Proceedings of the IEEE, vol. 88, no. 4, April 2000, pp. 516-541.

Towards Robust Integrated Circuits:

The Embryonics Approach

Daniel Mange, Member IEEE

Moshe Sipper, Senior Member IEEE

André Stauffer, Member IEEE

Gianluca Tempesti, Member IEEE

Logic Systems Laboratory
Swiss Federal Institute of Technology
Lausanne, Switzerland

D. Mange, M. Sipper, A. Stauffer, and G. Tempesti are with the Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, CH-1015 Switzerland. Email: name.surname@epfl.ch. URL: http://lslwww.epfl.ch. Phone: +41 21 693 2639. Fax: +41 21 693 37 05.

Abstract

The growth and operation of all living beings are directed by the interpretation, in each of their cells, of a chemical program, the DNA string or *genome*. This process is the source of inspiration for the Embryonics (embryonic electronics) project, whose final objective is the design of highly robust integrated circuits, endowed with properties usually associated with the living world: self-repair (cicatrization) and self-replication. The Embryonics architecture is based on four hierarchical levels of organization: 1) the basic primitive of our system is the *molecule*, a multiplexer-based element of a novel programmable circuit; 2) a finite set of molecules makes up a *cell*, essentially a small processor with an associated memory; 3) a finite set of cells makes up an *organism*, an application–specific multiprocessor system; 4) the organism can itself replicate, giving rise to a *population* of identical organisms. We begin by describing in detail the implementation of an artificial cell characterized by a *fixed architecture*, showing that multicellular arrays can realize a variety of different organisms, all capable of self-replication and self-repair. In order to allow for a wide range of applications, we then introduce a *flexible architecture*, realized using a new type of fine-grained FPGA (field-programmable gate array) whose basic element, our molecule, is essentially a programmable multiplexer. We describe the implementation of such a molecule, with built-in self-test, and illustrate its use in realizing two applications: a modulo-4 reversible counter (a unicellular organism) and a timer (a complex multicellular organism). Finally, we describe our ongoing research efforts to meet three challenges: a scientific challenge, that of implementing the original specifications formulated by John von Neumann for the conception of a self-replicating automaton; a technical challenge, that of realizing very robust integrated circuits capable of self-repair and self-replication; and a biological challenge, that of attempting to show that the microscopic architecture of artificial and natural organisms, i.e., their genomes, share common properties.

Keywords:

Embryonic electronics, field-programmable gate arrays (FPGAs), multiplexer-based FPGAs, built-in self-test, self-repairing FPGAs, self-replicating FPGAs.

I. Introduction

A. Towards Embryonics

A human being consists of approximately 60 trillion ($60x10^{12}$) cells. At each instant, in each of these 60 trillion cells, the *genome*, a ribbon of 2 billion characters, is decoded to produce the proteins needed for the survival of the organism. This genome contains the ensemble of the genetic inheritance of the individual and, at the same time, the instructions for both the construction and the operation of the organism. The parallel execution of 60 trillion genomes in as many cells occurs ceaselessly from the conception to the death of the individual. Faults are rare and, in the majority of cases, successfully detected and repaired. This process is remarkable for its complexity and its precision. Moreover, it relies on completely discrete information: the structure of DNA (the chemical substrate of the genome) is a sequence of four bases, usually designated with the letters A (adenine), C (cytosine), G (guanine), and T (thymine).

Our *Embryonics* project (for *embryonic electronics*) is inspired by the basic processes of molecular biology and by the embryonic development of living beings [1][43]. By adopting certain features of cellular organization, and by transposing them to the two-dimensional world of integrated circuits on silicon, we will show that properties unique to the living world, such as *self-replication* and *self-repair*, can also be applied to artificial objects (integrated circuits). We wish however to emphasize that the goal of bio-inspiration is *not* the modelization or the explication of actual biological phenomena.

B. Objectives and Strategy

Our final objective is the development of very large scale integrated (VLSI) circuits capable of self-repair and self-replication. Self-repair allows partial reconstruction in case of a minor fault, while self-replication allows complete reconstruction of the original device in case of a major fault. These two properties are particularly desirable for complex artificial systems requiring improved reliability in short, medium, or long term applications.

1. Short term applications [2]:

- Applications which require very high levels of reliability, such as avionics or medical electronics.
- Applications designed for hostile environments, such as space, where the increased radiation levels reduce the reliability of components.
- Applications which exploit the latest technological advances, and notably the drastic device shrinking, low power supply levels, and increasing operating speeds, which accompany the technological evolution to deeper submicron levels and significantly reduce the noise margins and increase the soft-error rates [33].

2. Medium term applications, where our aim is to develop very complex integrated circuits capable of on-line self-repair, dispensing with the systematic detection of faults at fabrication, impossible for systems consisting of millions of logic gates [34].
3. Long term applications, executed on systems built with imperfect components: this is von Neumann's historical idea [8], the basis of all present projects aimed at the realization of complex integrated circuits at the atomic scale (nanotechnologies) [35][36][37][38].

Self-replication, or "cloning", can be justified independently of self-repair:

- to replicate, within a field-programmable gate array (FPGA), functionally equivalent systems [39];
- to produce the massive quantity of future integrated circuits, implemented using nanotechnologies [30];
- to finally accomplish John von Neumann's unachieved dream, that is, the realization of a self-replicating automaton endowed with the properties of universal computation and construction [8].

These emerging needs require the development of a new design paradigm that supports efficient online VLSI testing and self-repair solutions. Inspired by the architecture of living beings, we will show how to implement online testing, self-repair, and self-replication using both hardware and software redundancy. The programmable degree of robustness of our systems, function of an overhead itself programmable, is one of the major original features of the Embryonics project.

Section II provides a bird's eye view of Embryonics. This Section is self-contained so as to give the reader a general overview of our project without delving into the technical details of the following Sections. The final Embryonics architecture is based on four hierarchical levels of organization:

- The basic primitive of our system is the *molecule*, the element of a novel programmable circuit, built around a multiplexer.
- A finite set of molecules makes up a *cell*, essentially a small processor with the associated memory.
- A finite set of cells makes up an *organism*, an application-specific multiprocessor system.
- The organism can itself replicate, giving rise to a *population* of identical organisms, the highest level of our hierarchy.

Section III describes in detail the implementation of a prototype of an artificial cell characterized by a *fixed architecture*. We then show that multicellular arrays can realize a variety of different organisms (electronic watch, random number generator, Turing machine), all capable of self-replication and self-repair.

We will see that, to meet the requirements of a wide range of applications, we need to develop an architecture characterized by a *flexible architecture*, that is, an architecture which is itself reconfigurable. This architecture will be based on a new type of fine-grained FPGA (field-programmable gate array) whose basic element, the *molecule*, is essentially a programmable multiplexer. Section IV describes the implementation of such a molecule, with built-in self-test, and shows its use for two applications of very different complexity: a modulo-4 reversible counter and a timer.

The main goal of the Embryonics project is to explore the potential of a novel, robust architecture, rather than to compare such an architecture with existing solutions. After describing the trials and errors of the project, the conclusion of Section V introduces our ongoing research along three axes, each representing a different challenge: a scientific challenge, that of implementing in our architecture the original specifications formulated by John von Neumann for the conception of a self-replicating automaton; a technical challenge, that of realizing real integrated circuits, capable of self-repair and of self-replication; and a biological challenge, that of seeking to show that the microscopic architecture of the artificial and natural organisms, that is, the structure of their genomes, share some common properties.

II. A Bird's Eye View of Embryonics

A. From Biology to Hardware

The majority of living beings, with the exception of unicellular organisms such as viruses and bacteria, share three fundamental features:

1. *Multicellular organization* divides the organism into a finite number of *cells*, each realizing a unique function (neuron, muscle, intestine, etc.). The same organism can contain multiple cells of the same kind.
2. *Cellular division* is the process whereby each cell (beginning with the first cell or *zygote*) generates one or two daughter cells. During this division, all of the genetic material of the mother cell, the *genome*, is copied into the daughter cell(s).
3. *Cellular differentiation* defines the role of each cell of the organism, that is, its particular function (neuron, muscle, intestine, etc.). This specialization of the cell is obtained through the expression of part of the genome, consisting of one or more *genes*, and depends essentially on the physical position of the cell in the organism.

A consequence of these three features is that each cell is "universal", since it contains the whole of the organism's genetic material, the genome. Should a minor (wound) or major (loss of an organ) trauma occur, living organisms are thus potentially capable of self-repair (cicatrization) or self-replication (cloning or budding) [1].

The two properties of self-repair and self-replication based on a multicellular tissue are unique to the living world. The main goal of the Embryonics project is the implementation of the above three features of living organisms in an integrated circuit in silicon, in order to obtain the properties of self-repair and self-replication.

Our artificial organism will ultimately be divided into cells, themselves decomposed into molecules, a structure which determines the plan of this Section: Subsection B describes the three fundamental features of the organism (multicellular organization, cellular differentiation, and cellular division), while in Subsection C we demonstrate that the organism, thanks to these three features, exhibits the two sought properties (self-replication and self-repair). Subsection D describes the cells and presents their essential features (multimolecular organization, molecular configuration, and molecular error detection), while Subsection E shows that the two sought properties (self-replication and self-repair) apply both at the cellular level as well as at the

organismic level. Subsection F underscores the four organizational levels of the hierarchy of the Embryonics project which are, from the bottom up, the molecule, the cell, the organism, and finally the population of organisms.

B. The Organism's Features: Multicellular Organization, Cellular Differentiation, and Cellular Division

The environment in which our quasi-biological development occurs is imposed by the structure of electronic circuits, and consists of a finite (but arbitrarily large) two-dimensional surface of silicon. This surface is divided into rows and columns, whose intersections define the cells. Since such cells (small processors and their memory) have an identical physical structure (i.e., an identical set of logic operators and of connections), the cellular array is homogeneous. As the program in each cell (our artificial genome) is identical, only the state of a cell (i.e., the contents of its registers) can differentiate it from its neighbors.

In this Section, we first show how to implement in our artificial organisms the three fundamental features of multicellular organization, cellular differentiation, and cellular division, by using a generic and abstract six-cell example. In the following Sections (Sections III and IV), we will propose an actual implementation and various applications.

Multicellular organization divides the artificial organism (`ORG`) into a finite number of cells (Fig. 2.1). Each cell (`CELL`) realizes a unique function, defined by a sub-program called the *gene* of the cell and selected as a function of the values of both the horizontal (`X`) and the vertical (`Y`) coordinates (in Fig. 2.1, the genes are labeled `A` to `F` for coordinates `X,Y=1,1` to `X,Y=3,2`). Our final artificial genome will be divided into three main parts: the *operative genome* (`OG`), the *ribosomic genome* (`RG`), and the *polymerase genome* (`PG`). Let us call operative genome (`OG`) a program containing all the genes of an artificial organism, where each gene (`A` to `F`) is a sub-program characterized by a set of instructions and by the cell's position (coordinates `X,Y=1,1` to `X,Y=3,2`). Fig. 2.1 is then a graphical representation of organism `ORG`'s operative genome.

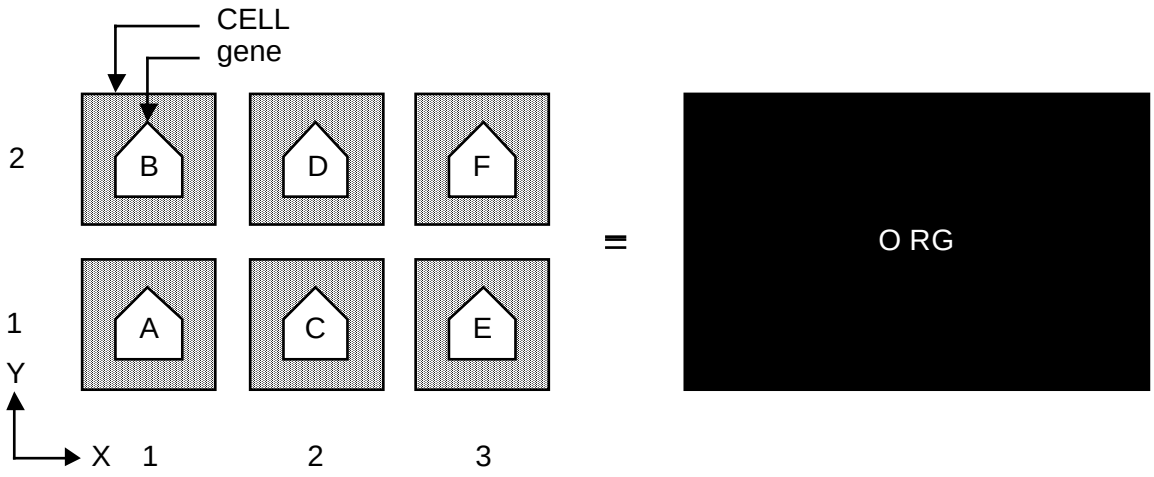


Fig. 2.1 Multicellular organization of a 6-cell organism ORG.

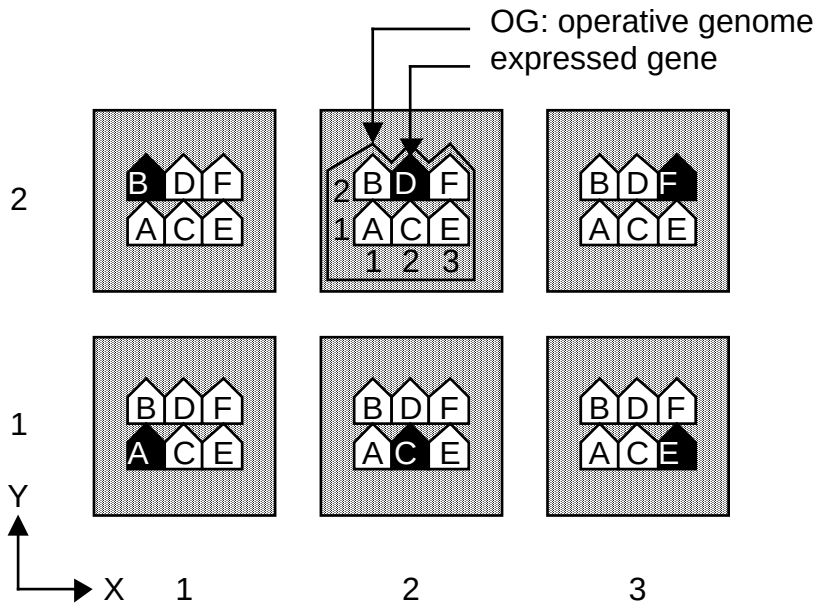
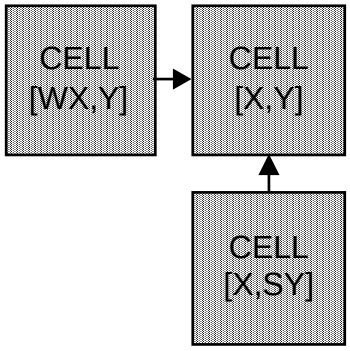


Fig. 2.2a

Let then each cell contain the entire operative genome OG (Fig. 2.2a): depending on its position in the array, i.e., its place within the organism, each cell can then interpret the operative genome and extract and execute the gene which defines its function. In summary, storing the whole operative genome in each cell makes the cell universal: given the proper coordinates, it can execute any one of the genes of the operative genome and thus implement *cellular differentiation*. In our artificial organism, any cell CELL[X,Y] continuously computes its coordinate X by incrementing the coordinate WX of its neighbor immediately to the west (Fig. 2.2b). Likewise, it continuously computes its coordinate Y by incrementing the coordinate SY of its neighbor immediately to the south. Taking into consideration these computations, Fig. 2.3 shows the final operative genome OG of the organism ORG.



(b)

Fig. 2.2 Cellular differentiation. (a) Global organization. (b) A cell `CELL[X,Y]` with its west neighbor `CELL[WX,Y]` and its south neighbor `CELL[X,SY]`; `X=WX+1`; `Y=SY+1`.

```
OG: operative genome
X = WX+1
Y = SY+1
case of X,Y:
  X,Y = 1,1: do gene A
  X,Y = 1,2: do gene B
  X,Y = 2,1: do gene C
  X,Y = 2,2: do gene D
  X,Y = 3,1: do gene E
  X,Y = 3,2: do gene F
```

Fig. 2.3 The operative genome `OG` of the organism `ORG`.

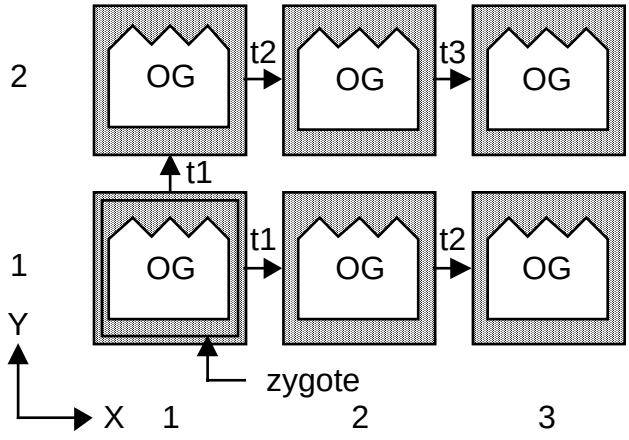


Fig. 2.4 Cellular division; `OG`: operative genome; `t1` … `t3`: three cellular divisions.

At startup, the first cell or *zygote* (Fig. 2.4), arbitrarily defined as having the coordinates `X,Y=1,1`, holds the one and only copy of the operative genome `OG`. After time `t1`, the genome of the zygote (*mother* cell) is copied into the neighboring (*daughter*) cells to the east (`CELL[2,1]`) and to the north (`CELL[1,2]`). This process of *cellular division* continues until the six cells of the organism `ORG` are completely programmed (in our example, the farthest cell is programmed after time `t3`).

