigenetic axis involves learning through environmental interactions that take place after formation of the individual. For a detailed introduction of the POE model the reader should refer to the above references.

This paper concentrates on one particular axis, namely, the ontogenetic one, the aim being to give a detailed account of our research into hardware implementations inspired by ontogenetic processes. After a brief introduction of large-scale programmable circuits (Section 2), we identify and describe three classes of bio-inspired hardware systems, situated along the ontogenetic axis: (1) self-replicating hardware (Section 3); (2) embryonic hardware (Section 4); and (3) L-systems based hardware, dubbed L-hardware (Section 5). For each class we present an example of a hardware realization, along with a discussion of possible applications. Our paper ends in Section 6 with conclusions and Pesavdirections for future research.

Before continuing it is important to note that our systems are not molecular machines but rather based on current-day electronics technology. In his paper, Drexler (1989) noted that living things are characterized by heavy use of diffusive transport, matching assembly, topological struc-
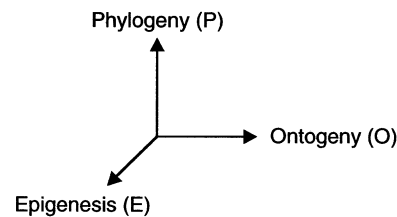

Fig. 1. The POE model. Partitioning the space of bio-inspired hardware systems along three axes: phylogeny, ontogeny, and epigenesis.

tures, and adaptive parts. He called systems that share these characteristics (whether living or not) O-style systems (where O stands for organic). Mechanical systems are characterized by heavy use of channeled transport, positional assembly, geometric structures, and inert parts. Drexler called systems that share these characteristics M-style systems (where M stands for mechanical). He further noted that the difference between O-style and M-style is not a hard distinction, but a matter of degree, and that they form, at least in principle, a continuum. Having said this, we emphasize that the ontogenetic hardware presented in this paper is essentially M-style. This must be kept in mind, especially when considering the biological inspiration, which is just that—inspiration.

2. Large-scale programmable circuits

An integrated circuit is called programmable when the user can configure its function by programming. The circuit is delivered after manufacturing in a generic state and the user can adapt it by programming a particular function. In this paper we consider solely programmable logic circuits, where the programmable function is a logic one, ranging from simple boolean functions to complex state machines. The programmed function is coded as a string of bits representing the configuration of the circuit. Note that there is a difference between programming a standard microprocessor chip and programming a programmable circuit—the former involves the specification of a sequence of actions, or instructions, while the latter involves a configuration of the machine itself, often at the gate level.

The most commonly used device in the past few years is the field-programmable gate array (FPGA) (Sanchez, 1996). An FPGA is an array of logic cells placed in an infrastructure of interconnections, which can be programmed at three distinct levels (Fig. 2): (1) the function of the logic cells; (2) the interconnections between cells; and (3) the inputs and outputs. All three levels are configured via a string of bits that is loaded from an external source, either once or several times. In
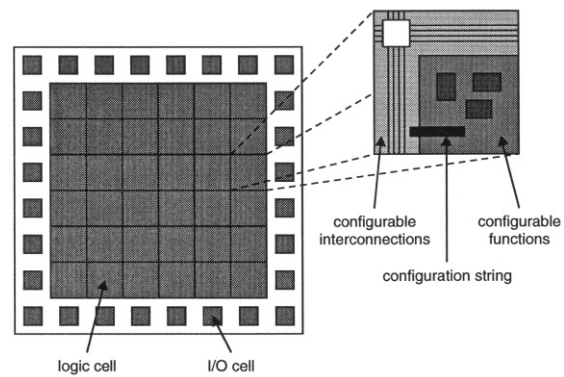

Fig. 2. A schematic diagram of a field-programmable gate array (FPGA). An FPGA is an array of logic cells placed in an infrastructure of interconnections, which can be programmed at three distinct levels: (1) the function of the logic cells; (2) the interconnections between cells; and (3) the inputs and outputs. All three levels are configured via a configuration bit string that is loaded from an external source, either once or several times.

the latter case the FPGA is considered re-configurable.

FPGAs are highly versatile devices that offer the designer a wide range of design choices. However, this potential power necessitates a suite of tools in order to design a system. Essentially, these generate the configuration bit string, given such inputs as a logic diagram or a high-level functional description. FPGAs have been widely used recently in the burgeoning field of evolvable hardware, giving rise to systems situated along the phylogenetic axis (Sanchez and Tomassini, 1996; Sanchez et al., 1997; Sipper et al., 1997).

3. Self-replicating hardware

The ontogenetic axis involves the development of a single individual from its own genetic material, essentially without environmental interactions. As can be seen in Fig. 3 (based on Dawkins, 1989) ontogeny can be considered orthogonal to phylogeny. The main process involved in the ontogenetic axis can be summed up as growth, or construction. Ontogenetic hardware exhibits such characteristics as replication and regeneration which find their use in many applications. To

date, we have identified three classes of ontogenetic hardware: self-replicating systems, described in this section; embryonic hardware; and L-hardware, described in the following two sections. Replicating systems have the ability to self-repair upon suffering heavy damage (Mange et al., 1996) and have been proposed as an economical means of space exploration (Freitas and Gilbreath, 1980). Replication can in fact be considered a special case of growth—this process involves the creation of an identical organism by duplicating the genetic material of a mother entity onto a daughter one, thereby creating an exact clone. As put forward by Sipper et al. (1997), it is important to distinguish between two distinct terms, replication and reproduction, which are often considered synonymous. Replication is an ontogenetic, developmental process, involving no genetic operators, resulting in an exact duplicate of the parent organism (as in the budding process of the hydra, Wolpert, 1991). Reproduction, on the other hand, is a phylogenetic process, involving genetic operators such as crossover and mutation, thereby giving rise to variety and ultimately to evolution (note that reproduction has been justly placed on the vertical axis of Fig. 3).
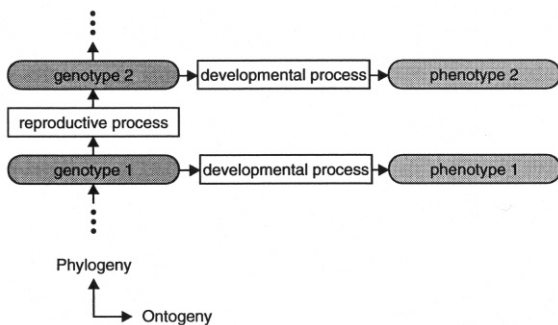

Fig. 3. The phylogenetic and ontogenetic axes can be considered orthogonal. The figure shows two generations preceded and followed by an indefinite number of generations. Ontogeny involves the development of the phenotype in a given generation (horizontal arrows), while phylogeny involves the succession of generations through reproduction of the genotype (vertical arrows). Note that genes, the basic constituents of the genome, act on two quite different levels: they participate in the developmental process, influencing the development of the phenotype in a given generation, and they participate in genetics, having themselves copied down the generations (reproduction) (Dawkins, 1989).
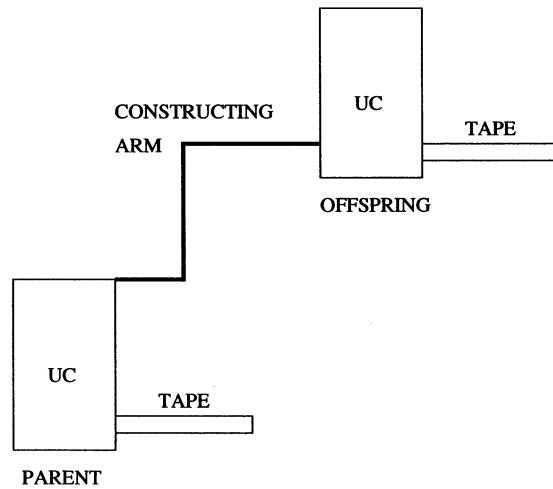
Fig. 4. A schematic diagram of von Neumann's self-replicating cellular automaton. The system is a universal constructor (UC), namely, a machine capable of constructing, through the use of a 'constructing arm', any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e. of self-replicating.