C. The Organism's Properties: Organismic Self-Replication and Organismic Self-Repair

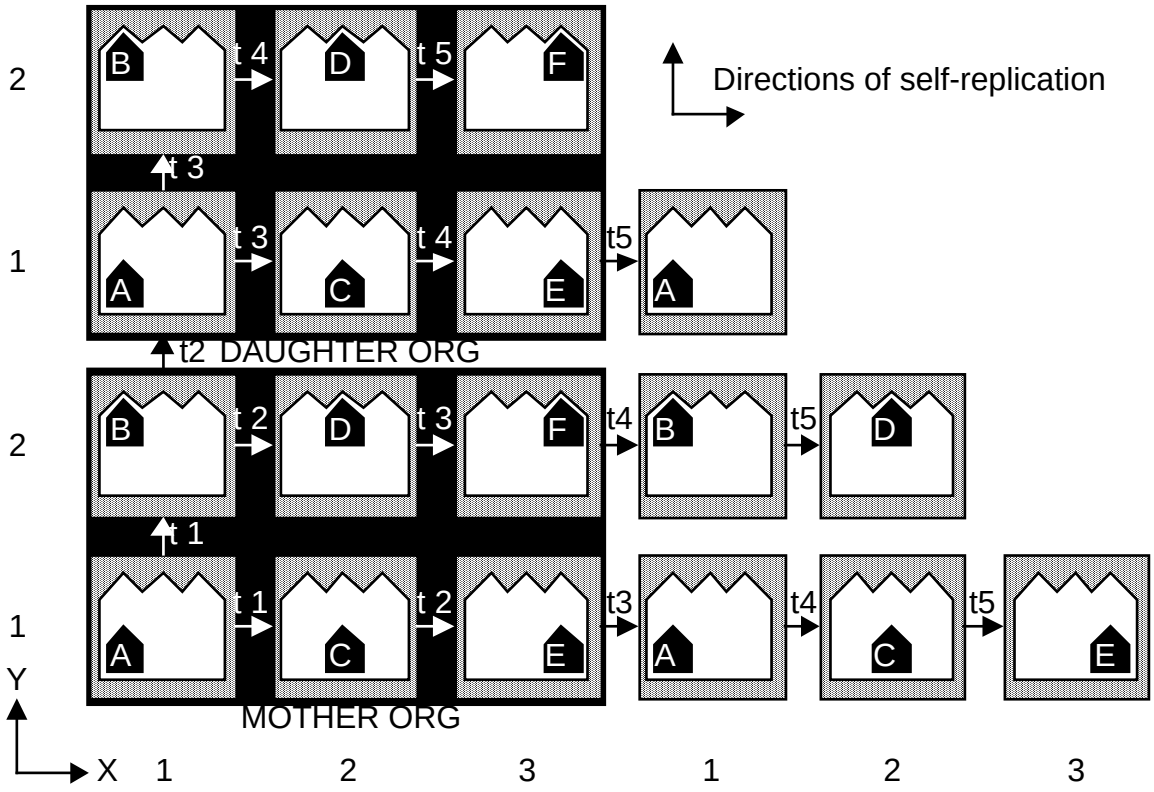


Fig. 2.5 Self-replication of a 6-cell organism `ORG` in a limited homogeneous array of 6x4 cells (situation at time `t5` after 5 cellular divisions);
`MOTHER ORG` = mother organism; `DAUGHTER ORG` = daughter organism.

The *self-replication* or *cloning of the organism*, i.e., the production of an exact copy of the original, rests on two assumptions:

- there exists a sufficient number of spare cells in the array (at least six in the example of Fig. 2.5) to contain the additional organism;

- the calculation of the coordinates produces a cycle (X=1→2→3→1… and Y=1→2→1… in Fig. 2.5, implying X=(WX+1) modulo 3 and Y=(SY+1) modulo 2).

As the same pattern of coordinates produces the same pattern of genes, self-replication can be easily accomplished if the program of the operative genome OG, associated with the homogeneous array of cells, produces several occurrences of the basic pattern of coordinates. In our example (Fig. 2.5), the repetition of the vertical coordinate pattern (Y=1→2→1→2) in a sufficiently large array of cells produces one copy, the *daughter organism*, of the original *mother organism*. Given a sufficiently large space, the self-replication process can be repeated for any number of specimens in the X and/or the Y axes.

In order to implement the *self-repair of the organism*, we decided to use spare cells to the right of the original organism (Fig. 2.6). The existence of a fault is detected by a KILL signal which is calculated in each cell by a built-in self-test mechanism realized at the molecular level (see Subsection E below). The state KILL=1 identifies the faulty cell, and the entire column to which the faulty cell belongs is considered faulty, and is deactivated (column X=2 in Fig. 2.6). All the functions (X coordinate and gene) of the cells to the right of the column X=1 are shifted by one column to the right. Obviously, this process requires as many spare columns to the right of the array as there are faulty cells or columns to repair (two spare columns, tolerating two successive faulty cells, in the example of Fig. 2.6). It also implies that the cell needs to be able to bypass the faulty column and to divert to the right all the required signals (such as the operative genome and the X coordinate, as well as the data busses).

Given a sufficient number of cells, it is obviously possible to combine self-repair in the X direction, and self-replication in both the X and Y directions.

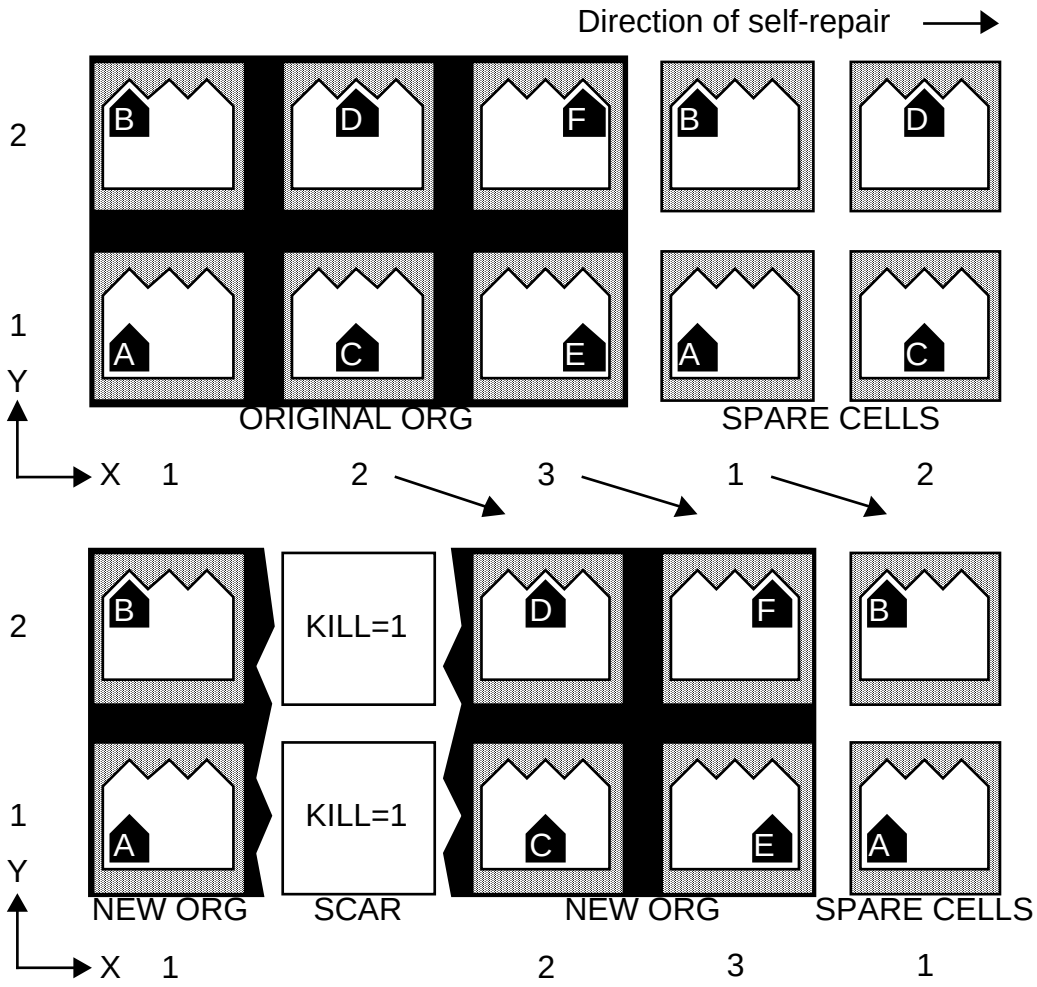


Fig. 2.6 Organismic self-repair.

D. The Cell's Features: Multimolecular Organization, Molecular Configuration, and Molecular Fault Detection

In each cell of every living being, the genome is translated sequentially by a chemical processor, the *ribosome*, to create the proteins needed for the organism's survival. The ribosome itself consists of molecules, whose description is an important part of the genome.

As mentioned, in the Embryonics project each cell is a small processor, sequentially executing the instructions of a first part of the artificial genome, the operative genome OG. The need to realize organisms of varying degrees of complexity has led us to design an artificial cell characterized by a flexible architecture, that is, itself configurable. It will therefore be implemented using a new kind of fine-grained, field-programmable gate array (FPGA).

Each element of this FPGA (consisting essentially of a multiplexer associated with a programmable connection network) is then equivalent to a *molecule*, and an

appropriate number of these artificial molecules allows us to realize application-specific processors. We will call *multimolecular organization* the use of many molecules to realize one cell. The configuration string of the FPGA (that is, the information required to assign the logic function of each molecule) constitutes the second part of our artificial genome: the *ribosomic genome* `RG`. Fig. 2.7a shows a generic and abstract example of an extremely simple cell (`CELL`) consisting of six molecules, each defined by a *molecular code* or `MOLCODE` (`a` to `f`). The set of these six `MOLCODE`s constitutes the ribosomic genome `RG` of the cell.

The information contained in the ribosomic genome `RG` thus defines the logic function of each molecule by assigning a molecular code `MOLCODE` to it. To obtain a functional cell, we require two additional pieces of information:

- the *physical position* of each molecule in the cellular space;
- the presence of one or more *spare columns*, composed of *spare molecules*, required for the self-repair described below (Subsection E).

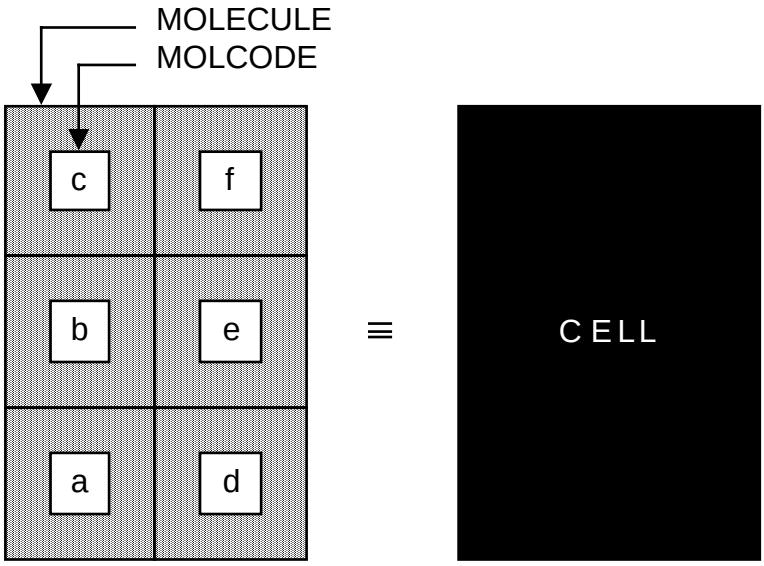
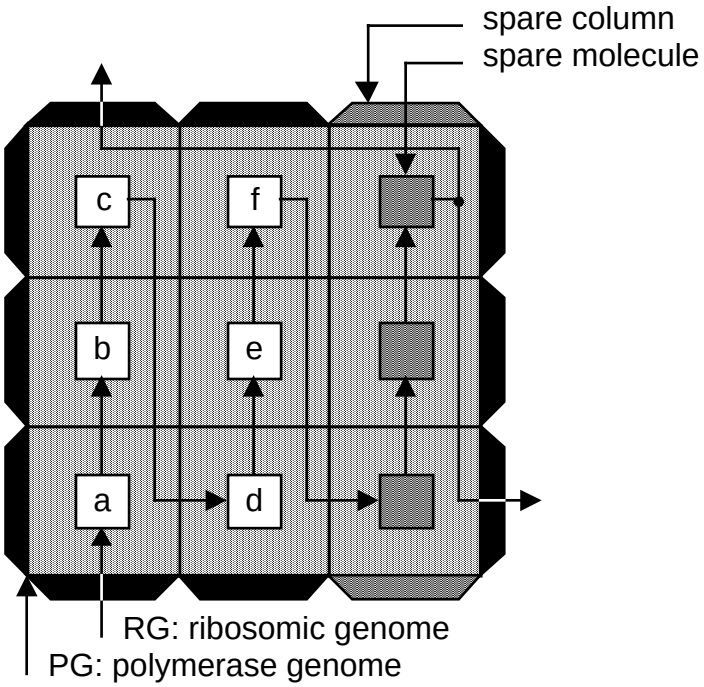
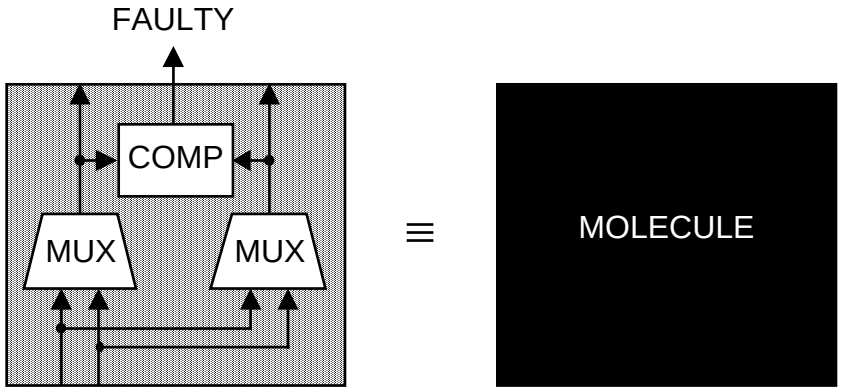


Fig. 2.7a



(b)



(c)

Fig. 2.7 The cell's features. (a) Multimolecular organization; `RG`: ribosomic genome: `a, b, c, d, e, f`. (b) Molecular configuration; `PG`: polymerase genome: height x width = 3x3; 001 = spare column. (c) Molecular fault detection; `MUX`: multiplexer; `COMP`: comparator.

The definition of these pieces of information is the *molecular configuration* (Fig. 2.7b). Their injection into the FPGA will allow:

1. the creation of a border surrounding the molecules of a given cell;
2. the insertion of one or more spare columns;
3. the definition of the connections between the molecules, required for the propagation of the ribosomic genome `RG`.

The information needed for the molecular configuration (essentially, the height and width of the cell in number of molecules and the position of the spare columns) makes up the third and last part of our artificial genome: the *polymerase genome* PG (a terminology which will be justified in Subsection E).

Finally, it is imperative to be able to automatically detect the presence of faults at the molecular level and to relay this information to the cellular level. Moreover, if we consider that the death of a column of cells is quite expensive in terms of wasted resources, the ability to repair at least some of these faults at the molecular level (that is, without invoking the organismic self-repair mechanism) becomes highly desirable. The biological inspiration for this process derives from the DNA's double helix, the physical support of natural genomes, which provides complete redundancy of the genomic information though the presence of complementary bases in the opposing branches of the helix. By duplicating the material of each molecule (essentially the multiplexer MUX) and by continuously comparing the signals produced by each of the two copies (Fig. 2.7c), it is possible to detect a faulty molecule and to generate a signal FAULTY=1, realizing the *molecular fault detection* which will make possible cellular self-repair (described below in Subsection E).

E. The Cell's Properties: Cellular Self-Replication and Cellular Self-Repair

A consequence of the multimolecular organization and of the molecular configuration of the FPGA (Subsection D and Fig. 2.7b) is the ability, for any given cell, to propagate its polymerase genome PG and its ribosomic genome RG in order to automatically configure two daughter cells, architecturally identical to the mother cell, to the east and to the north (Fig. 2.8), thus implementing *cellular self-replication*.

Cellular self-replication is a prerequisite for cellular division at the organismic level described above (Subsection B and Fig. 2.4), during which the operative genome is copied from the mother cell into the daughter cells. In living systems, a specific molecule, the *polymerase enzyme*, allows cellular replication through the duplication of the genome. It is by analogy to this enzyme that the third part of our artificial genome is called polymerase genome.

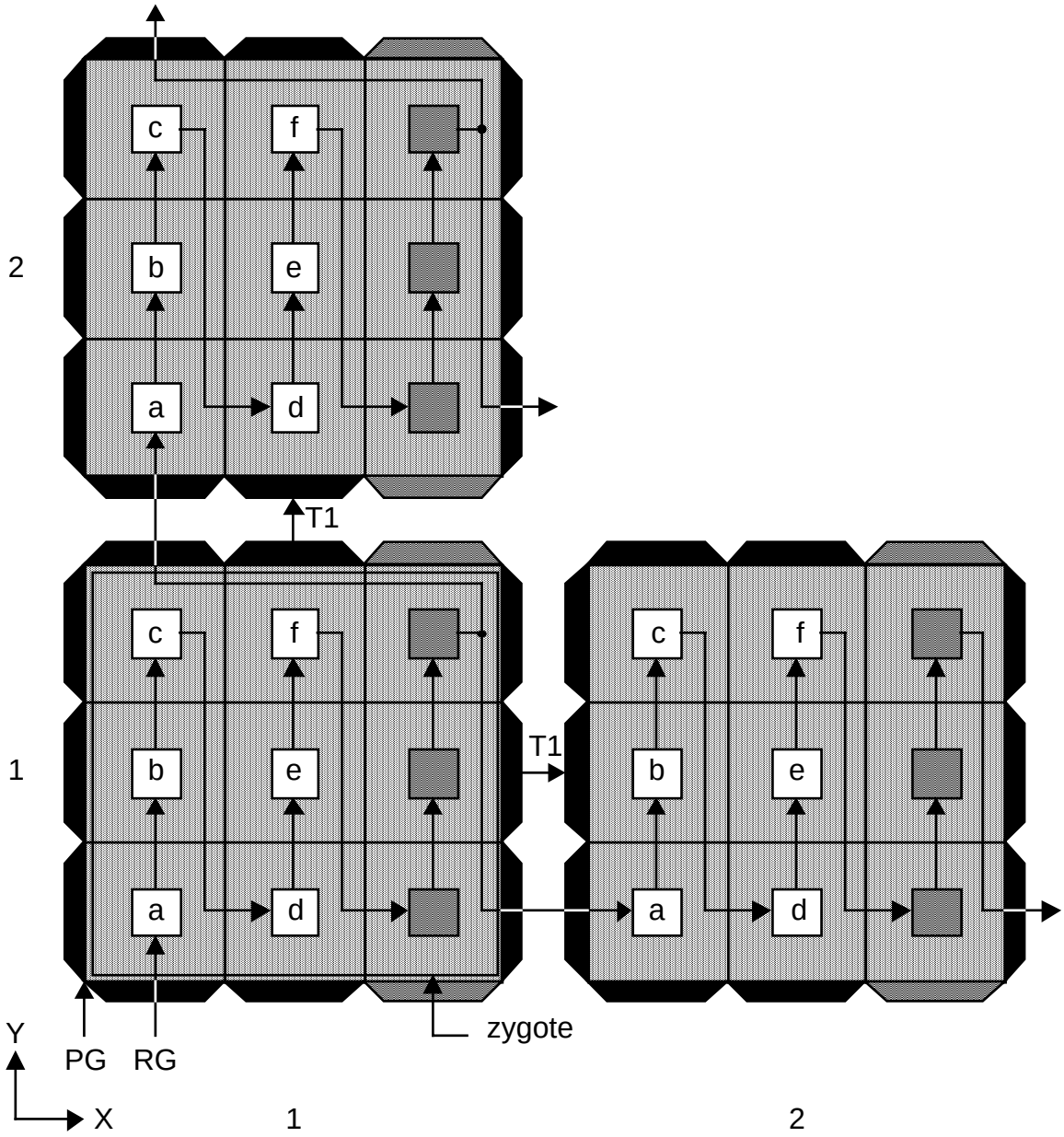


Fig. 2.8 Cellular self-replication; T1: one instance of cellular self-replication; RG: ribosomic genome; PG: polymerase genome.