Research on replicating systems began with von Neumann's work in the late 1940s on self-replicating machines. This work was later extended by others, and more recently we have seen the emergence of systems that exhibit other ontogenetic mechanisms, such as cellular division and cellular differentiation (discussed in the next section). This line of research can be divided into four stages, described below.

3.1. von Neumann's universal constructor

von Neumann (1966) (see also Pesavento, 1995) and his successors Banks (1970), Burks (1970) and Codd (1968) developed self-replicating automata capable of universal computation (i.e. able to simulate a universal Turing machine; Hopcroft and Ullman, 1979) and of universal construction (i.e. able to construct any automaton described by an artificial genome; Fig. 4). These systems are based on the cellular automaton model, originally conceived by Ulam and von Neumann in the 1940s to study the issue of self-replication within a formal framework (von Neumann, 1966).

Cellular automata (CA) are dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells (Toffoli and Margolus, 1987).

The cellular array (grid) is n-dimensional, where $n = 1,2,3$ is used in practice. In this paper we shall describe systems with $n = 1, 2$, i.e. one- and two-dimensional grids. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table, with an entry for every possible neighborhood configuration of states. The cellular neighborhood of a cell consists of the surrounding (adjacent) cells. For one-dimensional CAs, a cell is connected to r local neighbors (cells) on either side, as well as to itself, where r is a parameter referred to as the radius (thus, each cell has $2r + 1$ neighbors). For two-dimensional CAs, two types of cellular neighborhoods are usually considered: five cells, consisting of the cell along with its four immediate nondiagonal neighbors; and nine cells, consisting of the cell along with its eight surrounding neighbors. When considering a finite-sized grid, spatially periodic boundary conditions are frequently applied, resulting in a circular grid for the one-dimensional case, and a toroidal one for the two-dimensional case. A one-dimensional CA is illustrated in Fig. 5 (based on Mitchell, 1996).

While the complexity of the above self-replicating CAs is such that no full physical implementation has yet been possible[2], the von Neumann cell has recently been realized in hardware by Beuchat and Haenni (1997), who constructed a logic module that implements a single cell. Each module is embedded in a plastic box (of dimensions $8 \times 8 \times 4$ cm), whose top face contains a number of connection points and a LED display showing the current state of the cell (Fig. 6a). Several such modules can be fitted together to produce a small cellular array. The sides of the modules contain electrical contacts, which allow adjacent cells to transmit information to each other without additional wiring.

Each von Neumann module is composed of two units, a computation unit and a display unit. The computation unit, implemented in an FPGA (Section 2), calculates the cell's next state by directly communicating with the adjacent, neighboring cells. The cell's state is stored and sent to the display unit, implemented using a dot-matrix display, a microcontroller, and a small number of latches. This latter unit constantly reads the current state of the cell, and updates the display accordingly. These two units are shown in Fig. 6b.

As noted above, the complete von Neumann system cannot be fully realized due to its size. To date, Beuchat and Haenni (1997) used the above module to implement a 25-cell 'organ'. Von Neumann's machine is divided into many functional blocks, such as decoders and pulsers, known as organs. For example, a pulser P(11001) generates at the output cell (top right of Fig. 7) the sequence of excitations (signals) 11001 a fixed number of time steps after receiving an excitation (i.e. a 1 signal) at the input cell (bottom left of Fig. 7).

Rule table:

neighborhood:	111	110	101	100	011	010	001	000
output bit:	1	1	1	0	1	0	0	0

Grid:

$t = 0$	0	1	1	0	1	0	1	1	0	1	1	0	0	1	1
$t = 1$	1	1	1	1	0	1	1	1	1	1	1	0	0	1	1

Fig. 5. Illustration of a one-dimensional, 2-state CA (based on Mitchell, 1996). The connectivity radius is $r = 1$, meaning that each cell has two neighbors, one to its immediate left and one to its immediate right. Grid size is $n = 15$. The rule table for updating the grid is shown on top. The grid configuration over one time step is shown at the bottom, where configuration refers to a legal assignment of states to cells in the grid. Spatially periodic boundary conditions are applied, meaning that the grid is viewed as a circle, with the leftmost and rightmost cells each acting as the other's neighbor.

---

[2] For example, the size of von Neumann's machine has been estimated at 100 000 cells (Pesavento, 1995; Sipper et al., 1997).
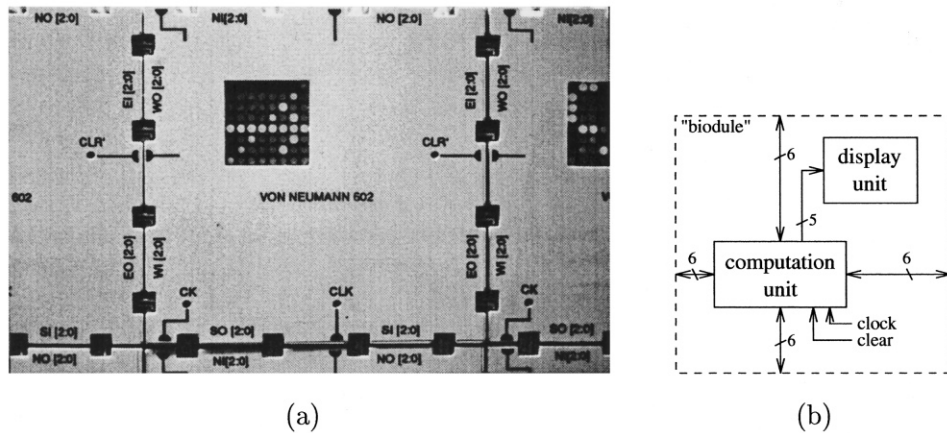
Fig. 6. Hardware implementation of the von Neumann cell. (a) Top face of the von Neumann module, including connection points to other cells and a LED display showing the current state of the cell. (b) General schema: the module is composed of two units, a computation unit, computing the cell's next state, and a display unit, handling the dot-matrix display.

It should be noted that while von Neumann's system exhibits self-replication it is not capable of detecting and recovering from faults, an essential property for a system of such size. Von Neumann himself had raised the issue of self-repair in his model though he had not actually shown how this can be attained. We shall come back to this issue in the next section, where embryonic hardware will be shown to possess not only the ability to self-replicate but also to self-repair.

3.2. Langton's self-replicating loop

Langton (1984) and his successors Byl (1989), Reggia et al. (1993) and Morita and Imai (1997) developed self-replicating automata which are much simpler than the universal constructors. These machines, however, lack any computing and constructing capabilities, their sole functionality being that of self-replication (Fig. 8).

3.3. Tempesti's self-replicating loop with finite computational capabilities

Tempesti (1995) developed a self-replicating CA, similar to that of Langton's, yet with the added capability of attaching to the automaton an executable program which is duplicated and executed in each of its copies. The program is stored within the loop, interlaced with the replication code (Fig. 9).

3.4. Self-replicating loop with universal computational capabilities (Perrier et al., 1996)

Perrier et al. (1996) demonstrated a self-replicating loop that is capable of implementing any program, written in a simple yet universal programming language. The system consists of three parts, loop, program, and data, all of which are replicated, followed by the program's execution on the given data. Thus, Perrier et al. (1996) demonstrated a viable, self-replicating machine with programmable capabilities (Fig. 10).

Though to date only the von Neumann cell has been implemented in hardware, the other systems described above could be similarly realized.

4. Embryonic hardware

One of the defining characteristics of a biological cell concerns its role as the smallest part of a living being which carries the complete plan of the being, that is its genome (Mange and Stauffer, 1994). In this respect, the self-replicating automata of the previous section are unicellular organisms: there is a single genome describing (and contained within) the entire machine.
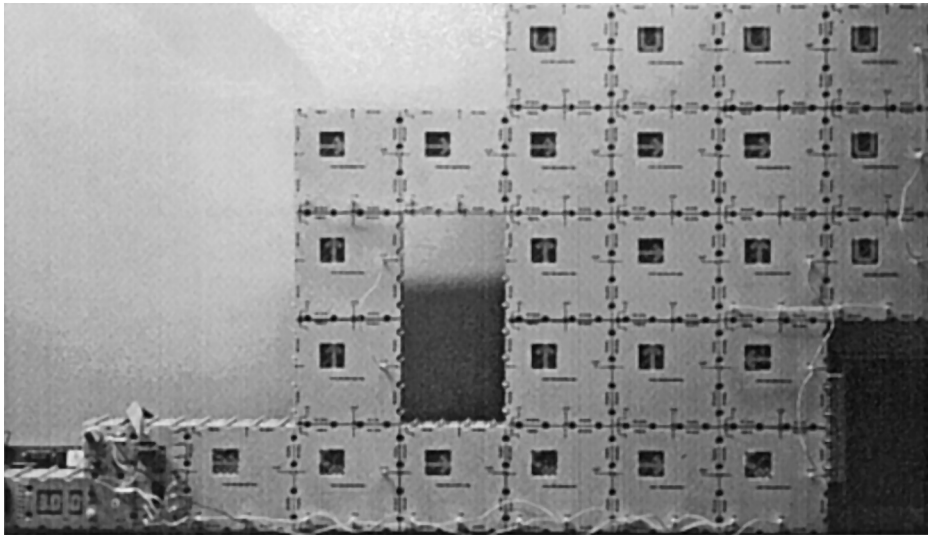
Fig. 7. Hardware implementation of one of the organs of von Neumann's universal constructor, known as a pulser, using the module of Fig. 6. The above pulser P(11001) generates at the output cell (top right) the sequence of excitations (signals) 11001 a fixed number of time steps after receiving an excitation (i.e. a 1 signal) at the input cell (bottom left). Note that the 25 von Neumann modules are not arranged as a 5×5 square—in fact, the arrangement is that of a 7×7 square, where unused cells are simply not implemented. This allows for the construction of a larger organ for the price (literally) of a smaller one.

Mange and Stauffer (1994), Mange et al. (1996, 1997a) and Marchal et al. (1994, 1996) proposed a new architecture called embryonics, or embryonic electronics. Based on three features usually associated with the ontogenetic process in living organisms, namely, multicellular organization, cellular differentiation, and cellular division, they introduced a new cellular automaton, complex enough for universal computation, yet simple enough for a physical implementation through the use of commercially available digital circuits. They developed an artificial cell, dubbed biodule (biological module), comprising of three structures found in living cells (Fig. 11): (1) a plastic box constitutes the external membrane, ensuring the cell's material encasement and realizing all the electronic functions necessary for communication with neighboring cells; (2) a processor responsible for interpreting the genome constitutes the cytoplasm, in analogy to a ribosome; and (3) a random access memory (RAM) acts as the cell's nucleus, containing a copy of the entire genetic makeup, i.e. a genome composed of a linear sequence of genes.

The biodule cell is used as an elementary unit from which multicellular organisms can ontogenetically develop to perform useful tasks. Cellular differentiation takes place by having each cell compute its coordinates (i.e. position) within a one- or two-dimensional space, after which it can extract the specific gene within the genome responsible for the cell's functionality. Cellular division occurs when a mother cell, the zygote, arbitrarily placed within the grid, multiplies to fill a large portion of the space, thus forming a multicellular organism. In addition to self-replication, this artificial organism also exhibits self-repair capabilities, another biologically inspired phenomenon. Such self-replicating machines are multicellular artificial organisms, in the sense that each of the several cells comprising the organism contains one copy of the complete genome.

To date, three applications have been demonstrated, a random number generator (Mange et al., 1996), a universal Turing machine (Stauffer et al., 1996; Mange et al., 1997a), and the biowatch (Stauffer et al., 1997), described ahead. The biowatch is an artificial 'organism' designed to

```
. . . . . . . .              . . . . . . . .    . . . . . . . .
. 1 7 0 1 4 0 1 4 .          . 7 0 1 7 0 1 7 0 .  . 1 4 0 1 4 0 1 1 .
. 0 . . . . . . 0 .          . 1 . . . . . . 1 .  . 0 . . . . . . 1 .
. 7 .       . 1 .            . 1 .       . 7 .  . 7 .       . 1 .
. 1 .       . 1 .            . 1 .       . 0 .  . 1 .       . 1 .
. 0 .       . 1 .            . 1 .       . 1 .  . 0 .       . 7 .
. 7 .       . 1 .            . 1 .       . 7 .  . 7 .       . 0 .
. 1 . . . . . . 1 . . . . .  . 0 . . . . . . 0 . . . 1 . . . . . . 1 .
. 0 7 1 0 7 1 0 7 1 1 1 1 1 .  . 4 1 0 4 1 0 7 1 0 7 5 0 6 1 0 7 1 0 7 .
. . . . . . . . . . . . . .  . . . . . . . . . . . . . . . . . . .
```
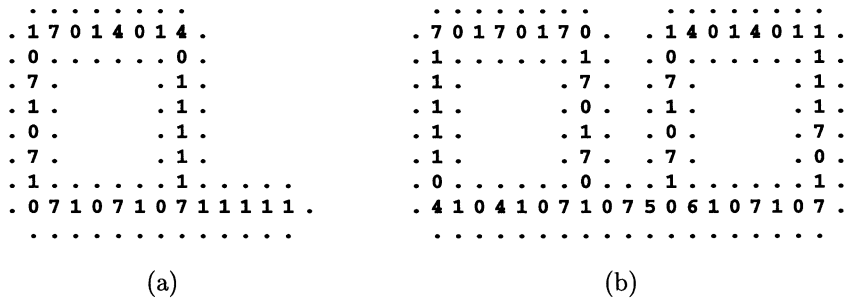
          (a)   (b)

Fig. 8. Langton's self-replicating loop. The structure, embedded within an 8-state CA space (i.e. each cell can be in one of eight possible states) consists of a looped pathway, containing instructions, with a construction arm projecting out from it. Upon encountering the arm junction, the instruction is replicated, with one copy propagating back around the loop again and the other copy propagating down the construction arm, where it is translated as an instruction when it reaches the end of the arm. Note that the loop lacks any computing and constructing capabilities, its sole functionality being that of self-replication. (a) Time step 0. (b) Time step 126.

count minutes (from 0 to 59) and seconds (from 0 to 59), comprising in effect a modulo-3600 counter that is able to self-replicate and self-repair. The basic architecture, shown in Fig. 12a, consists of four cells (four biodules), with two cells acting as the minutes counter and the other two acting as the seconds counter. The biowatch is shown in Fig. 13.