The presence of spare columns, defined by the molecular configuration, and the automatic detection of faulty molecules (Subsection D, Figs. 2.7b and 2.7c) allow *cellular self-repair*: each faulty molecule is deactivated, isolated from the network, and replaced by a neighboring molecule, which will itself be replaced by a neighbor, and so on until a spare molecule (SM) is reached (Fig. 2.9a).

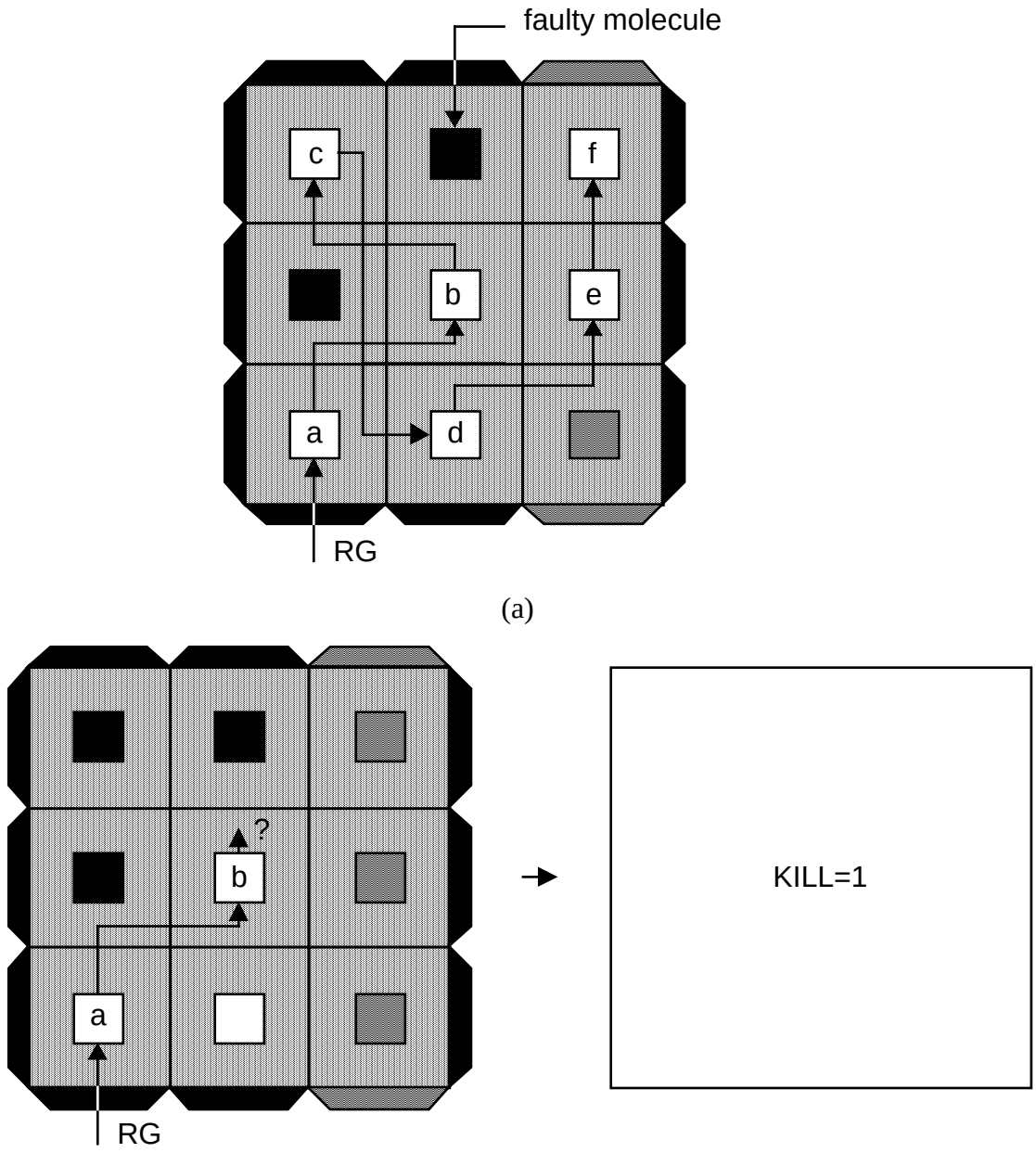


Fig. 2.9 Cellular self-repair. (a) Possible self-repair (at most one faulty molecule per row). (b) Impossible self-repair (more than one faulty molecule per row): `KILL=1` (self-repair at the organismic level).

The number of faulty molecules handled by the molecular self-repair mechanism is necessarily limited: in the example of Fig. 2.9a, we tolerate at most one faulty molecule per row. If more than one molecule is faulty in one or more rows (Fig. 2.9b), molecular self-repair is impossible, in which case a global signal `KILL=1` is generated to activate the organismic self-repair described above (Subsection C and Fig. 2.6).

F. The Embryonics Landscape

The final architecture of the Embryonics project is based on four hierarchical levels of organization which, described from the bottom up, are the following (Fig. 2.10):

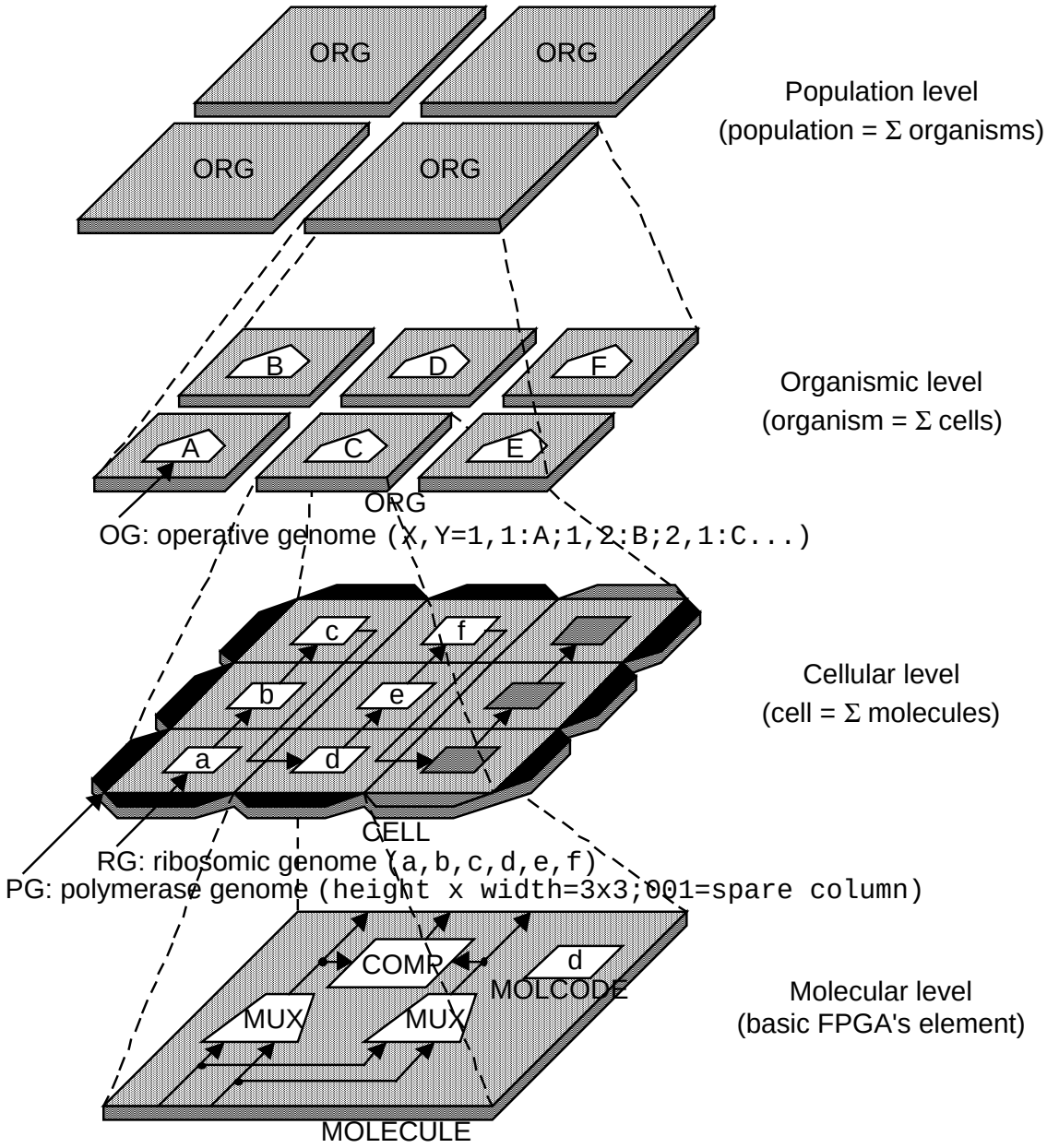


Fig. 2.10 The Embryonics landscape: a 4-level hierarchy.

- The basic primitive of our system is the *molecule*, the element of our new FPGA, consisting essentially of a multiplexer associated with a programmable connection network. The multiplexer is duplicated to allow the detection of faults. The logic function of each molecule is defined by its molecular code or `MOLCODE`.
- A finite set of molecules makes up a *cell*, essentially a processor with the associated memory. In a first programming step of the FPGA, the polymerase genome `PG` defines the topology of the cell, that is, its width, height, and the presence and positions of columns of spare molecules. In a second step, the ribosomic genome `RG` defines the logic function of each molecule by assigning its molecular code or `MOLCODE`.
- A finite set of cells makes up an *organism*, an application-specific multiprocessor system. In a third and last programming step, the operative genome `OG` is copied into the memory of each cell to define the particular application executed by the organism (electronic watch, random number generator, and a Turing machine being examples shown by us to date)
- The organism can itself self-replicate, giving rise to a *population* of identical organisms, the highest level of our hierarchy.

III. A Cellular Implementation and its Applications

In the Embryonics Project, each cell of our artificial organism will be implemented by a small processor executing sequentially the instructions of the operative genome OG. While our final objective is the conception of a new FPGA (the molecular FPGA described in Subsection II.D), which will allow us to adapt the architecture of our cells to a given problem, we decided to test our approach on a demonstration system in which each artificial cell is characterized by a *fixed architecture*. Subsection A describes in detail our artificial cell, while in Subsections B, C, and D we show that multicellular arrays can realize a variety of very different organisms (electronic watch, random number generator, a Turing machine), all capable of self-replication and self-repair. Finally, Subsection E deals with the range of applications for our cell and its limitations.

A. A Cell Based on a Binary Decision Machine

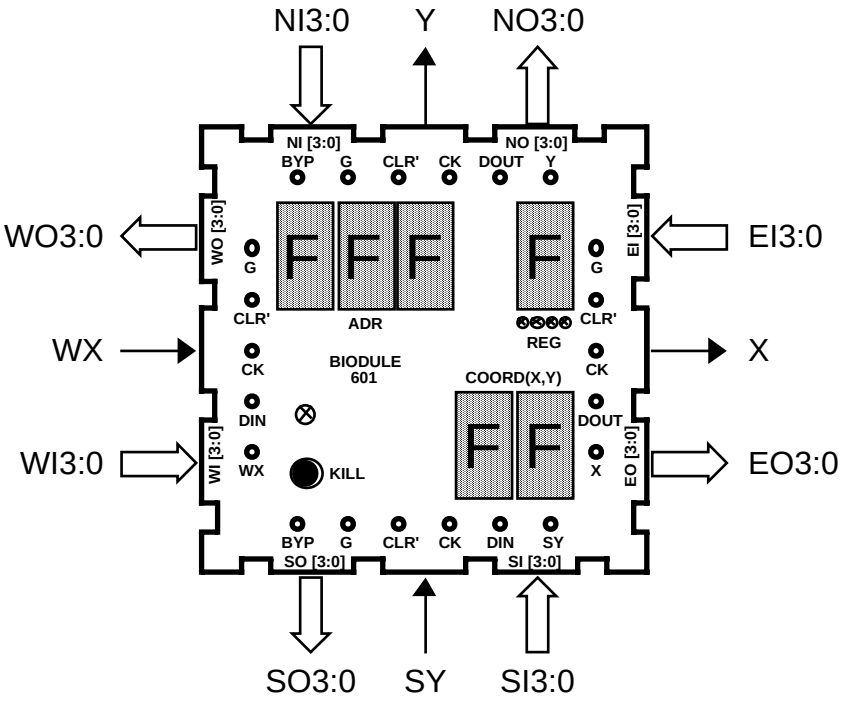
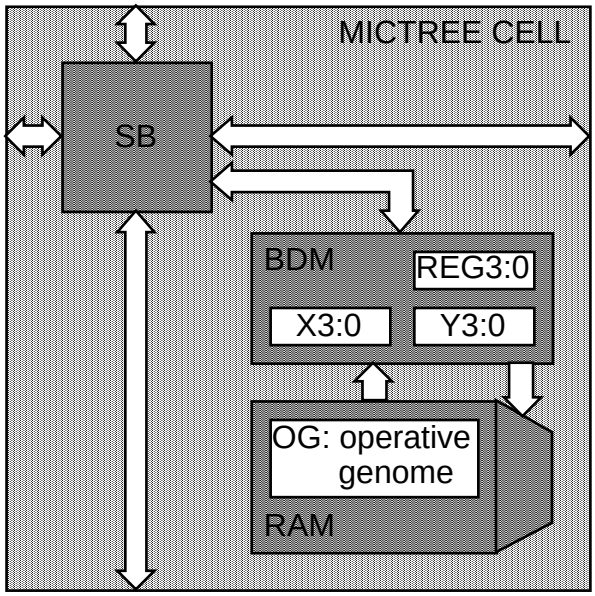


Fig. 3.1a

Our artificial cell, called *MICTREE* (for *microinstruction tree*) is basically a *binary decision machine* [3], [4], [5] implemented using standard electronic components and embedded into a plastic container (Fig. 3.1a). These containers can easily be joined to obtain two-dimensional arrays as large as desired (Fig. 3.4b).



(b)

Fig. 3.1 The MICTREE artificial cell. (a) Front panel of a demonstration module implementing the cell. (b) Block diagram; SB: switch block; BDM: binary decision machine; RAM: random access memory.

The MICTREE cell sequentially executes microprograms written using the following set of instructions (Fig. 3.1b):

- **do REG =** DATA
- **do X =** DATA
- **do Y =** DATA
- **do** VAROUT **=** VARIN
- **if** VAR **else** LABEL
- **goto** LABEL

The state register REG and both coordinate registers X and Y are 4-bit wide (REG3:0, X3:0, and Y3:0). The variable VAROUT designates any one of the 4-bit wide output busses in the four cardinal directions (VAROUT∈{SO3:0,WO3:0,NO3:0,EO3:0}), while the variables VARIN designates any one of the 4-bit wide input busses or the state register REG (VARIN∈{SI3:0,WI3:0,NI3:0,EI3:0,REG3:0}). Thus, the instruction "**do** VAROUT **=** VARIN" allows the value of any of the input busses or of the state register REG to be sent to any of the cell's four cardinal neighbors. The test variables VAR include the set VARIN and the following additional variables: WX3:0 (the X coordinate sent by the cell's western neighbor), SY3:0 (the Y coordinates sent

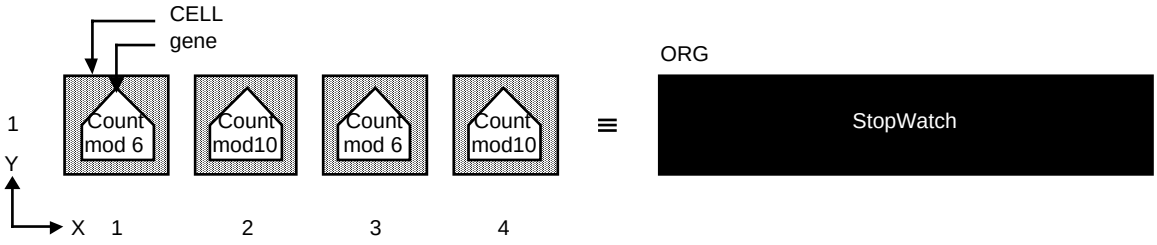
by the cells' southern neighbor), and G (a global variable, usually reserved for the synchronization clock).

The coordinates are transmitted from cell to cell serially, but are computed in parallel within the cell. Therefore, each cell performs a series-to-parallel conversion on the incoming coordinates WX and SY, and a parallel-to-series conversion of the coordinates X and Y it computes and propagates to the east and north. The genome microprogram is also coded serially: it enters the cells through the DIN pin(s) and is then propagated through the DOUT pin(s). The pins CK and CLR' are used, respectively, for the propagation of the clock signal and to reset the binary decision machines, while the signal BYP (bypass), connecting all the cells in a column, is used for self-repair. In a MICTREE cell, pressing the KILL button alerts a cell to the presence of a fault. The KILL button therefore replaces, in the demonstration system, the molecular fault detection mechanism mentioned in Section II. The effect of the KILL button is to deactivate the column containing the faulty cell: all the cells in the column "disappear" from the array, that is, become transparent with respect to all horizontal signals. Since the computation of the coordinates occurs locally depending on the neighbors' coordinates, such disappearance automatically engenders a recomputation of the coordinates of all the cells to right of the deactivated column, completing the reconfiguration of the array.

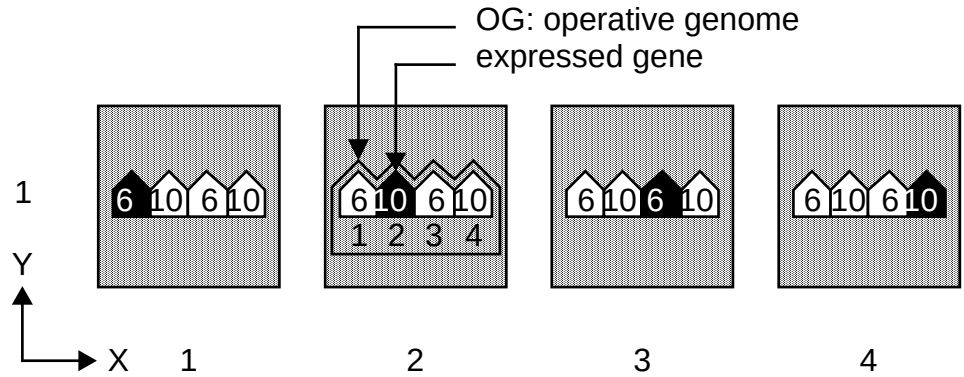
The size of the artificial organism embedded into an array of MICTREE cells is limited in the first place by the coordinate space (X=0…15, Y=0…15, that is, a maximum of 256 cells in our current implementation, a limit imposed by the size of the coordinate registers) and then by the size of the memory of the binary decision machine storing the genome microprogram (1024 instructions). An editor, a compiler for the assembly language, and a loader simplify the task of writing and debugging the microprograms and generating the genome's binary code, charged serially through the DIN input of the mother cell.

B. The StopWatch and the BioWatch

For clarity's sake, we will begin with a simple example of a one-dimensional artificial organism: a *timer*, called *StopWatch*, designed to count seconds (from 00 to 59) and minutes (from 00 to 59) [44][45]. This organism is implemented with four cells (Fig. 3.2a) and is characterized by two distinct genes: "`Countmod10`", which counts modulo 10 the units of seconds or minutes, and "`Countmod6`", which counts modulo 6 tens of seconds or tens of minutes. The conception of these genes using the instructions of Subsection A is described in detail elsewhere [4].



(a)



(b)

```
OG: operative genome
X = (WX+1) mod 4
case of X:
  X = 1: Countmod 6  (10 minutes)
  X = 2: Countmod 10 (minutes)
  X = 3: Countmod 6  (10 seconds)
  X = 4: Countmod 10 (seconds)
```

(c)

Fig. 3.2 StopWatch, a digital timer. (a) Multicellular organization. (b) Cellular differentiation. (c) The operative genome `OG`.

Fig. 3.2a shows the operative genome `OG` of StopWatch (i.e., the set of all the genes with the corresponding `X` coordinate). By storing in each cell the entire operative genome `OG`, we implement cellular differentiation (Fig. 3.2b). In order to verify the

property of self-replication of the organism (Fig. 3.3a), the computation of the X coordinate occurs modulo 4. The final program, the operative genome OG, is shown in Fig. 3.2c.

The self-replication of StopWatch can be accomplished if:

1. there exists a sufficient number of spare cells (four cells to the right of the original organism in Fig. 3.3a);
2. the calculation of the X coordinate produces a cycle (X=1→2→3→4→1…).

If both these conditions are satisfied, the *mother organism* produces an exact copy of itself, the *daughter organism*.

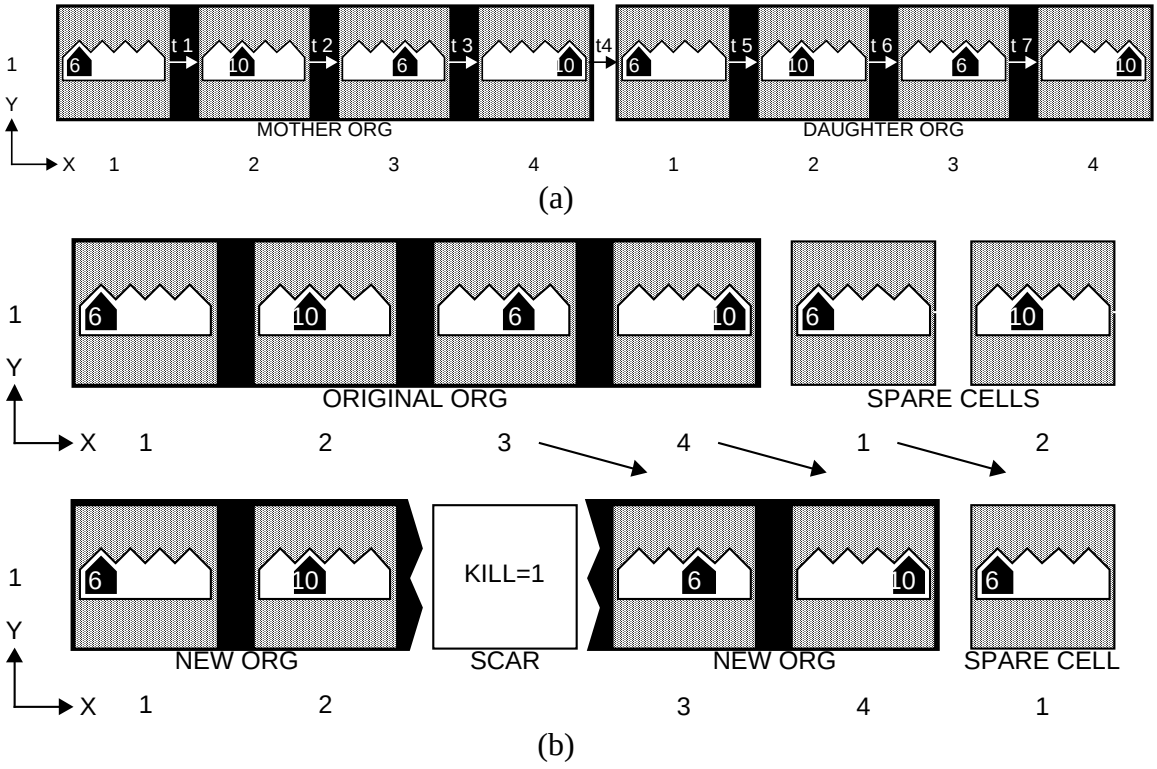
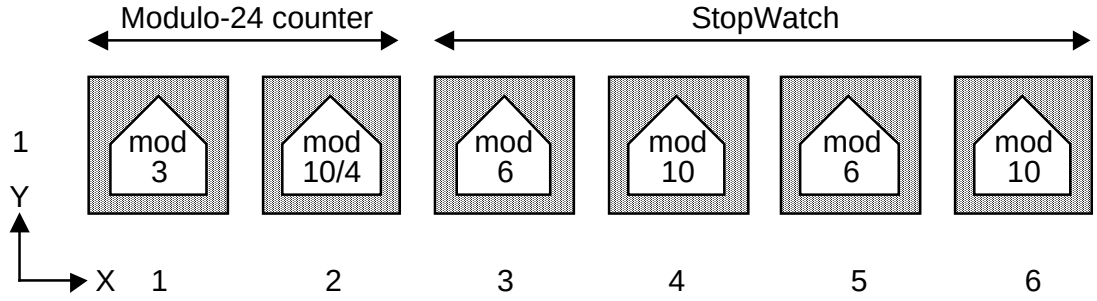


Fig. 3.3 The 4-cell StopWatch organism. (a) Self-replication in an array of cells (t1 … t7: seven cellular divisions). (b) Self-repair with two spare cells and one faulty cell.

To demonstrate the self-repair of StopWatch, we will use the four cells to the right of the original organism (Fig. 3.3b) as spare cells. Once a faulty cell has been identified (state KILL=1 in cell X=3 for the original organism of Fig. 3.3b), all the functions (X coordinate and gene) of the cells on the right of column X=2 are shifted by one column to the right. In the one-dimensional example of StopWatch, the presence of four spare cells allows the organism to tolerate four successive faulty cells.

By adding a modulo-24 counter for counting the hours (from 00 to 23) to our modulo-3600 counter (the StopWatch), we can easily realize a *full digital watch*, called *BioWatch*, (Fig. 3.4) [4]. The modulo-24 counter is the composition of two partial counters, one for the units of hours, the other for the tens of hours. The final genome of the full digital watch thus consists of four distinct genes, distributed among six cells identified by the horizontal coordinates `X=1` to `X=6`.



(a)

(b)

Fig. 3.4 BioWatch, a full digital watch. (a) Multicellular organization (`X=1`: tens of hours; `X=2`: units of hours). (b) Experiment with 8 MICTREE cells and 2 spare cells.

With a greater number of MICTREE cells, it would be easy to introduce additional features to our electronic watch, that is, functions other than the counting of seconds, minutes, and hours. For example, computing the date, keeping track of the day of the week, or handling leap years. In any case, the genomic design of the BioWatch

guarantees extreme flexibility through genome reprogramming, as well as considerable reliability, thanks to the self-repair and self-replication properties.

C. A Random Number Generator

Wolfram [6] exhaustively studied *uniform one-dimensional cellular automata* consisting of identical cells defined by a binary state and a neighborhood with a connectivity radius equal to 1 (i.e., the future state of a given cell depends on the present state `Q` of the cell itself and on that of its immediate neighbors to the west `QW` and to the east `QE`). Such a cell is defined by a truth table of 2^3=8 lines, and there exist 2^8=256 such truth tables. Each of these functions is called the *rule* of the automaton and is identified by a decimal number between 0 and 255. Hortensius et al. [7] have shown that a well-chosen arrangement of Wolfram cells of type 90 and 150 produces a *non-uniform cellular automaton* which is, in fact, a *random number generator*. For a 5-cell automaton, the final arrangement is shown in Fig. 3.5. In such an arrangement, the *periodic condition* (i.e., the values of the inputs of the leftmost and rightmost cells) is equal to `0`. The global state `00000` is a fixed point of the generator, while the remaining 2^5-1=31 states form a cycle of maximum length.

The properties of self-replication and self-repair of the random number generator were demonstrated and have been described elsewhere [5].

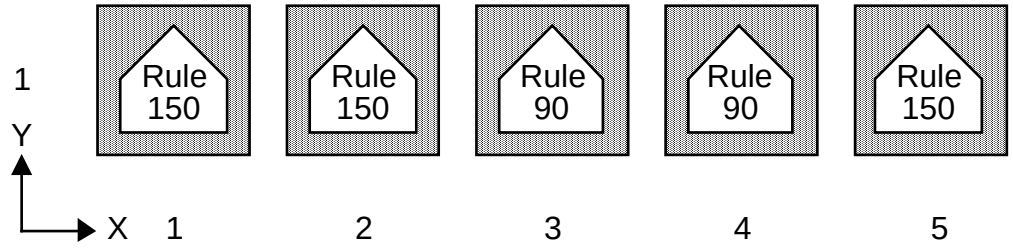


Fig. 3.5a

```
OG: operative genome
X = (WX+1) mod 5
case of X:
  X = 1: rule 150  (Q+ = QW⊕Q⊕QE)
  X = 2: rule 150
  X = 3: rule  90  (Q+ = QW⊕QE)
  X = 4: rule  90
  X = 5: rule 150
```

(b)

Fig. 3.5 A random number generator. (a) Multicellular organization.

(b) The operative genome `OG`.

D. A Specialized Turing Machine

With a more theoretical goal in mind, in order to compare the capabilities of the Embryonics project with those of von Neumann's self-replicating automaton [8], we wanted to show that an artificial multicellular organism can implement a specialized Turing machine and exhibit the properties of self-replication and self-repair [9].

The example we settled upon is that of a *parenthesis checker*, as described by Minsky [10]. The function of the machine is to decide whether a sequence of left (open) and right (closed) parentheses is well-formed (i.e., if for every open parenthesis in the sequence there exists a corresponding closed parenthesis). A specialized Turing machine for checking parentheses consists of a *tape*, decomposed into squares, and a *finite state machine* with a *read/write head*.

For the very simple example of Fig. 3.6, which checks the sequence A( )A (where A is a symbol delimiting the sequence of parentheses), we can implement the Turing machine as a multicellular organism with the following structure:

- For Y=1, we use five cells to display the sequence A( )A. These five cells correspond to the tape of the Turing machine.
- For Y=2, we use five cells to implement the read head of the Turing machine. In fact, the head is realized by only one of the cells (X=2 in Fig. 3.6), the others being inactive. The head is mobile: depending on its current state (→, ←, or ↑), it can move, in the next clock cycle, to the right, to the left, or stay in place.

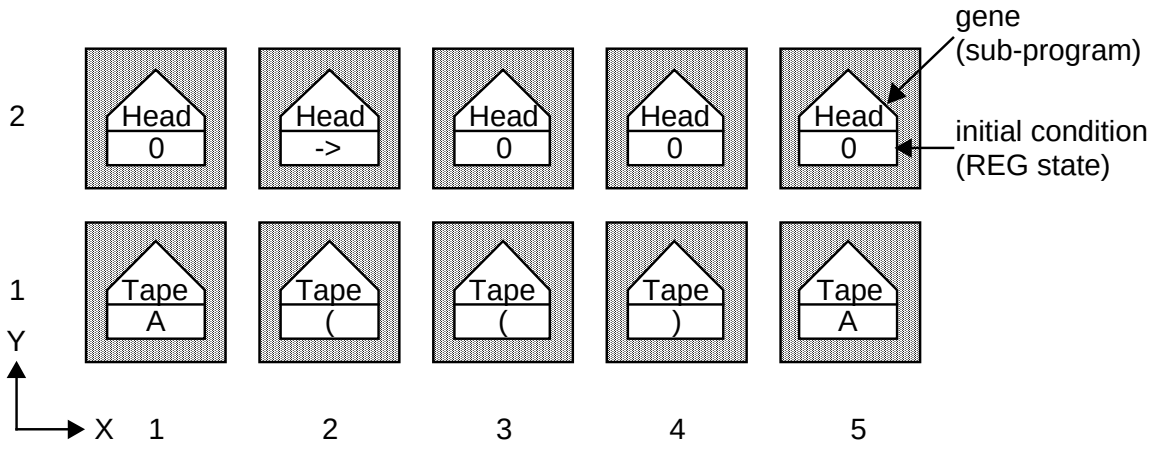


Fig. 3.6 Multicellular organization of a specialized Turing machine, a parenthesis checker.

There exist, finally, only two distinct genes: the *Head* gene, which implements the algorithm for checking the parentheses and is executed by the cells of row Y=2, and the *Tape* gene, which describes the state of the tape and is executed by the cells of

row Y=1. In addition, each cell is defined by an *initial condition*, that is, a specific value of the register REG at the beginning of the algorithm (REG∈{A,(,)} for Y=1, REG∈{0, →} for Y=2 in Fig. 3.6).

The properties of self-replication and self-repair of this application are demonstrated elsewhere [4], [9].

E. Final Remarks

The main focus of this Section was the description of an artificial cell, called MICTREE, based on a binary decision machine capable of executing a microprogram of up to 1024 instructions. Any organism realized using MICTREE cells satisfies the three features of the Embryonics project (Subsection II.B): multicellular organization, cellular differentiation, and cellular division. The MICTREE cell, itself realized with a commercial FPGA and a RAM, was finally embedded into a demonstration module, and we showed that an array of these modules exhibits the two desired properties of self-repair and self-replication.

The trivial applications of the MICTREE family are those in which all the cells in the array contain the same gene: the genome and the gene then become indistinguishable and the calculation of the coordinates is superfluous. In this case, the cellular array is not limited in space. One-dimensional (e.g., Wolfram's) [6] and two-dimensional (e.g., Conway's Life) [11] uniform cellular automata are natural candidates for this kind of realization. The non-trivial applications are those in which the cells in an array have different genes: the genome is then a collections of genes, and the coordinates become necessary. The cellular array is then limited by the coordinate space (16x16=256 cells in the proposed realization). One-dimensional (like the examples of the StopWatch, BioWatch, and the random number generator) and two-dimensional (specialized Turing machine) cellular automata fall into this category. Let us also mention that the realization of uniform cellular automata with the automatic calculation of an initial condition (realized by setting the internal register REG to a pre-determined value in each cell of the organism at the start) is an important special case which also requires separate genes and a coordinate system.

IV. A Molecular Implementation and its Applications

In Section III, we introduced the implementation of an artificial cell, called MICTREE. Its architecture is *fixed*, and it is thus easy to find an application whose requirements exceed the capabilities of the MICTREE cell: a number of instructions greater then 1024, horizontal or vertical coordinates superior to 16, or a register of more than 4 bits are all demands which would require a redesign of the original cell. To meet the requirements of all possible applications, we want to develop an artificial cell endowed with a *flexible architecture*, that is, an architecture which is itself configurable. This architecture will be realized using a novel fine-grained field-programmable gate array (FPGA).

A consequence of our choices is that we require a methodology to generate, starting from a set of specifications, the configuration of our FPGA, consisting of a homogeneous network of elements, the *molecules*, defined by an identical architecture and a usually distinct state (the *molecular code*, or `MOLCODE`).

To fulfill this requirement, we have selected a particular representation: the *ordered binary decision diagram* (OBDD) [12], [13], [14]. This representation, with its well-known intrinsic properties such as canonicity, was chosen for two main reasons:

- it is a graphical representation which exploits well the two-dimensional space and immediately suggests a physical realization on silicon;
- its structure leads to a natural decomposition into molecules realizing a logic test, easily implemented by a multiplexer.

Our choice led us to define our FPGA as a homogeneous multimolecular array where each molecule contains a programmable multiplexer with one control variable, implementing precisely a logic test. The three main features of this FPGA, introduced in Subsection II.D (Fig. 2.10), are the following:

- *Multimolecular organization* divides the cell into an array of physically identical elements, the molecules. The configuration string of all the molecules of a cell is equivalent to the ribosomic genome `RG`.
- *Molecular configuration* determines the physical position of each molecule in the cellular space according to the information contained in the polymerase genome `PG`.

- *Molecular fault detection* detects and localizes faults occurring at the molecular level.

The plan of this Section is the consequence of this structure. Subsection A describes the core of our molecule (the programmable multiplexer and its short- and long-distance connections) and defines the molecular code `MOLCODE`, the building block of the ribosomic genome `RG`. Subsection B introduces the *space divider*, a state machine which allows multiple molecules to be grouped in order to form a cell, and defines the polymerase genome `PG`. Subsection C ends the description of the molecule with the introduction of the automatic fault detection system required for self-repair. Subsection D presents the implementation of a prototype of our molecule, while Subsections E and F present two applications of widely different complexity (a counter and a binary decision machine). Finally, Subsection F deals with the range of applications for our molecule and its limitations.