Self-replication of the four-cell biowatch can take place provided the following two conditions are satisfied: (1) there exists a sufficient number of spare cells (unused cells to the right-hand side of the array, at least four in our case); and (2) the calculation of the cell's coordinates produces a cycle, i.e. $X = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \rightarrow \ldots$, where X is the cell's horizontal position, or coordinate (Fig. 12b). As the same pattern of coordinates produces the same pattern of genes, self-replication can be attained by duplicating the basic coordinate pattern, as long as there are available empty cells (Fig. 12c).

Self-repair is achieved as follows (Fig. 12d): identifying a cell to be eliminated, by pressing upon the KILL button of the biodule (Fig. 11), inactivates that cell, which is thereupon considered 'dead' (the $X = 2$ cell in Fig. 12d). All cells to the right of the dead one are then displaced one position to the right. Obviously, this process requires as many spare cells to the right, as there are faulty cells to repair.

In the above system the detection of a malfunctioning cell is done manually, requiring that an outside observer press the KILL button, i.e. pinpoint the fault. To overcome this shortcoming, as well as to attain universal construction, Mange et al. (1997b) have recently added a molecular level to the cellular level, described above. Essentially, in this new addition to the embryonics family each cell is composed of yet finer elements, referred to as molecules. The system as a whole now possesses a three-part genome, corresponding to the three phases it undergoes: (1) first the 'polymerase' part of the genome is injected, setting the boundaries between cells, i.e. fixing the dimensions of the molecular array; (2) then the 'ribosomic' part of the genome is introduced, programming the system at the molecular level; and (3) finally the 'functional' part of the genome is injected into the random access memory of each cell, programming the system at the cellular level to carry out its intended function (e.g. a biowatch). The presence of a completely programmable molecular 'tissue' allows the implementation of a cell of any dimension (subject, of course, to the availability of sufficient hardware resources), thus enabling universal construction (Mange et al., 1997b).

A major advantage of this novel two-level system is the ability to automatically detect a malfunctioning cell, and repair it, provided there is a sufficient number of reserve molecules (within the

```
                              1 5
                              1 5
                              1 5
6 7 6 6 5 5 5 5 5 5 5 2 5 5 5 5 5 5 5 5 5 5 5 5                    2                                 1
1 1 1 6 1 1 5 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5       1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    7 1                                        1 5     1   1                                        1
    6 1                                        1 5                                                  1
    6 1   5           5 5 5 5   5              1 5                                                  1
    6 1   5           5       5               1 5                                                  1
    6 1   5           5       5               1 7                                                  1
    6 1   5           5       5               1 2                                                  1 2
    6 1   5           5 5 5 5   5              1 7                                                  1
    6 1   5                   5   5           1 5                                                  1
    8 1   5                   5   5           1 6                                                  1
    6 1   5                   5   5           1 6                                                  1
    2 1   5 5 5 5     5 5 5 5     5 5 5 5      1 6                              2 1                  1
    9 1                                        1 6                                                  1
    6 1                                        1 6                                                  1
    6 1                                        1 6                                                  1
    6 1                                        1 6                                                  1
    8 1                                        1 6                                                  1
    6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   1
    6 6 6 8 6 6 7 6 6 7 6 6 2 6 9 6 6 7 6 6 6 6 6 6 6 6 6 6 5 7 3 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2
    6 1
    6 1
    6 1
```

<center>(a)</center>

```
                                                                                                1
                                                                                                1
                              1                                                                 1
1 7 2 7 5 6 6 6 6 6 6 6 6 6 6 6 7 6 6 9 6                9 2 6 8 6 6 6 6 6 6 7 6 6 5 5 5 5 5
  5 1 1 5 7 1 1 1 1 1 1 1 1 1 1 1 1 1 2                 6 1 1 5 8 1 1 1 1 1 1 1 1 1 1 1 1 2
  5 1                           1 6       4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 1   5 6           1 5
  5 1                           1 6                         6 1   5 6           1 5
  5 1   5         5 5 5 5   5   1 7                         8 1   5 9           1 5
  5 1   5         5       5     1 6                         6 1   5 6           1 5
  5 1   5         5       5     1 7                         6 1   5 6           1 5
  5 1   5         5 5 5 5   5   1 6                         6 1   5 8           1 5
  5 1   5               5   5   1 6                         8 1   5 6           1 5
  5 1   5               5   5   1 8                         6 1   5 6         6 5   1 5
  5 1   5               5   5   1 6                         6 1   5 6 6 8 6 6 7 6 5 1 5
  5 1   5 5 5 5   5 5 5 5   5 5 5 5   1 6                   7 1   5 5 5 5   5 5 5 5   1 5
  5 1                           1 6                         6 1                     1 5
  5 1                           1 6                         6 1                     1 5
  5 1                           1 8                         7 1                     1 5
  5 1                           1 6                         6 1                     1 5
  5 1                           1 6                         6 1                     1 5
2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6          2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 1 1 1
5 5 5 5 5 5 6 6 7 6 6 6 6 6 6 6 8 6 2 9 1        6 9 6 6 7 6 6 6 6 6 6 6 6 6 5 7 2 7
1                                               1
```
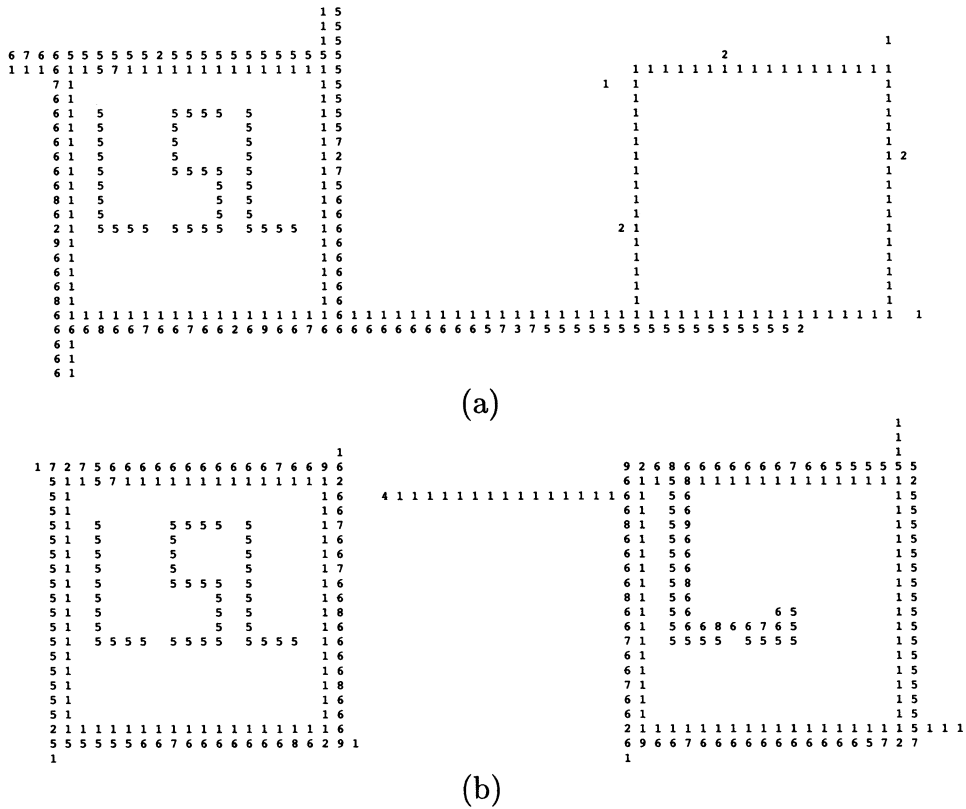
<center>(b)</center>