A. A Molecule Based on a Multiplexer

The main features of our artificial molecule, henceforth referred to as *MUXTREE* (for *multiplexer tree*) [4], [15], [16], are the following (Fig. 4.1):

- Each of the two inputs of the multiplexer `MUX` (inputs `0` and `1`) are programmable. The input is either a logic constant (`0` or `1`), the output of one of the neighboring molecules to the south (`SIN`), southeast (`EIN`), or southwest (`WIN`), the output of the molecule's flip-flop (`Q`), or one of the vertical long-distance connection busses `SIBUS` or `SOBUS`.
- The output of the molecule (`NOUT`) is, as a consequence, directly connected to the inputs of the multiplexers of the neighboring molecules to the north, northeast, and northwest.
- The implementation of sequential systems requires the presence, in each molecule, of a synchronous memory element, a D-type flip-flop (`FF`).
- Long-distance connections are needed to connect a molecule to any other molecule in the array. The switch block `SB` (Fig. 4.2) allows any connection between the horizontal and vertical busses.

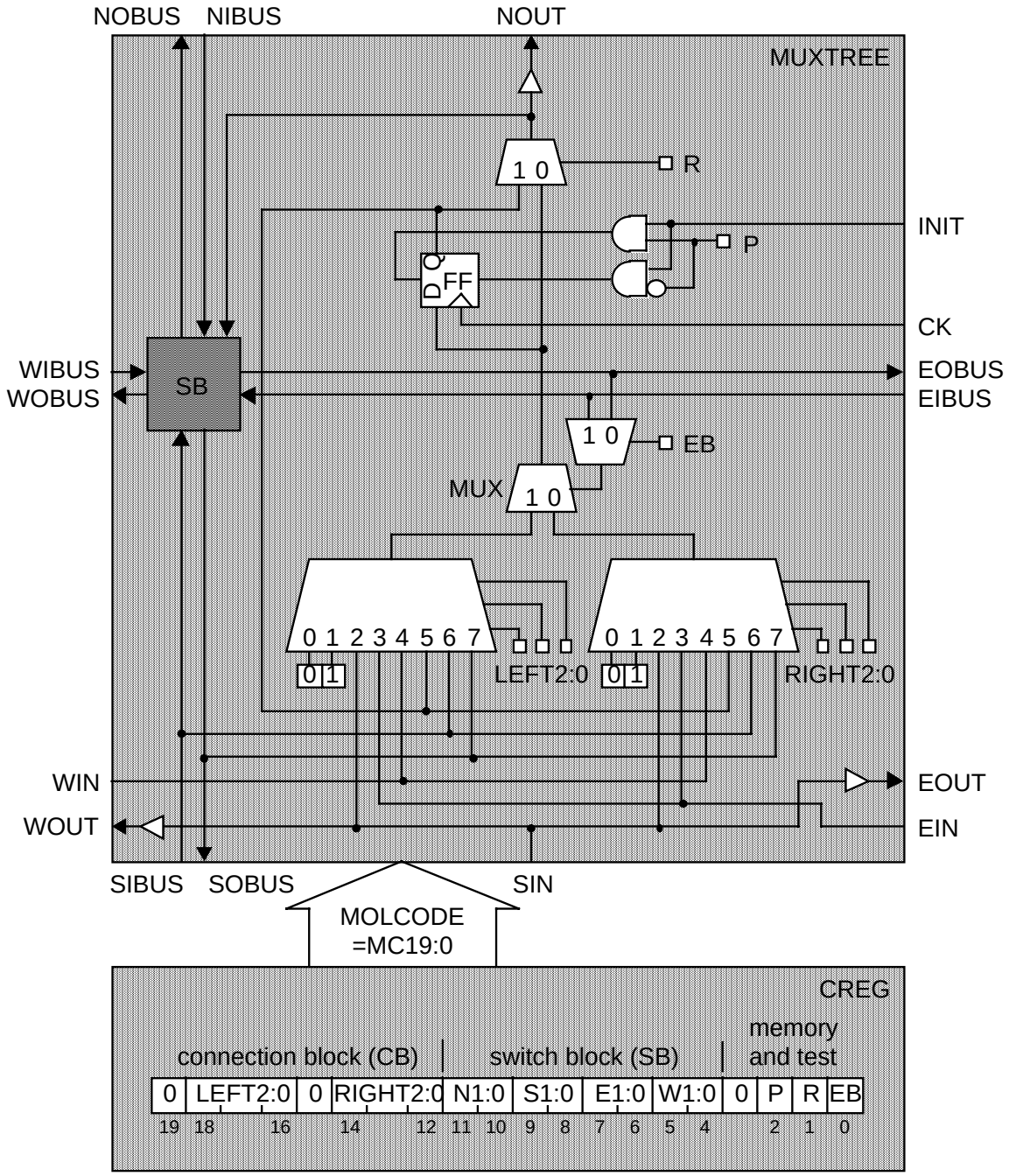


Fig. 4.1 Logic layout of a MUXTREE molecule, including
the configuration register CREG and the switch block SB.

In brief, the core of the molecule remains the one-variable multiplexer, optionally followed by a flip-flop. Inputs and outputs are programmable and can be connected either to the immediate neighbors according to a topology suitable for binary decision diagrams (where information flows bottom-up), or to faraway molecules through a network of perfectly symmetric busses.

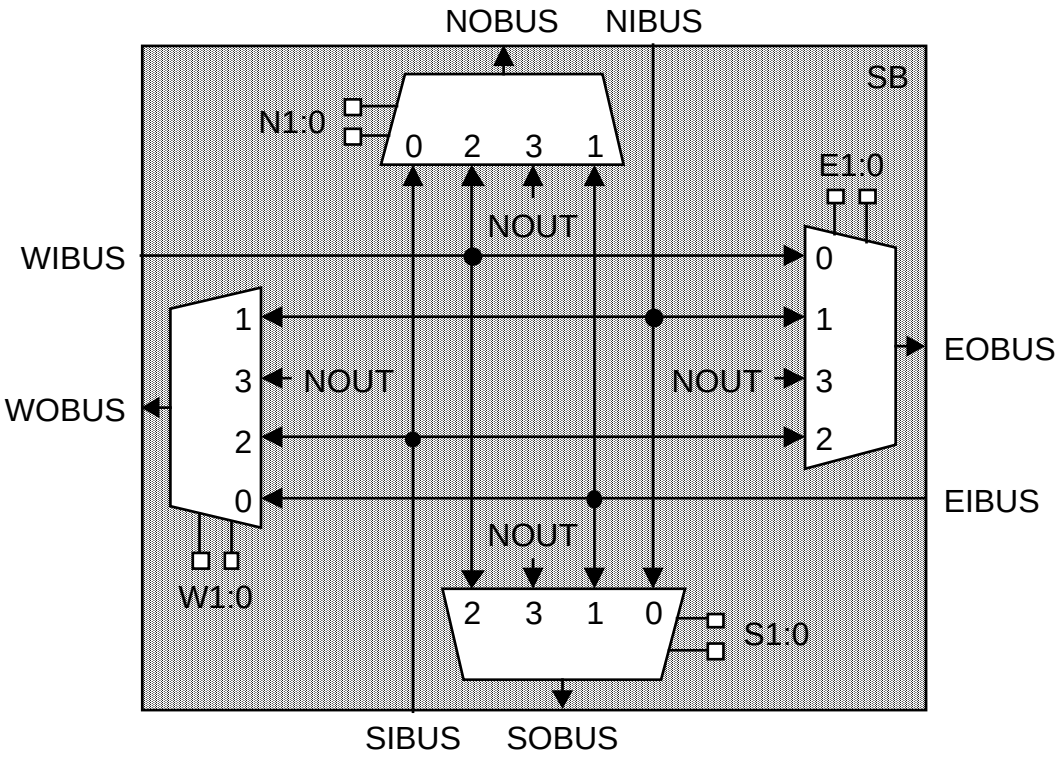


Fig. 4.2 Detailed architecture of the switch block SB.

All the information necessary for programming the MUXTREE molecule, that is, the 17 field-programmable bits which make up the molecular code MOLCODE, is organized as a 20-bit word (MC19:0) so as to simplify its hexadecimal representation, and is stored in the *configuration register* CREG (Fig. 4.1). From right to left we have:

- EB (MC0) selects EIBUS or EOBUS as the control variable for the MUX multiplexer;
- R (MC1) selects the output of the multiplexer (combinational) or the output of the flip-flop (sequential) as the output NOUT of the molecule;
- P (MC2) allows the synchronous set or reset of the flip-flop;
- the SB bits (MC11:4) define the connections of the long-distance busses, as shown in Fig. 4.2;
- the CB bits (MC18:12) define the inputs of the multiplexer MUX, as shown in Fig. 4.1.

B. A Molecule with a Space Divider

The information contained in the `MOLCODE` defines the logic function of each molecule. To obtain a functional cell, i.e., an assembly of MUXTREE molecules, we require two additional pieces of information, defining the physical position of each molecule within a cell and the presence and position of the spare columns required by the self-repair mechanism (Subsection C).

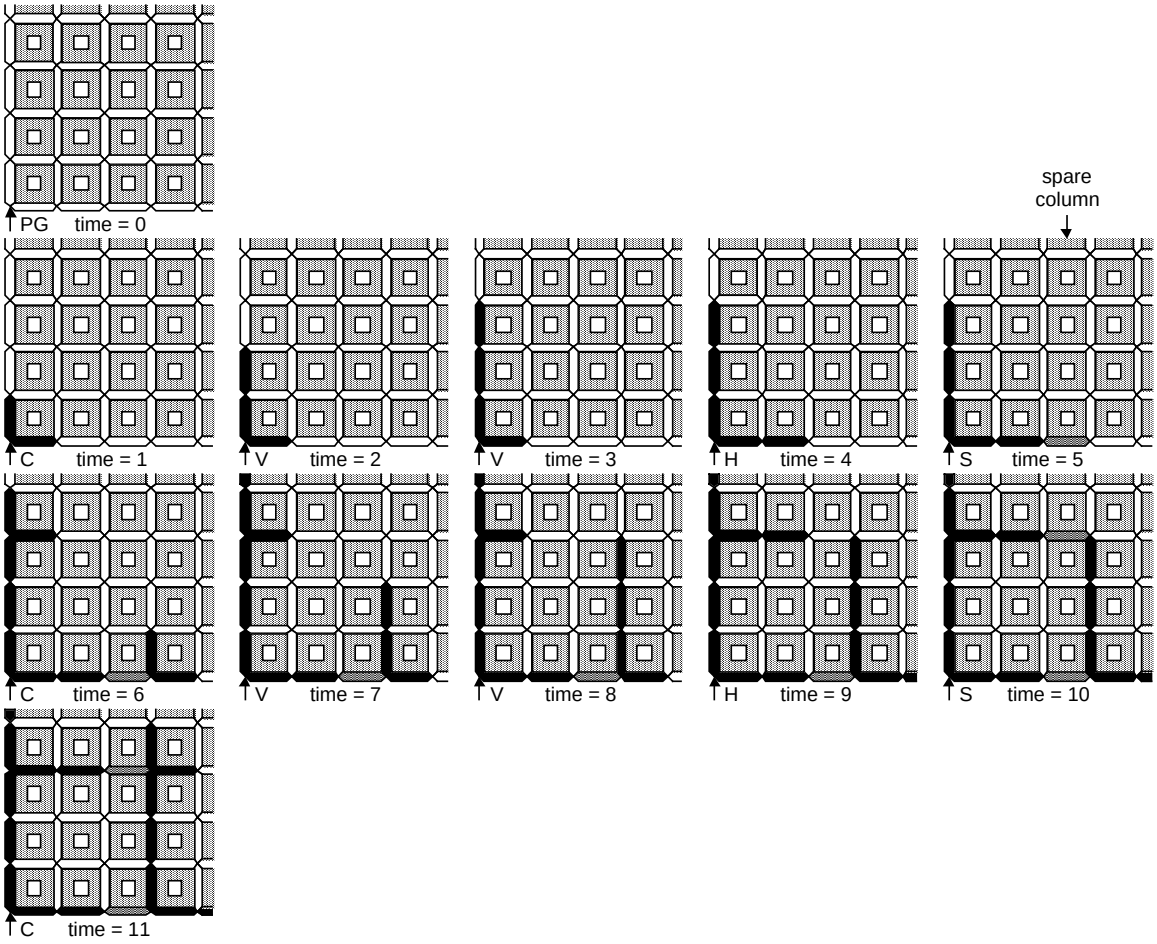


Fig. 4.3 Example of a space divider (height=3, width=3, 1 spare column out of 3); `PG`: polymerase genome: `C,V,V,H,S,C,…`

The mechanism which we have adopted consists of introducing in the FPGA a regular network of automata (state machines) called *space divider* [4], [16], [17]. Each vertical or horizontal band of the example of Fig. 4.3 is an instance of this automaton. Using the space divider, it is thus possible to divide the entire space of the FPGA into cells of identical size and to specify the position of the spare columns. Fig. 4.3 shows an FPGA divided into cells of height 3 and width 3, with one out of every three

columns being spare. The polymerase genome PG can be inferred from Fig. 4.3 and consists of a cycle of the following states:

PG = C,V,V,H,S,C,…

where C represents a corner, V a vertical band, H an horizontal band, and S an horizontal band associated with a spare column.

More generally, if we use the notation {X}*[n] to represent the state (or the sequence of states) X repeated n times, a cell of height h and width w will be defined by the following polymerase genome:

PG = C,{V}*[h-1],{H}*[w-1],C,…

where the presence of spare columns will be indicated by replacing one or more occurrences of H by S.

The details of the design of the space divider are described elsewhere [4].

C. A Molecule with Fault Detection

The specifications of the molecular self-repair system must include the following features:

- it must operate in real time;
- it must preserve the memorized values, that is, the state of the D-type flip-flop contained in each molecule;
- it must assure the automatic detection of a fault (self-test), its localization, and its repair (self-repair) at the molecular level;
- it must involve an acceptable overhead;
- finally, in case of multiple faults (too many faulty molecules), it must generate a global signal KILL=1 which activates the suppression of the cell and starts the self-repair process of the complete organism (Subsection II.C).

The need to meet all these specifications forced us to adopt a set of compromises with regard to the fault detection capabilities of the system. A self-repairing MUXTREE molecule can be divided into three parts (Fig. 4.4) [4], [17], [18]:

- The functional part of the molecule (the multiplexers and the internal flip-flop) is tested through space redundancy: the logic is duplicated (M1 and M2) and the outputs of the two copies compared to detect a fault. A third copy of the flip-flop (FF3) was added to allow self-repair (i.e., to recover the state of the flip-flop).
- The configuration register (CREG) is tested every time the configuration is entered (and thus on the field but not on-line). Being implemented as a shift register, it can

be tested using a dedicated test sequence introduced in all the elements in parallel before the actual configuration of the FPGA.

- Faults on the connections (and in the switch block SB) can be detected, but cannot be repaired, both because they cannot be localized to a particular connection and because our self-repair system relies on these connections to reconfigure the array. In the current implementation, therefore, we decided not to test the connections directly, a limitation which is in accordance with the current state of the art [19]. In a future version of our system, it will be possible to test and repair the connections using a double rail architecture [20].

The hardware overhead (in terms of silicon area) required to implement all of the above features (including both self-test and self-repair) in the current version of the MUXTREE molecules is estimated to approximately 40% of the original area.

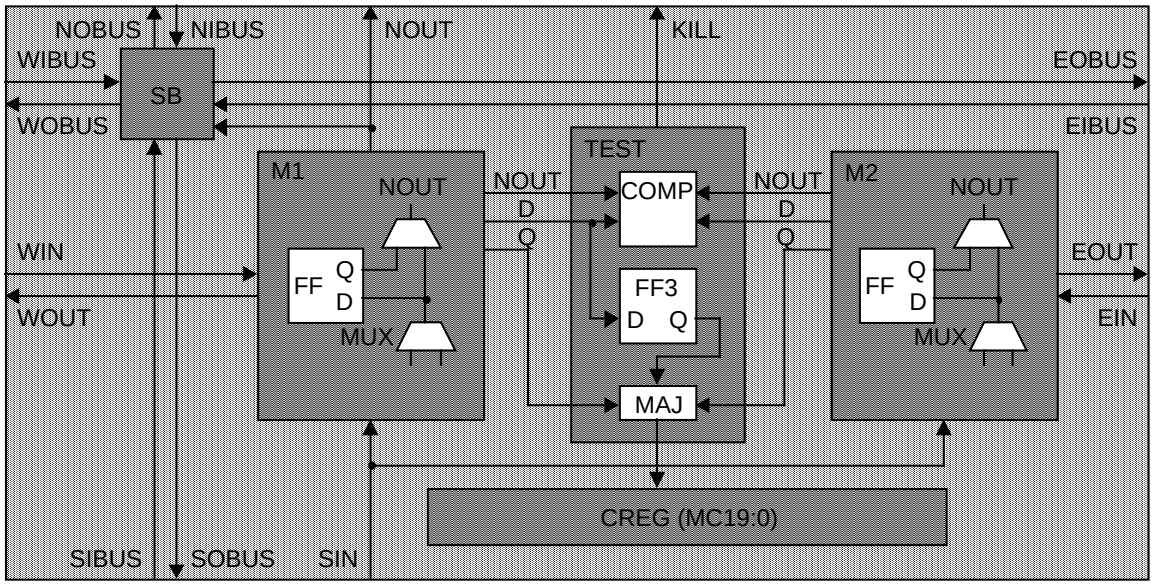


Fig. 4.4 A self-testing MUXTREE molecule using space redundancy.

To meet the specifications, and in particular the requirement that the hardware overhead be minimized, our self-repair system exploits the programmable frequency and distribution of the spare columns (Subsection B) by limiting the reconfiguration of the array to a single molecule per line between two spare columns (Fig. 4.5). This choice allows us to minimize the amount of logic required for the reconfiguration of the array, while keeping a more than acceptable level of robustness. This mechanism is also in accordance with the current state of the art [20].

It should be added that, should the self-repair capabilities of the MUXTREE molecular level be exceeded, a global `KILL` signal is generated and the system will attempt to reconfigure at the higher (cellular) level through the process described in Subsection II.C.

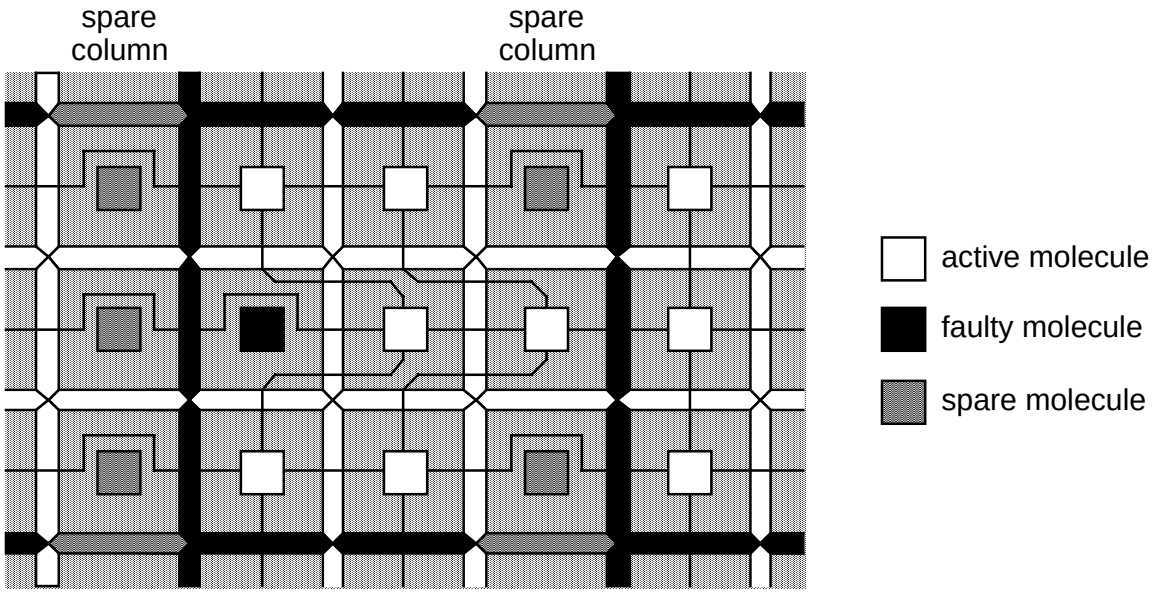
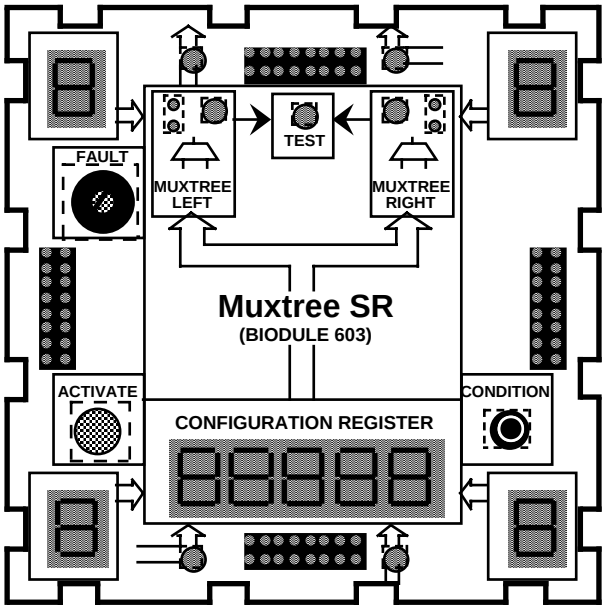


Fig. 4.5 The self-repair mechanism for an array of MUXTREE molecules.

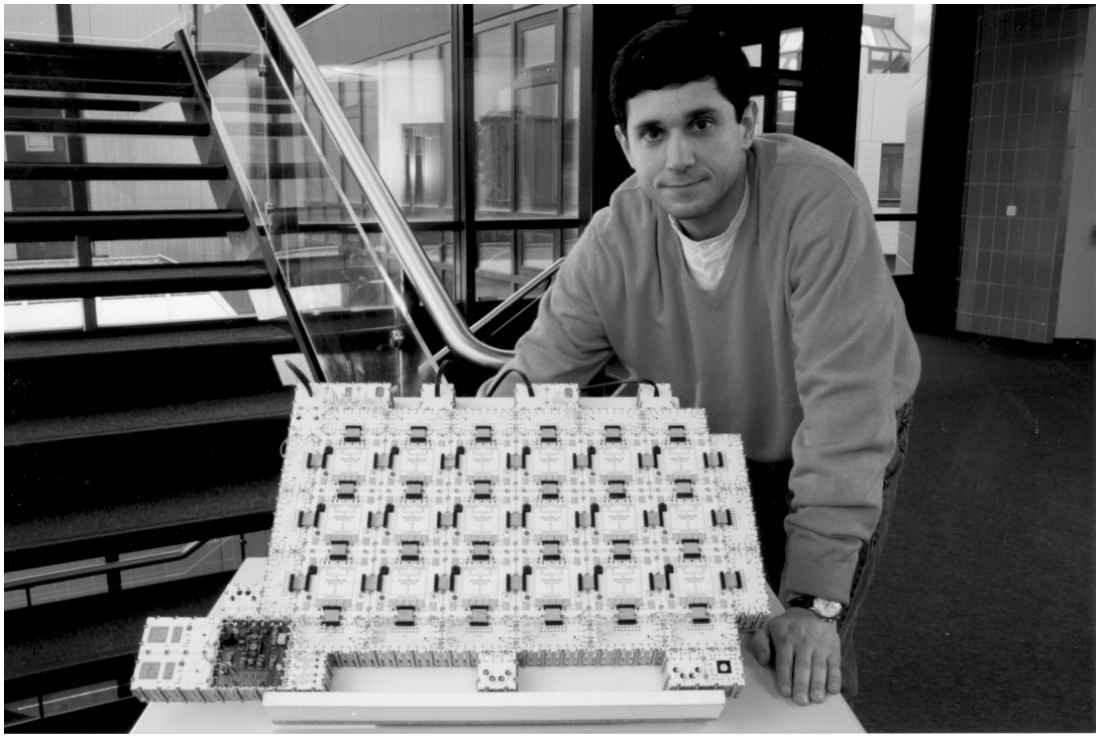
D. A Molecule's Implementation

While our long-term objective is the conception of very large scale integrated circuits, we started by realizing a demonstration system in which each MUXTREE molecule is embedded into a plastic container (Fig.4.6a) [4], [16]. These containers can easily be joined to obtain two-dimensional arrays as large as desired (Fig. 4.6b).

The MUXTREE molecule is itself realized using a reprogrammable off-the-shelf FPGA and is configured to implement the following subsystems:



(a)



(b)

Fig. 4.6 The MUXTREE molecule. (a) Front panel of a demonstration module implementing the molecule. (b) An array of MUXTREE molecules.

- The molecule itself (Figs. 4.1, 4.2, and 4.4), including the 20-bit configuration register `CREG`, the switch block `SB` for long distance connections, and the two copies (`M1` and `M2`) of the functional part of the element used for self-test, whose outputs are compared (`COMP`) to determine if a molecule is faulty.

- Four copies of the automaton used as a space divider (Fig. 4.3). The four copies are required to allow each module to work independently of the presence of neighbors.
- The logic required to inject a fault in the circuit, including an activation circuit (a 4-bit manual encoder, used to select one out of 16 possible faults, and a pushbutton to activate the fault) and the gates required to force specific lines to a given value, thus simulating the presence of stuck-at faults.
- A set of 7-segment displays (with the associated decoders) and light-emitting diodes used to display the state of the circuit.

E. A Modulo-4 Up-Down Counter

For clarity's sake, we will start with a simple example of artificial organism, a single cell (Fig. 4.7) realizing a modulo-4 up-down counter defined by the following sequences:

- for `M=0`: `Q1,Q0=00`→`01`→`10`→`11`→`00`→… (counting up);
- for `M=1`: `Q1,Q0=00`→`11`→`10`→`01`→`00`→… (counting down).

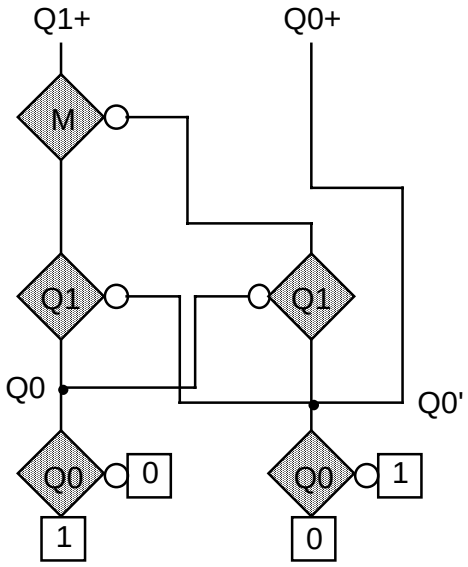
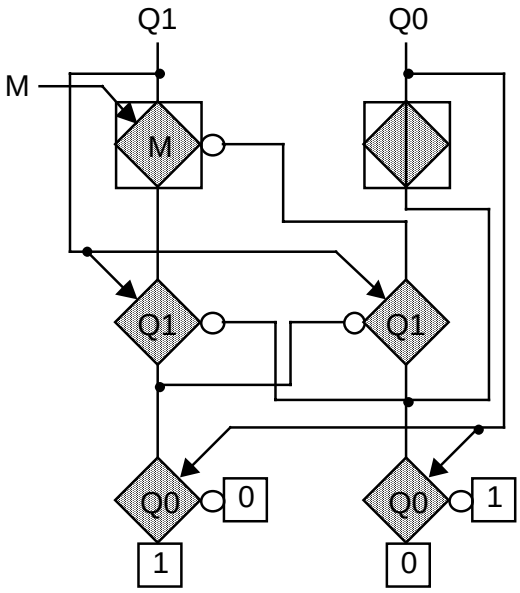


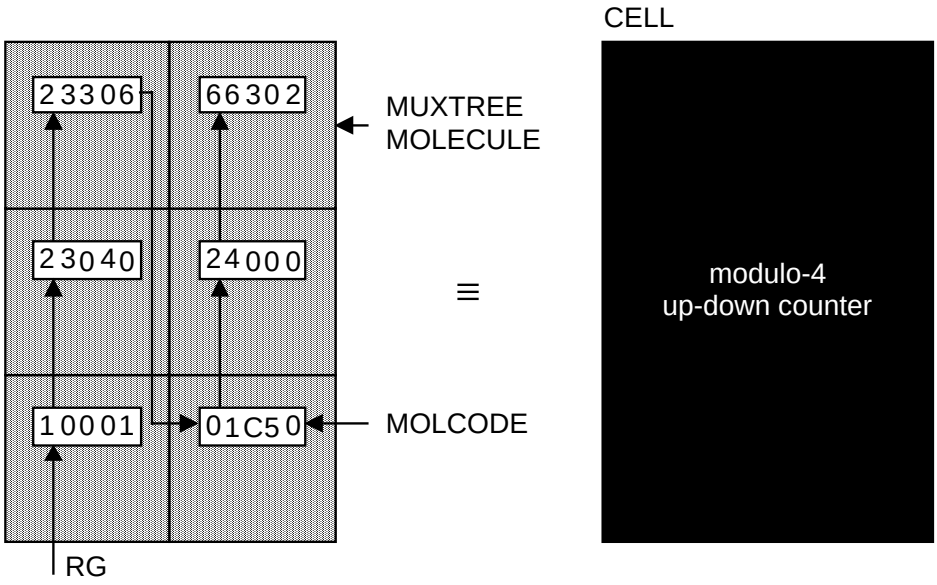
Fig. 4.7a

It can be verified that the two ordered binary decision diagrams `Q1+` and `Q0+` of Fig. 4.7a (where each test element is represented by a diamond with a single input, a "true" output, and a "complemented" output identified by a small circle) represent a possible realization of the counter [3], [4]. The leaf elements, represented as squares, define the output values of the given functions (`Q1+` and `Q0+` in the example) computed with the following equations:

```
Q1+ = M (Q1·Q0 + Q1'·Q0') + M'(Q1·Q0' + Q1'·Q0)
Q0+ = Q0'
```



(b)



(c)

Fig. 4.7 Modulo-4 up-down counter. (a) Ordered binary decision diagrams for `Q1+` and `Q0+`. (b) Multiplexer diagram using MUXTREE molecules. (c) 6 MUXTREE molecule cell; `RG`: ribosomic genome.

For our design, we decided to implement directly the ordered binary decision diagrams on silicon, and to build our fine-grained basic molecule (MUXTREE) around a test element (a diamond). Such a layout can be realized (Fig. 4.7b) by implementing each test element with a one-variable multiplexer (the MUXTREE molecule), keeping the same interconnection diagram, and assigning the values of the leaf elements to the appropriate multiplexer inputs. The two state functions `Q1` and `Q0` are the outputs of the D flip-flops of the top row of MUXTREE molecules (diamonds embedded in a square in Fig. 4.7b) and, carried by the long-distance horizontal and vertical busses, become the control variables for the multiplexers of the bottom two rows.

The counter can be thus be implemented by an array of 3 rows by 2 columns, that is, by a cell made up of 6 MUXTREE molecules. From the multiplexer diagram of Fig. 4.7b and from the description of the MUXTREE molecule (Figs. 4.1 and 4.2) we can then compute the 17 control bits of each molecular code, finally generating the `MOLCODEs` of Fig. 4.7c. The ribosomic genome `RG` is, ultimately, the string of the `MOLCODEs` of our artificial cell, each `MOLCODE` being a word of five hexadecimal digits (Fig. 4.7c).

The manual computation of the molecular code can be very awkward. Thus, in order to automate this part of the development, we have developed a graphical tool, the MUXTREE editor [4].

Thanks to the conception of the new family of field-programmable gate arrays MUXTREE, we are therefore able to realize any given logic system, combinational or sequential, using a completely homogeneous multimolecular network. This realization is simplified by the direct mapping of the ordered binary decision diagrams onto the array.

F. A Shift Binary Decision Machine

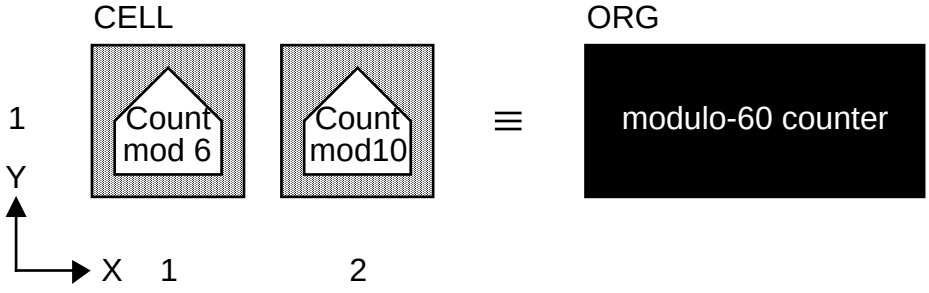


Fig. 4.8 A modulo-60 counter made up of two cells.

In the preceding Subsection, we have shown that an assembly of six MUXTREE molecules was sufficient to realize a very simple *unicellular artificial organism*: a modulo-4 up-down counter. Yet our final goal is the development of truly *multicellular organisms*, in which each cell is a binary decision machine similar to the MICTREE cell of Subsection III.A. In a first experimental stage, which is the subject of this Subsection, we designed an artificial cell embedding a special kind of binary decision machine, a *shift binary decision machine*, with a read/write memory capable of storing 36 10-bit micro-instructions for the operative genome `OG` [4]. Assembling two such cells allows us to realize the simplest multicellular organism, a one-dimensional two-cell organism (Fig. 4.8). The specifications of the organism are those of a modulo-60 counter, which is in fact a subset of StopWatch (Subsection III.B). The operative genome `OG` consists of two genes, "`Countmod10`" and "`Countmod6`", whose execution depends solely on the X coordinate.

The shift binary decision machine is specially designed to fit into an array of MUXTREE molecules: due to the difficulty of embedding a classic random access memory (RAM) in such an array (mainly due to the excessive number of molecules needed for decoding the RAM address), the actual program memory, or *shift memory*, consists of shift registers implemented using the D flip-flops of the MUXTREE molecules.

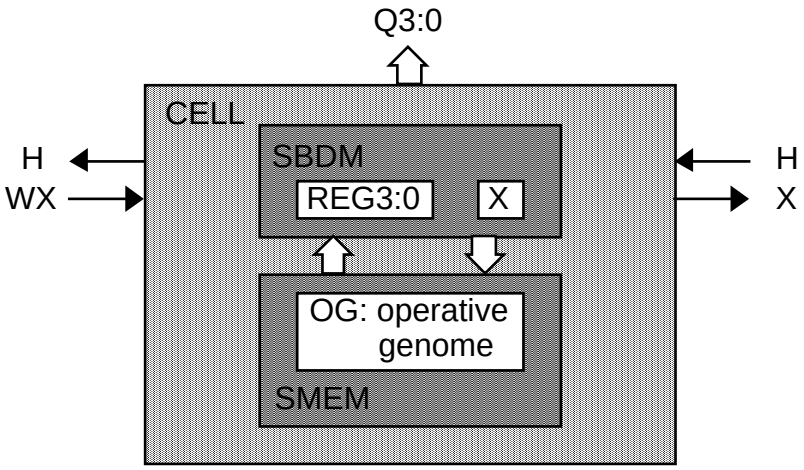
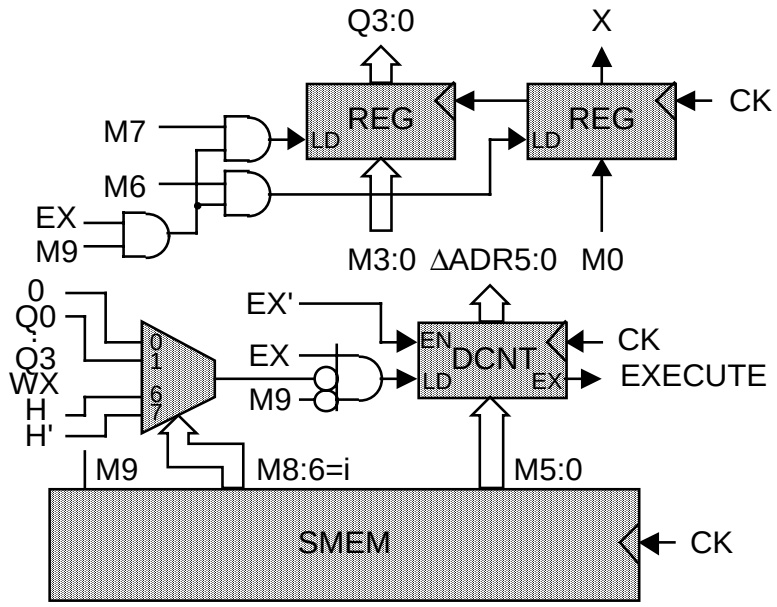


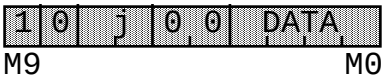
Fig. 4.9a

The final cell (Figs. 4.9a and 4.9b) presents two 1-bit input variables (the counter's clock signal `H` and the coordinate `WX` sent by the western neighbor), one 4-bit output variable (the counter state `Q3:0`), and two 1-bit output variables, (the coordinate `X` and the clock signal `H`). Its instruction set and the corresponding binary formats are shown in Fig. 4.9c.