Fig. 9. Tempesti's loop is a self-replicating automaton, with the added capability of attaching an executable program which is duplicated and executed in each of its copies. This is demonstrated above for a simple program that writes out (after the loop's replication) LSL, acronym of the Logic Systems Laboratory. (a) Time step 240: the program is being copied into the daughter loop. (b) Time step 341: the program is being executed in the daughter loop.

cell). When these latter are exhausted, and only in this case, the cell is incapable of repair at the molecular level; it thereupon (automatically) generates a KILL signal that brings about the regenerative process at the cellular level (as described above). The automatic detection of faults is based on so-called BIST (Built-In Self-Test) techniques (Abramovici and Stroud, 1995; McCluskey, 1986).

5. L-hardware

Lindenmayer systems—or L-systems for short—were originally conceived as a mathematical theory of plant development (Lindenmayer, 1968; Prusinkiewicz and Lindenmayer, 1990). The central concept of L-systems is that of rewriting, which is essentially a technique for defining complex objects by successively replacing parts of a simple initial object using a set of rewriting rules or productions. The most ubiquitous rewriting systems operate on character strings. Though such systems first appeared at the beginning of this century (Prusinkiewicz and Lindenmayer, 1990), they have been attracting wide interest as of the 1950s with Chomsky's work on formal grammars, who applied the concept of rewriting to describe the syntactic features of natural languages (Chomsky, 1956). L-systems, introduced by Lindenmayer (1968), are string-rewriting systems, whose essential difference from Chomsky grammars lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are ap-

plied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems, with productions intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time (Prusinkiewicz and Lindenmayer, 1990).

Our L-hardware is based on the concept of L-systems, and is demonstrated below for a specific set of productions. Consider strings (words) built of two letters, *A* and *B*. Each letter is associated with a rewriting rule. The rule $A \rightarrow AB$ means that the letter *A* is to be replaced by the string *AB*, and the rule $B \rightarrow A$ means that the letter *B* is to be replaced by *A* (Prusinkiewicz and Lindenmayer, 1990). The rewriting process starts from a distinguished string called the axiom. For example, let the axiom be the single letter *B*. In the first derivation step (the first step of rewriting), axiom *B* is replaced by *A* using production $B \rightarrow A$. In the second step production $A \rightarrow AB$ is applied to replace *A* with *AB*. In the next derivation step both letters of the word *AB* are replaced simultaneously: *A* is replaced by *AB* and *B* is replaced by *A*. This process is shown in Fig. 14 for four derivation steps.

```
 . . . . . . . .
. 7 0 1 7 0 1 7 0 .
. 1 . . . . . . 1 .
. 1 .         . 7 .
. 1 .         . 0 .
. 1 .         . 1 .
. 1 .         . 7 .
. 0 . . . . . . 0 . . . .
. 4 1 0 4 1 0 7 1 0 7 1 1 .
. A . . . . . . . . . . .
. P .         . D .
. P .         . D .
. P .         . D .
. P .         . D .
. P .           .
. P .
. P .
  .
```
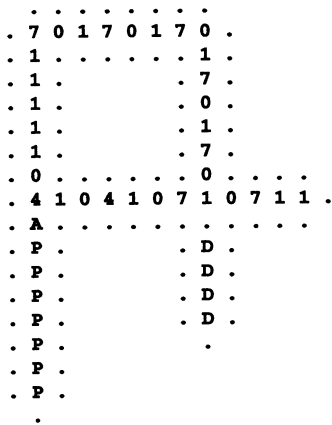
Fig. 10. A self-replicating loop with programmable capabilities. The system consists of three parts, loop, program, and data, all of which are replicated, followed by the program's execution on the given data. P denotes a state belonging to the set of program states, D denotes a state belonging to the set of data states, and A is a state which indicates the position of the program.
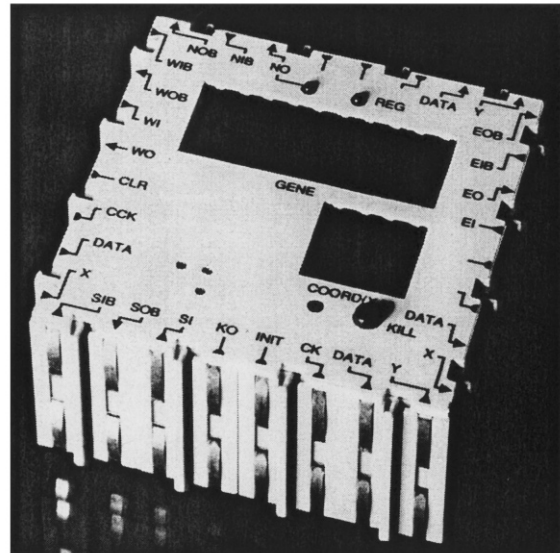

Fig. 11. An artificial cell: the biodule. A processor responsible for interpreting the genome constitutes the cytoplasm, in analogy to a ribosome, along with a random access memory (RAM) acting as the cell's nucleus, containing a copy of the entire genetic makeup, i.e. the genome. Displayed on the top cover are the cell's coordinates, as well as the specific gene within the genome that determines its functionality; these are acquired during cellular differentiation. The KILL button is used to induce the self-repair (regeneration) mechanism.