(b)

if VARi **else** ΔADR



do Yj=DATA

(c)

Fig. 4.9 Shift binary decision machine (SBDM). (a) Block diagram; SMEM: shift memory. (b) Logic diagram; DCNT: down-counter. (c) Instruction set and binary format.

The shift memory requires the use of an instruction down-counter. The 36 instructions of the program are stored in the shift register SMEM and are continually shifted at each cycle of the internal clock CK. Their execution depends on the state of a logic signal EXECUTE, which detects the state ΔADR5:0=00000 of the down-counter DCNT. We can then identify the following two modes of operation:

- For EXECUTE=0 (ΔADR≠0), the test (**if**…) and assignment (**do**…) instructions have no effect.
- For EXECUTE=1 (ΔADR=0), the assignment instruction (**do** Yj = DATA) is executed. For VARi=1, the execution of the test instruction (**if** VARi **else** ΔADR) has no effect, while if the opposite is true (VARi=0) the value ΔADR

(which indicates the number of instructions not to be executed) is charged into the down-counter DCNT.

```
00  if WX else 02        12  if Q0 else 05
01  do X=0               13  if WX else 02
02  if 0 else 01         14  do Q=0110
03  do X=1               15  if 0 else 0E
04  if H else 23         16  do Q=0000
05  if H' else 23        17  if 0 else 0C
06  if Q3 else 04        18  do Q=0101
07  if Q0 else 01        19  if 0 else 0A
08  if 0 else 0D         1A  if Q1 else 05
09  do Q=1001            1B  if Q0 else 02
0A  if 0 else 19         1C  do Q=0100
0B  if Q2 else 0E        1D  if 0 else 06
0C  if Q1 else 05        1E  do Q=0011
0D  if Q0 else 02        1F  if 0 else 04
0E  do Q=1000            20  if Q0 else 02
0F  if 0 else 14         21  do Q=0010
10  do Q=0111            22  if 0 else 01
11  if 0 else 12         23  do Q=0001
```

Fig. 4.10 Modulo-60 counter operative genome OG.

As shown in Fig. 4.10, the modulo-60 counter program, i.e., the operative genome OG of the artificial organism, is 36 instructions long (ADR=00 to 23 in hexadecimal notation).

The layout of the cell, with one spare column every three columns, is an array of 30x30=900 MUXTREE molecules (Fig. 4.11), where the white molecules have no logic functionality but are used exclusively for interconnections. This structure involves the following hardware resources:

- a 36x10-bit shift memory SMEM;
- a 6-bit down-counter DCNT;
- a 4-bit register REG to store the state Q;
- a 1-bit register REG to store the horizontal coordinate X;
- an 8-to-1 test variable multiplexer;
- a 2-to-2 demultiplexer to load a variable;
- a couple of random logic gates.

The ribosomic genome RG is the sum of the 20x30=600 MOLCODEs of the 600 active MUXTREE molecules. The polymerase genome PG can be inferred from Subsection B and has the following form:

```
PG = C,{V}*[29],H,S,{H,H,S}*[9],C…
```

The final organism [4], consisting of at least two cells as in Fig. 4.11, has been successfully simulated thanks to the VHDL language. Only the self-repair mechanism, both at the cellular and at the molecular levels, is missing from this first realization of an artificial cell based on an array of MUXTREE molecules.

G. Final Remarks

The main focus of this Section was the description of a new FPGA molecule, called MUXTREE, based on a programmable multiplexer with the following additions:

- an automaton, the space divider, used to divide the molecular array into subsets of identical dimensions, the cells;
- a built-in self-test mechanism capable of detecting, localizing, and either repairing a faulty molecule at the molecular level or, should this prove impossible, generating a `KILL` signal which activates the self-repair at the cellular level.

Any cell made up of MUXTREE molecules satisfies the three features of the Embryonics project (Subsection II.D): multimolecular organization, molecular configuration, and molecular fault detection. The MUXTREE molecule, itself realized with a commercial FPGA, was embedded into a demonstration module, and we showed that an array of such modules exhibits the two desired properties of cellular self-replication and cellular self-repair.

The trivial applications of the MUXTREE molecule are those of unicellular organisms: the genome and the gene are then indistinguishable and the calculation of the coordinates is superfluous. The cell is then equivalent to a hardwired logic system and is defined exclusively by its ribosomic and polymerase genomes, the operative genome being superfluous: this is the case of the modulo-4 up-down counter of Subsection E, realized with six molecules. The non-trivial applications are those of multicellular organisms, in which the cells in an array have different genes: the genome is then a collection of genes, and the coordinates become indispensable. The cell is then a binary decision machine which executes a program equivalent to the operative genome `OG`. In order to minimize the hardware resources, a possible implementation of a cell is based on a particular type of binary decision machine, coupled with a shift memory: this is the case of the modulo-60 counter described in Subsection F, realized with 900 molecules. In this last example, the minimization hardware results in a slowdown in the execution of the program, since no jumps are possible and all the instructions have to be accessed sequentially.

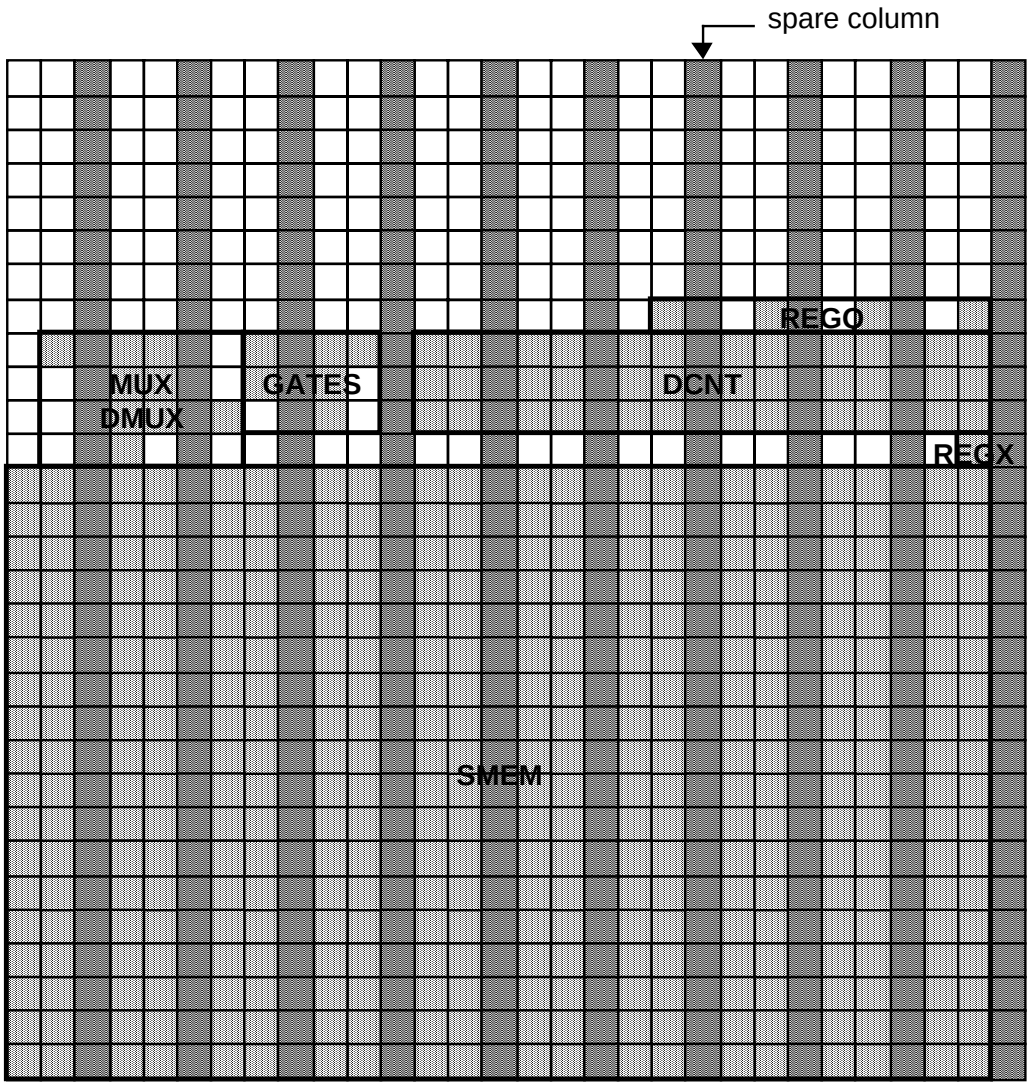


Fig. 4.11 Floor plan of the shift binary decision machine (array of 30 x 30 = 900 MUXTREE molecules, with a spare column every three columns).

The way is thus open for the realization of cells of any complexity, based on our novel FPGA, i.e., our array of MUXTREE molecules.

V. Conclusion

A. The Trials and Errors of Embryonics

The design and implementation of the demonstration modules of MICTREE, our artificial cell (Subsection III.A), and MUXTREE, our artificial molecule (Subsection IV.D), are but milestones on a long road leading to two very different future products: the microscopic molecule which will form the heart of a new self-repairing VLSI circuit (Subsection V.C), and the macroscopic molecule, currently under construction, to be used in the giant BioWatch 2001 project (Subsection V.C). We can divide our experimental process into three phases.

In the first (historical) phase [15] our initial project was based upon a simplified three-level hierarchy, instead of the four-level hierarchy of Figure 2.10. Each artificial cell included a configuration layer (composed of a processor executing the artificial genome, calculating the coordinates, and, as a function of these coordinates, determining the 20-bit gene), and an application layer (composed of a single multiplexer with connections controlled by the gene). In practice, our entire artificial genome was used to determine the functionality of what is now only one of our artificial molecules. A demonstration module, the BIODULE 600, was designed and implemented [15], allowing the experimental verification of the concepts of Embryonics (self-repair and self-replication) in very simplified examples.

The main drawback of the BIODULE 600 cell was the lack of balance between the application layer (a multiplexer with a single control variable) and the configuration layer (a processor storing and executing the program genome). The development of a new cell, called MICTREE, constituted the second phase of the Embryonics project, and was aimed at correcting this imbalance. In the MICTREE cell, the application and configuration layers are indistinguishable. By accepting a reduction in execution speed (the program is executed sequentially as opposed to multiplexers working in parallel), we obtain a considerable gain in computation power (1024 executable instructions per cell instead of a multiplexer, equivalent to a single test instruction).

The demonstration module implementing the MICTREE cell revealed two major shortcomings:

- The finite dimensions of the cell (memory, registers, etc.) prevented us from implementing digital systems of any dimension.

- The lack of an automatic built-in self-test system.

In the third phase of the project we were naturally led to the design of a new cell with a flexible architecture: a fine-grained FPGA based on the MUXTREE molecule, was the answer to this new challenge.

But the demonstration module of the MUXTREE molecule (Section IV.D) will need to be modified to constitute the elemental unit of future implementations (VLSI circuits and the giant BioWatch 2001). All the mechanisms involved in the display of results and the manual injection of faults will be removed; the principal shortcoming of the MUXTREE module -- the very small memory (1 bit per molecule) -- must be changed radically. To overcome this difficulty, we are designing a mechanism which will allow us to use the 20-bit configuration register for memory storage, reducing considerably the size of the artificial cell.

In the first three phases of our project, as well as in the work currently in progress, we have observed the same fundamental mechanism: linear information within the artificial genome configures a two-dimensional physical substrate -- our FPGA -- to generate the desired application:

Genome + FPGA = Application.

Table 1 presents experimental data concerning the genomes of the applications described in Ref. [15], in this paper, and those currently in progress. Ignoring the contribution of the polymerase genome (PG) we can state the following two observations:

- The ribosomic genome (RG) comprises the major part of the complete genome; to wit, the configuration string of the FPGA is of much higher complexity than the executed program (the operative genome, OG). We note that with living beings the majority of the genetic material also consists of the ribosomic genome.
- The complexity of the ribosomic genomes of the BIODULE 600 and MICTREE elements (the configuration string of the commercial FPGA) is an order of magnitude greater than the ribosomic genomes necessitated by the elements of the MUXTREE and NEW MUXTREE FPGAs, constructed specifically for the implementation of the Embryonics project

In conclusion, we note that configuring (ribosomic genome) is much more complex than programming (operative genome). The development of an FPGA adapted to our project diminishes greatly the complexity of the configuration task.

B. A Scientific Challenge: Von Neumann Revisited

The early history of the theory of self-replicating machines is basically the history of John von Neumann's thinking on the matter [8], [21]. Von Neumann's automaton is a homogeneous two-dimensional array of elements, each element being a finite state machine with 29 states. In his historic work, von Neumann showed that a possible configuration (a set of elements in a given state) of his automaton can implement a *universal constructor* able to build onto the array any computing machine described in a dedicated part of the universal constructor, the *tape*. Self-replication is then a special case of construction, occurring when the universal constructor itself is described on the tape. Moreover, von Neumann demonstrated that his automaton is endowed with two major properties: *construction universality*, the capability of describing on the tape and building onto the array a machine of any dimension, and *computation universality*, the capability of describing and building a universal Turing machine.

It must be reminded that, in biology, the *cell* is the smallest part of the living being containing the complete blueprint of the being, the genome. On the basis of this definition, von Neumann's automaton can be considered as a *unicellular organism*, since it contains a single copy of the genome, i.e., the description stored on the tape. Each element of the automaton is thus a part of the cell, i.e., a *molecule*. Von Neumann's automaton, therefore, is a *molecular automaton*, and self-replication is a very complex process due to the interactions of hundreds of thousands of molecules.

Arbib [22] was the first to suggest a truly "cellular" automaton, in which every cell contains a complete copy of the genome, and a hierarchical organization, where each cell is itself composed of smaller regular elements, the "molecules". The Embryonics project is therefore the first actual implementation of Arbib's concept, as each of its elements contains a copy of the genome. Each element of our automaton is thus a cell in the biological sense, and our automaton is truly a *multicellular automaton.*

The verification of the property of *universal computation*, that is, the design of a universal Turing machine on our multicellular array, is one of the major ongoing projects in our laboratory (note that we have already shown in Subsection III.D the implementation of a *specialized* Turing machine, a parenthesis checker). The property of *universal construction* raises issues of a different nature, since it requires (always according to von Neumann) that our MICTREE cells be able to implement organisms of any dimension. This challenge is met, as shown in Section IV, by decomposing a

cell into molecules and tailoring the structure of cells to the requirements of a given application.

In conclusion, the original specifications of the historical automaton of von Neumann will be entirely satisfied after the implementation of a universal Turing machine on a multicellular array, and after the realization of the corresponding cells on our FPGA composed of MUXTREE molecules. The dream of von Neumann will then become a reality, with the additional properties of self-repair and real-time operation; moreover, we envisage the possibility of *evolving* the genome using genetic algorithms.

C. A Technical Challenge: Towards New Robust Integrated Circuits

Keeping in mind that our final objective is the development of very large scale integrated (VLSI) circuits capable of self-repair and self-replication, as a first step, which is the subject of this paper, we have shown that a hierarchical organization based on four levels (molecule, cell, organism, population of organisms) allows us to confront the complexity of real systems (Section II). The realization of demonstration modules at the cellular level (MICTREE cells, Section III) and at the molecular level (MUXTREE molecule, Section IV) demonstrates that our approach can satisfy the requirements of highly diverse artificial organisms and attain the two sought-after properties of self-repair and self-replication.

The programmable robustness of our system depends on a redundancy (spare molecules and cells) which is itself programmable. This feature is one of the main original contributions of the Embryonics project. It becomes thus possible to program (or re-program) a greater number of spare molecules and spare cells for operation in hostile environments (e.g., space exploration). A detailed mathematical analysis of the reliability of our systems is currently under way at the University of York [40][41].

As we have seen, the MUXTREE molecule (Section IV) is the main hardware prototype we realized in order to test the validity of our approach. However, the size of the demonstration module used to implement a single MUXTREE molecule prevents us from realizing systems which require more than a few logic elements. In the long term, we hope to overcome this difficulty through the realization of a dedicated VLSI circuit which will contain a large number of elements; in the short term, however, such a solution is not yet within our reach. To obtain a larger number of programmable elements, we investigated the possibility of exploiting a system based on an array of Xilinx FPGAs mounted on a single printed circuit board and

configured so as to implement an array of MUXTREE molecules [16]. Such a system, while far from affording the same density as a VLSI chip, would nevertheless allow us to obtain a much larger number of elements than an array of demonstration modules (Fig. 4.6a), particularly since we would not be limited to a single MUXTREE molecule for each Xilinx chip.

The first step in the design of this system was therefore an analysis of the number of MUXTREE molecules which can be realized in a single Xilinx FPGA. To this end, we defined a layout consisting of a 4x4 array of our logic elements (Fig. 5.1). Without attempting any kind of optimization in the layout of the molecules, we removed the logic dedicated exclusively to the demonstration module and tried to determine the smallest Xilinx FPGA capable of containing the whole array. Running our design through the Xilinx routing software produced some very disappointing results: we determined that the smallest FPGA that can hold the entire array is a XC4025, that is, an FPGA theoretically capable of realizing circuits of up to 25000 logic gates (many more than those required by our 4x4 array of molecules). While a system based on an array of such chips could allow us to obtain a fairly large array of MUXTREE molecules, and indeed would be an interesting intermediate step in the creation of our VLSI circuit, it is unlikely to allow the realization of systems requiring hundreds of molecules. One of the next steps in our project, which we will begin as soon as we are in possession of a quasi-definitive version of our FPGA, will be the design of an optimized layout of our cell, to be implemented, in all probability, on an array of Xilinx FPGAs of the 6200 family. In fact, this family of FPGAs, while unfortunately discontinued by Xilinx, nevertheless presents a number of advantages as far as our project is concerned, and notably the striking resemblance between its elements and our MUXTREE molecules (which could theoretically allow us a one-to-one mapping of our molecules to the elements).