We implemented L-hardware in a one-dimensional cellular automaton with connectivity radius $r = 1$. Essentially, each cell in the CA grid represents a letter in the string. The simultaneous replacement of all letters creates a problem where the implementation is concerned since the string does not necessarily maintain its original size. Our solution is to turn the parallel replacement procedure into a sequential one—rather than replace all letters simultaneously they are replaced one at a time, with a pointer to keep track of the current letter being replaced. In order to carry out this process, two auxiliary letters are introduced, *a* and *b*, along with four additional productions: $\{A \rightarrow a, B \rightarrow b, a \rightarrow AB, b \rightarrow A\}$. In terms of CA states, this requires four pointer sub-states and five string sub-states (Fig. 15a). Basically, the sequential process works on the lower-case letters, the end result being a string of upper-case letters, thus emulating a parallel replacement procedure.
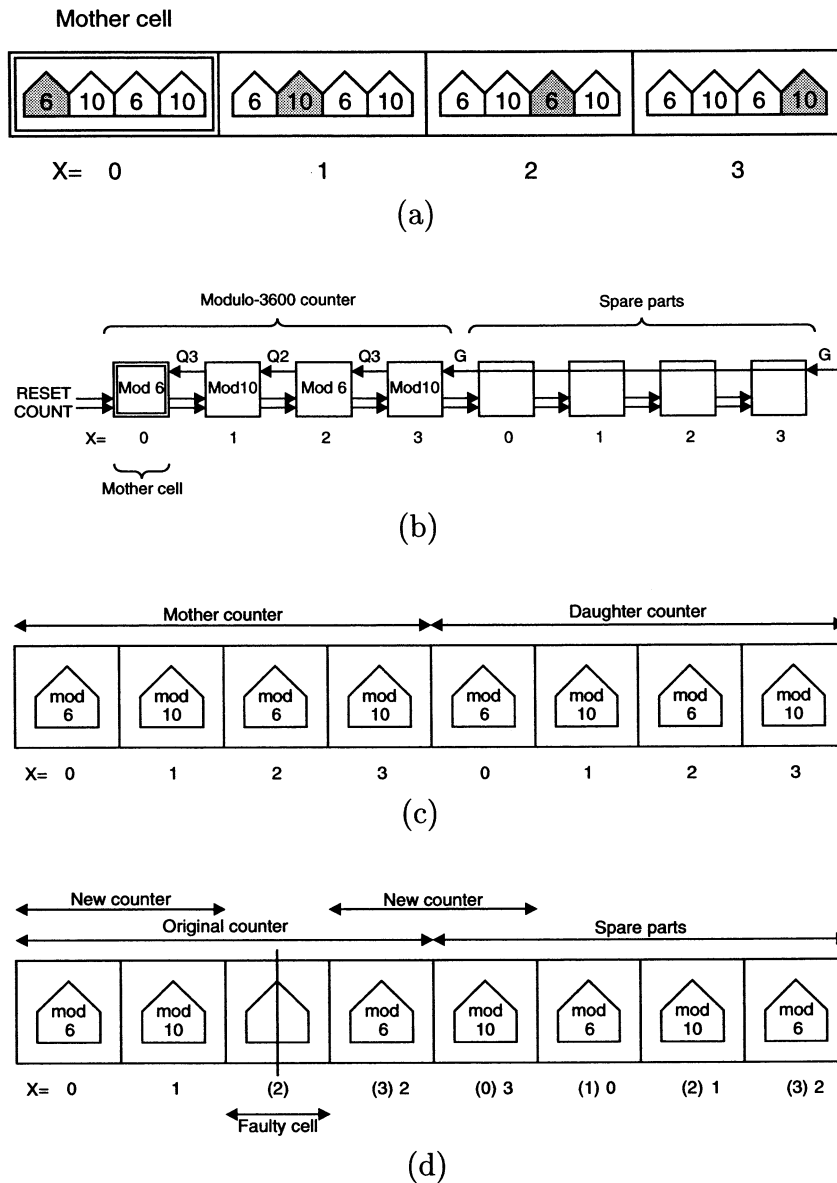
Mother cell

(a)

(b)

(c)

(d)

Fig. 12. The Biowatch. (a) The basic system architecture consists of four cells (four biodules), aligned in a row, with two cells acting as the min counter and the other two acting as the second counter. The 'organism' is shown after cellular division and cellular differentiation have taken place: each cell now holds the four-gene genome (cellular division), and has computed its coordinates, thereafter extracting the specific gene within the genome responsible for its functionality (cellular differentiation). (b) In order to demonstrate the properties of self-replication and self-repair, four additional spare cells were added to the right, with calculation of cellular coordinates producing a cycle. The modulo-3600 counter is locally synchronous—each cell's global clock arrives from one of the outputs of the counter to its right (Q2 or Q3) or from a reference clock G, with a 1 Hz frequency. (c) As the same pattern of coordinates produces the same pattern of genes, self-replication of the four-cell biowatch takes place by duplicating the pattern of coordinates within the spare cells. (d) Self-repair is achieved by displacing all cells to the right of an inactive ('dead') cell (the $X = 2$ cell above). Old coordinates are shown in parentheses.
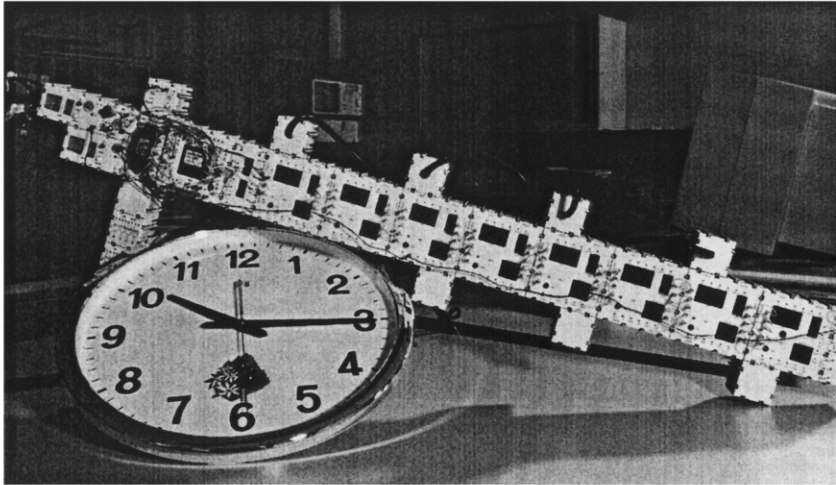
Fig. 13. An eight-cell biowatch, consisting of eight connected biodules.

A derivation step in the CA starts by respectively replacing (in parallel) *A* and *B* with *a* and *b*, thus applying the first two auxiliary productions, $A \rightarrow a$ and $B \rightarrow b$ (Fig. 15b, time step 1). The pointer, originally positioned at the leftmost cell, then moves to the right end of the string (steps 2 and 3), and then back to the left again (steps 4 and 5), introducing a space where a new letter can be written (note that pointer 'movement' is simply a visual interpretation of the state changes occurring in the CA). This is done in order to accommodate the additional letter in the derived string, required by application of rule $a \rightarrow AB$. The letter *a* can now be replaced by *AB* (step 6). The pointer keeps moving back and forth along the string (i.e. along the CA cells), working in accordance with the auxiliary productions, until all lower-case letters have been replaced by upper-case ones (steps 7–17). The return of the pointer to the leftmost cell indicates that the sequential rewriting process has ended, i.e. the original parallel derivation step has been executed.

The hardware implementation uses FPGA circuits in a similar manner to the biodule of the previous section. Each CA cell of the L-hardware requires five state bits: two bits for the pointer sub-states and three bits for the string sub-states.

6. Concluding remarks

We described research whose inspiration is drawn from the process of ontogenetic development. By adopting certain features of cellular organization, and by transposing them to the world of integrated circuits on silicon, we showed that certain properties unique to the living world, such as self-replication, self-repair, and growth, can also be attained in artificial objects (integrated circuits). We identified and described three classes of ontogenetic hardware: (1) self-replicating hardware; (2) embryonic hardware; and (3) L-hardware. For each class we presented an example of a hardware realization, and discussed some possible applications.

```
B
|
A
|
A   B
|
A   B   A
|
A   B   A   A   B
```
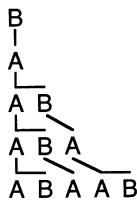
Fig. 14. Example of a derivation in an L-system. The set of productions, or rewriting rules is: $\{A \rightarrow AB, B \rightarrow A\}$. The process is shown for four derivation steps.
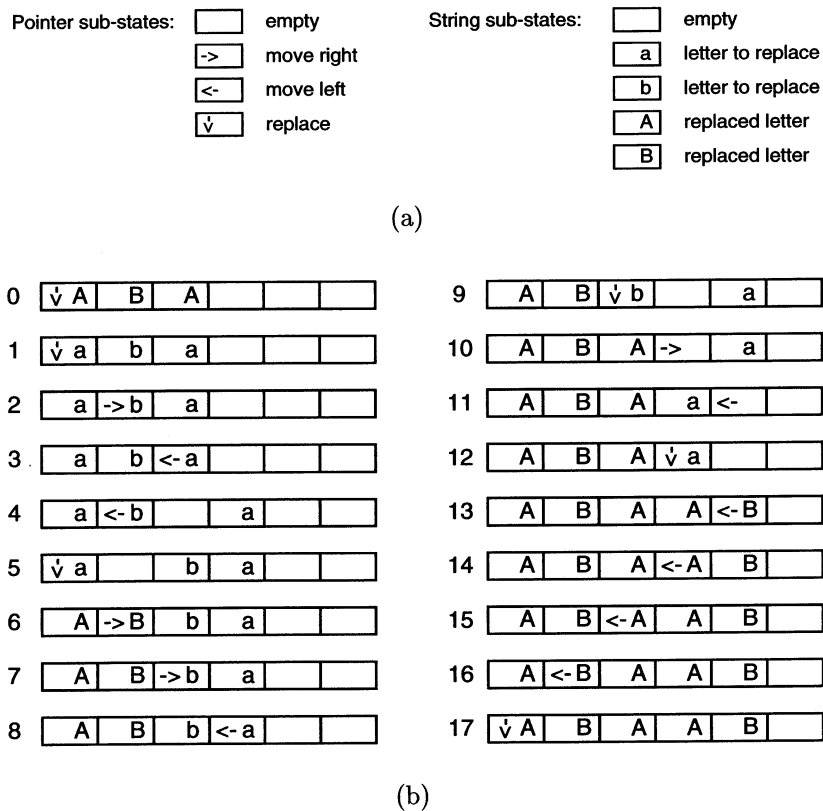
Pointer sub-states: ☐ empty   String sub-states: ☐ empty

-> move right   a letter to replace

<- move left   b letter to replace

↓ replace   A replaced letter

   B replaced letter

(a)

```
0  ↓A  B   A                    9   A   B  ↓b        a
1  ↓a  b   a                    10  A   B   A  ->    a
2   a  ->b a                    11  A   B   A   a  <-
3   a   b  <-a                  12  A   B   A   A ↓a
4   a  <-b     a                13  A   B   A   A  <-B
5  ↓a      b   a                14  A   B   A   A  <-A  B
6   A  ->B b   a                15  A   B  <-A  A   B
7   A   B  ->b a                16  A  <-B  A   A   B
8   A   B   b  <-a              17  ↓A  B   A   A   B
```

(b)

Fig. 15. L-hardware is implemented in a one-dimensional cellular automaton with connectivity radius $r = 1$. Since the string does not necessarily maintain its original size, the parallel replacement procedure is executed sequentially, using two auxiliary letters, a and b, along with four additional productions: $\{A \rightarrow a, B \rightarrow b, a \rightarrow AB, b \rightarrow A\}$. Basically, the sequential process works on the lower-case letters, with a pointer to keep track of the current letter being replaced, the end result being a string of upper-case letters. (a) The state of a CA cell is composed of two sub-states: pointer and string. For example, state '$\downarrow A$' means that the pointer is situated in this cell, and that the cell holds the letter A. (b) Demonstration of the last derivation step of Fig. 14. Note that one parallel derivation step requires, in this case, 17 CA time steps.

Several other works can be placed along the ontogenetic axis, including, e.g. cellular encoding (Gruau, 1996), graph generation systems (Kitano, 1990), and self-replicating programs (Koza, 1994). We have not discussed these as they are currently implemented solely in software, while our emphasis was on hardware systems. These works can provide a basis for future research. Other lines of investigation which suggest themselves concern the melange of the different concepts presented in this paper. For example, we discussed the incorporation of the self-replication idea within the embryonic-hardware framework. Another possibility, which we are currently exploring, is the combination of L-hardware and embryonic hardware, to attain more robust systems.

Another extension which suggests itself is the combination of two and ultimately all three axes of the POE model (Section 1) in order to attain novel bio-inspired hardware, as discussed by Sipper et al. (1997) and Sanchez et al. (1997). For example, Sipper and Tomassini (1996a,b) and Sipper (1997) evolved, via an evolutionary algorithm known as cellular programming, non-uniform cellular automata to act as random number generators. Mange et al. (1996) showed that such evolved generators can be implemented by a multicellular automaton that exhibits self-replication

and self-repair. Thus, the eventual combination of these two projects can be considered to be in the phylogenetic-ontogenetic plane. Perez and Sanchez (1996a,b) have developed neural network hardware that implements an unsupervised clustering algorithm, where the network's topology changes dynamically. Currently, the topology changes through epigenetic (neural network) mechanisms, however, one can imagine extending this system into the ontogenetic–epigenetic plane by obtaining the topology via an ontogenetic process (Kitano, 1990).

Continued research on ontogenetic hardware may yield novel bio-inspired systems endowed with replicative, growth, and regenerative capabilities. Such systems hold potential both scientifically, as vehicles for studying natural phenomena, as well as practically, showing a range of ensuing applications.

Acknowledgements

References

Abramovici, M., Stroud, C., 1995. No-overhead BIST for FPGAs. Proc. 1st IEEE Int. On-Line Testing Workshop, pp. 90–92.

Banks, E.R., 1970. Universality in cellular automata. IEEE 11th Ann. Symp. on Switching and Automata Theory. Santa Monica, CA, pp. 194–215.

Beuchat, J.-L., Haenni, J.-O., 1997. Von Neumann's 29-State Cellular Automaton: A Hardware Implementation. Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland (submitted).

Burks, A. (Ed.), 1970. Essays on Cellular Automata. University of Illinois Press, Urbana, IL.

Byl, J., 1989. Self-reproduction in small cellular automata. Physica D 34, 295–299.

Chomsky, N., 1956. Three models for the description of language. IRE Trans. Info. Theory 2 (3), 113–124.

Codd, E.F., 1968. Cellular Automata. Academic Press, New York.

Dawkins, R., 1989. The evolution of evolvability. In: Langton, C.G. (Ed.), Artificial Life, Vol. VI of SFI Studies in the Sciences of Complexity. Addison–Wesley, Reading, MA, pp. 201–220.

Drexler, K.E., 1989. Biological and nanomechanical systems: Contrasts in evolutionary capacity. In: Langton, C.G. (Ed.), Artificial Life, Vol. VI of SFI Studies in the Sciences of Complexity. Addison–Wesley, Reading, MA, pp. 501–519.

Freitas, Jr., R.A., Gilbreath, W.P. (Eds.), 1980. Advanced automation for space missions. Proc. 1980 NASA/ASEE Summer Study, Ch. 5, Replicating Systems Concepts: Self-replicating Lunar Factory and Demonstration. NASA, Scientific and Technical Information Branch (available from US G.P.O.), Washington D.C.

Gruau, F., 1996. Artificial cellular development in optimization and compilation. In: Sanchez, E., Tomassini, M. (Eds.), Towards Evolvable Hardware, Vol. 1062 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 48–75.

Hopcroft, J.E., Ullman, J.D., 1979. Introduction to Automata Theory Languages and Computation. Addison–Wesley, Redwood City, CA.

Kitano, H., 1990. Designing neural networks by genetic algorithms using graph generation systems. Complex Syst. 4, 461–476.

Koza, J.R., 1994. Artificial life: Spontaneous emergence of self-replicating and evolutionary self-improving computer programs. In: Langton, C.G. (Ed.), Artifiicial Life III, Vol. XVII of SFI Studies in the Sciences of Complexity. Addison–Wesley, Reading, MA, pp. 225–262.

Langton, C.G., 1984. Self-reproduction in cellular automata. Physica D 10, 135–144.

Lindenmayer, A., 1968. Mathematical models for cellular interaction in development, Parts I and II. J. Theor. Biol. 18, 280–315.

Mange, D., Goeke, M., Madon, D., Stauffer, A., Tempesti, G., Durand, S., 1996. Embryonics: A new family of coarse-grained field-programmable gate array with self-repair and self-reproducing properties. In: Sanchez, E., Tomassini, M. (Eds.), Towards Evolvable Hardware, Vol. 1062 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 197–220.

Mange, D., Madon, D., Stauffer, A., Tempesti, G., 1997a. Von Neumann revisited: A Turing machine with self-repair and self-reproduction properties. Robotics and Autonomous Systems (to appear).

Mange, D., Stauffer, A., Tempesti, G., 1997b. Self-replicating and self-repairing Field-Programmable Processor Arrays

(FPPAs) with universal construction. Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)—Workshop on Evolvable Systems. Morgan Kaufmann, San Mateo, CA (to appear).

Mange, D., Stauffer, A., 1994. Introduction to embryonics: Towards new self-repairing and self-reproducing hardware based on biological-like properties. In: Thalmann, N.M., Thalmann, D. (Eds.), Artificial Life and Virtual Reality. Wiley, Chichester, England, pp. 61–72.

Marchal, P., Piguet, C., Mange, D., Stauffer, A., Durand, S., 1994. Embryological development on silicon. In: Brooks, R.A., Maes, P. (Eds.), Artificial Life IV. MIT Press, Cambridge, MA, pp. 365–370.

Marchal, P., Nussbaum, P., Piguet, C., Durand, S., Mange, D., Sanchez, E., Stauffer, A., Tempesti, G., 1996. Embryonics: The birth of synthetic life. In: Sanchez, E., Tomassini, M. (Eds.), Towards Evolvable Hardware, Vol. 1062 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 166–196.

McCluskey, E.J., 1986. Logic Design Principles with Emphasis on Testable Semicustom Circuits. Prentice–Hall, Englewood Cliffs, NJ.

Mitchell, M., 1996. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.

Morita, K., Imai, K., 1997. Logical universality and self-reproduction in reversible cellular automata. In: Higuchi, T., Iwata, M., Weixin, L. (Eds.), Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES96), Vol. 1259 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 152–166.

Perez, A., Sanchez, E., 1996a. FPGA implementation of an adaptable-size neural network. In: von der Malsburg, C., von Seelen, W., Vorbruggen, J.C., Sendhoff, B. (Eds.), Proc. Int. Conf. on Artificial Neural Networks (ICANN96), Vol. 1112 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 383–388.

Perez, A., Sanchez, E., 1996b. Neural networks structure optimization through on-line hardware evolution. In: Proc. World Cong. on Neural Networks (WCNN96), INNS (International Neural Networks Society) Press, pp. 1041–1044.

Perrier, J.-Y., Sipper, M., Zahnd, J., 1996. Toward a viable, self-reproducing universal computer. Physica D 97, 335–352.

Pesavento, U., 1995. An implementation of von Neumann's self-reproducing machine. Artif. Life 2 (4), 337–354.

Prusinkiewicz, P., Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer–Verlag, New York.

Ransom, R.J., 1981. Computers and Embryos: Models in Developmental Biology. Wiley, Chichester.

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

Sanchez, E., 1996. Field-programmable gate array (FPGA) circuits. In: Sanchez, E., Tomassini, M. (Eds.), Towards Evolvable Hardware, Vol. 1062 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 1–18.

Sanchez, E., Mange, D., Sipper, M., Tomassini, M., Perez-Uribe, A., Stauffer, A., 1997. Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In: Higuchi, T., Iwata, M., Weixin, L. (Eds.), Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES96), Vol. 1259 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 35–54.

Sanchez, E., Tomassini, M. (Eds.), 1996. Towards Evolvable Hardware, Vol. 1062 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg.

Sipper, M., 1997. Evolution of Parallel Cellular Machines: The Cellular Programming Approach. Springer–Verlag, Heidelberg.

Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Perez-Uribe, A., Stauffer, A., 1997. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. IEEE Trans. Evol. Comput. 1 (1), 83–97.

Sipper, M., Tomassini, M., 1996a. Co-evolving parallel random number generators. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (Eds.), Parallel Problem Solving from Nature—PPSN IV, Vol. 1141 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 950–959.

Sipper, M., Tomassini, M., 1996b. Generating parallel random number generators by cellular programming. Int. J. Mod. Phys. C 7 (2), 181–190.

Stauffer, A., Mange, D., Goeke, M., Madon, D., Tempesti, G., Durand, S., Marchal, P., Nussbaum, P., 1997. FPPA: A field-programmable processor array with biological-like properties. In: Hartenstein, R.W., Prasanna, V.K. (Eds.), 4th Reconfigurable Architectures Workshop (RAW '97). IT press Verlag, Bruchsal, pp. 45–48.

Stauffer, A., Mange, D., Goeke, M., Madon, D., Tempesti, G., Durand, S., Marchal, P., Piguet, C., 1996. MICRO-TREE: Towards a binary decision machine-based FPGA with biological-like properties. Proc. Int. Workshop on Logic and Architecture Synthesis. Institut National Polytechnique de Grenoble, France.

Tempesti, G., 1995. A new self-reproducing cellular automaton capable of construction and computation. In: Moran, F., Moreno, A., Merelo, J.J., Chacon, P. (Eds.), ECAL '95: 3rd Eur. Conf. on Artifiicial Life, Vol. 929 of Lecture Notes in Computer Science. Springer–Verlag, Heidelberg, pp. 555–563.

Toffoli, T., Margolus, N., 1987. Cellular Automata Machines. MIT Press, Cambridge, MA.

von Neumann, J., 1966. Theory of Self-Reproducing Automata. Edited and completed by A.W. Burks. University of Illinois Press, IL.

Watson, J.D., Hopkins, N.H., Roberts, J.W., Steitz, J.A., Weiner, A.M., 1987. Molecular Biology of the Gene, 4th ed. Benjamin/Cummings, Menlo Park, CA.

Wolpert, L., 1991. The Triumph of the Embryo. Oxford University Press, New York.