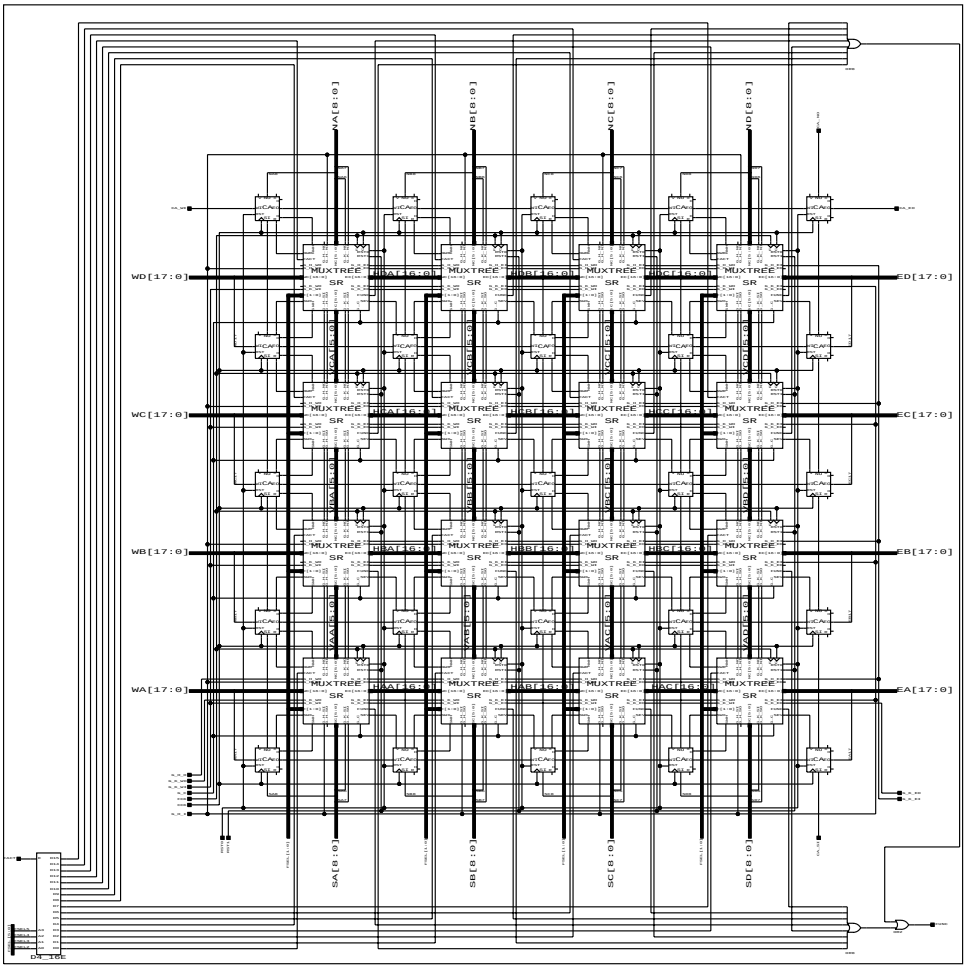


Fig. 5.1 Layout of a 4x4 array of MUXTREE molecules
using a single Xilinx XC 4025 HQ 240 FPGA.

To the best of our knowledge, there exist today few projects, industrial or academic, which aim at integrating the properties of self-repair and/or of self-replication on FPGAs. In the framework of the Embryonics project, a fine-grained FPGA based on a demultiplexer [4], [23] and a coarse-grained FPGA based on a binary decision machine [24] have been developed at the Centre Suisse d'Électronique et de Microtechnique in Neuchâtel (Switzerland). A fine-grained FPGA based on a multiplexer is also under development at the University of York (United Kingdom) [25]. Industrial projects dealing with self-repairing FPGAs (without self-replication) are also underway at NEC (Japan) [20] and at Altera (U.S.A.) [31].

In our laboratory, the next major step in the Embryonics project is the design of the BioWatch 2001, a complex machine which we hope to present on the occasion of a major scientific and cultural event which will take place in the year 2001 in Switzerland. The function of the machine will be that of a self-replicating and self-

repairing watch, implemented with macroscopic versions of our artificial cells and molecules (Fig. 5.2).

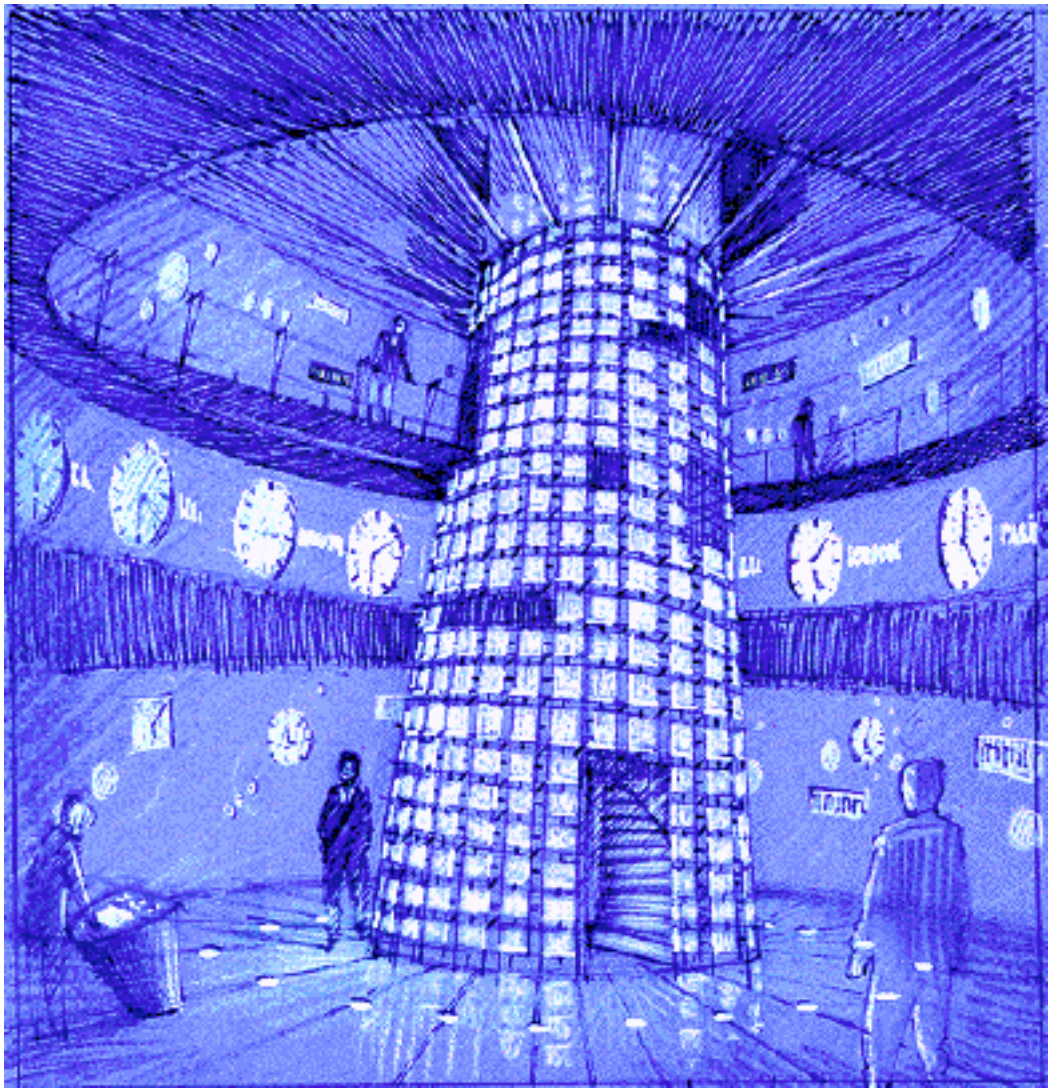


Fig. 5.2 An artist's rendition of a possible realization of the BioWatch 2001 [Art by Anne Renaud].

A far-future technical application of the Embryonics project is in the domain of nanotechnology [30]. The concept of a self-replicating machine, or "assembler", capable of arranging "the very atoms" was first introduced by Drexler as a possible solution to the problem of the increasing miniaturization of VLSI circuits: as manufacturing technology advances beyond conventional lithography, some new, accurate, and low-cost approach to the fabrication of VLSI circuits is required, and self-replicating assemblers could be a remarkably powerful tool for this kind of application.

D. A Biological Challenge: Artificial and Natural Genomes

In our Embryonics project, the design process for a multicellular automaton requires the following stages:

- The original specifications are mapped onto a homogeneous array of cells (binary decision machines with their associated read/write memory). The software (a microprogram) and the hardware (the architecture of the cell) are tailored according to the needs of the specific application (Turing machine, electronic watch, random number generator, etc.). In biological terms, this microprogram can be seen as the *operative genome* `OG`, or, in other words, the *operative part* of the final artificial genome. In the example of our specialized Turing machine, the parenthesis checker (Subsection III.D), the operative part of the genome consists of (Fig. 5.3): *coordinate genes* (`Xcoord`, `Ycoordlocalconfig`, `Initcond`), which handle the computation of the coordinates and the calculation of the initial conditions, similar to the *homeboxes* or *HOX genes* recently found to define the general architecture of living beings [26]; *switch genes* (`G` and `SY0` tests), used to express the functional genes according to the cell's position in the organism (that is, according to the value of the cell's coordinates) [27]; and *functional genes* (`Headgene`, `Tapegene`), which effect the operative functions of our artificial organism (i.e., calculating the head and tape states), analogous to the genes which constitute the coding part of the natural genome.

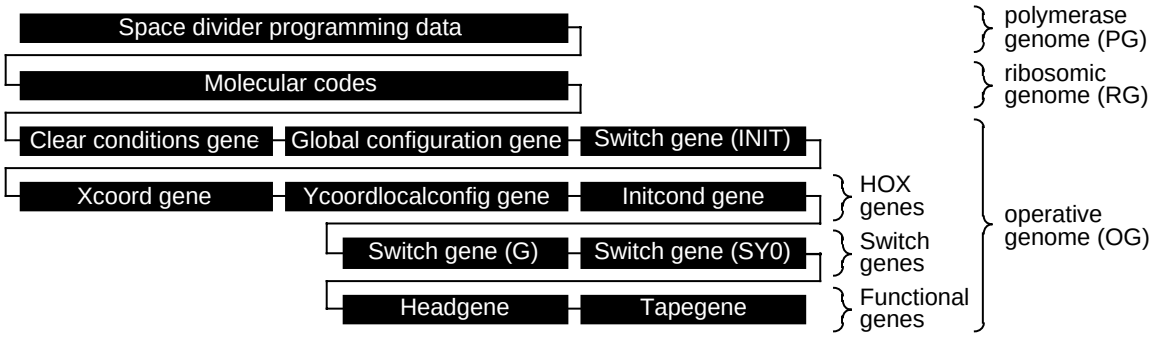


Fig. 5.3 The artificial genome of the parenthesis checker.

- The hardware of the cell is implemented with a homogeneous array of molecules, the MUXTREE molecules. Spare columns are introduced in order to improve the global reliability. With our artificial cell, being analogous to the ribosome of a natural cell, the string of the molecular codes `MOLCODEs` can be considered as the *ribosomic genome* `RG` or the *ribosomic part* of the final genome.

- The dimensions of the final molecular array, as well as the frequency of the spare columns, define the string of data required by the finite state machine, the space divider, which creates the boundaries between cells. As this information will allow to create all the daughter cells starting from the first mother cell, it can be considered as equivalent to the *polymerase genome* PG or the *polymerase part* of the genome.

With respect to this design process, the programming of the molecular array of MUXTREE elements takes place in reverse order:

- the polymerase genome is injected in order to set the boundaries between cells;
- the ribosomic genome is injected in order to configure the molecular FPGA and to fix the final architecture of the cell;
- the operative genome is stored within the read/write memory of each cell in order for it to execute the specifications.

The existence of these different categories of genes is the consequence of purely logical needs deriving from the conception of our multicellular automaton.

One of the most promising domains of molecular biology, *genomics*, is the research of a syntax of the genome, that is, rules dictating the ordering of different parts of the genome, the genes [28], [29]. One can imagine the artificial and the natural genomes sharing common, invariant properties. Should this indeed be the case, the Embryonics project could contribute to biology itself [32][42].

Acknowledgements

We are grateful to Anne Renaud for her rendition of a possible realization of the BioWatch 2001, to Alain Herzog for the photographs (Figs. 3.4b and 4.6b), and to André Badertscher for the implementation of the MUXTREE molecule and of the MICTREE cell.

This work was supported in part by the Swiss National Foundation under grant 21-54113.98, by the Consorzio Ferrara Richerche, Università di Ferrara, Ferrara, Italy, and by the Leenaards Foundation, Lausanne, Switzerland.

References

[1] L. Wolpert. *The Triumph of the Embryo*. Oxford University Press, New York, 1991.

[2] M. Nicolaidis. "Future Trends in Online Testing: a New VLSI Design Paradigm?". *IEEE Design and Test of Computers*, Vol. 15, No. 4, 1998, p. 15.

[3] D. Mange. *Microprogrammed Systems: An Introduction to Firmware Theory*. Chapman & Hall, London, 1992.

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

[5] D. Mange, M. Goeke, D. Madon, A. Stauffer, G. Tempesti, S. Durand. "Embryonics: A New Family of Coarse-Grained Field-Programmable Gate Array with Self-Repair and Self-Reproducing Properties". In E. Sanchez, M. Tomassini, eds., *Towards Evolvable Hardware*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1062, Berlin, 1996, pp. 197-220.

[6] S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[7] P. D. Hortensius, R. D. McLeod, B. W. Podaima. "Cellular Automata Circuits for Built-In Self-Test". *IBM Journal of Research and Development*, Vol. 34, No. 2/3, 1990, pp. 389-405.

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

[9] D. Mange, D. Madon, A. Stauffer, G. Tempesti. "Von Neumann Revisited: A Turing Machine with Self-Repair and Self-Reproduction Properties". *Robotics and Autonomous Systems*, Vol. 22, No. 1, 1997, pp. 35-58.

[10] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[11] E. R. Berlekamp, J. H. Conway, R. K. Guy. "What is Life?". *Winning Ways for Your Mathematical Plays*, Academic Press, New York, 1982, pp. 817-850.

[12] S. B. Akers. "Binary Decision Diagrams". *IEEE Transactions on Computers*, C-Vol. 27, No. 6, June 1978, pp. 509-516.

[13] R. E. Bryant. "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams". *ACM Computing Surveys*, Vol. 24, No. 3, 1992, pp. 293-318.

[14] C. Meinel, T. Theobald. *Algorithms and Data Structures in VLSI Design*. Springer-Verlag, Berlin, 1998.

[15] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, C. Piguet. "Embryonics: A New Methodology for Designing Field-Programmable Gate Arrays with Self-Repair and Self-Replicating Properties". *IEEE Transactions on VLSI Systems*, Vol. 6, No. 3, September 1998, pp. 387-399.

[16] G. Tempesti. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes*. Ph.D. Thesis No. 1827, EPFL, Lausanne, 1998.

[17] G. Tempesti, D. Mange, A. Stauffer. "Self-Replicating and Self-Repairing Multicellular Automata". *Artificial Life*, Vol. 4, No. 3, 1998, pp. 259-282.

[18] G. Tempesti, D. Mange, A. Stauffer. "A Robust Multiplexer-Based FPGA Inspired by Biological Systems". *Journal of Systems Architecture: Special Issue on Dependable Parallel Computer Systems*, Vol. 43, No. 10, 1997.

[19] R. Negrini, M.G. Sami, R. Stefanelli. *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays*. The MIT Press, Cambridge, MA, 1989.

[20] A. Shibayama, H. Igura, M. Mizuno, M. Yamashina. "An Autonomous Reconfigurable Cell Array for Fault-Tolerant LSIs". In: *Proc. 44th IEEE International Solid-State Circuits Conference*, San Francisco, California, February 1997, pp. 230-231 and 462.

[21] M. Sipper. "Fifty Years of Research on Self-Replication: an Overview". *Artificial Life*, Vol. 4, No. 3, 1998, pp. 237-257.

[22] M. A. Arbib. *Theories of Abstract Automata*. Prentice-Hall, New Jersey, 1969.

[23] P. Marchal, C. Piguet, D. Mange, A. Stauffer, S. Durand. "Embryological Development on Silicon". *Artificial Life IV*, MIT Press, 1994, pp. 365-370.

[24] P. Nussbaum, B. Girau, A. Tisserand. "Field Programmable Processor Arrays". In M. Sipper, D. Mange, A. Perez-Uribe, eds., *Evolvable Systems: From Biology to Hardware*, Lecture Notes in Computer Science, Vol. 1478, Springer-Verlag, Berlin, 1998, pp. 311-322.

[25] C. Ortega, A. Tyrrell, "MUXTREE revisited: Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays". In M. Sipper, D. Mange, A. Perez-Uribe, eds., *Evolvable Systems: From Biology to Hardware*, Lecture Notes in Computer Science, Vol. 1478, Springer-Verlag, Berlin, 1998, pp. 206-217.

[26] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. Argetsinger Steitz, A. M. Weiner. *Molecular Biology of the Gene*. Benjamin-Cummings Publishing Company, Menlo Park, California, 4th edition, 1987.

[27] S. F. Gilbert. *Developmental Biology*. Sinauer Associates Inc., Massachusetts, 3rd edition, 1991.

[28] D. Duboule. "The Evolution of Genomics". *Science*, Vol. 278, 24 Oct. 1997, p. 555.

[29] S. Bentolila. "A Grammar Describing "Biological Binding Operators" to Model Gene Regulation". *Biochimie* 78, 1996, pp. 335-350.

[30] R. C. Merkle. "Making Smaller, Faster, Cheaper Computers". *Proceedings of the IEEE*, Vol. 86, No. 11, November 1998, pp. 2384-2386.

[31] C. F. Lane, S. T. Reddy, B. I Wang. *Means and apparatus to minimize the effect of silicon processing defects in programmable logic devices*. Altera Corporation. U.S. Patent 5592102. Filed Oct. 19, 1995.

[32] M. Barbieri. "The Organic Codes: The Basic Mechanism of Macroevolution". *Rivista di Biologia / Biology Forum* 91, 1998, pp. 481-514.

[33] "A D&T Roundtable: Online Test". *IEEE Design & Test of Computers*, Vol. 16, No. 1, January-March 1999, pp. 80-86.

[34] Y. Zorian. "Testing the Monster Chip". *IEEE Spectrum*, Vol. 36, No. 7, July 1999, pp. 54-60.

[35] G. D. Watkins. "Novel Electronic Circuitry", Predictive Paper Reprint. *Proceedings of the IEEE*, Vol. 86, No. 11, November 1998, p. 2383.

[36] P. Kuekes. "Molecular Manufacturing: Beyond Moore's Law". Invited Talk. *Proc. Field-Programmable Custom Computing Machines (FCCM'99)*, Napa, CA, April 1999.

[37] J. R. Heath, P. J. Kuekes, G. S. Snider, R. S. Williams. "A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology". *Science*, Vol. 280, No. 5370, 12 June 1998, pp. 1716-1721.

[38] R. F. Service. "Organic Molecule Rewires Chip Design". *Science*, Vol. 285, No. 5426, 16 July 1999, pp. 313-315.

[39] S. R. Park, W. Burleson. "Configuration Cloning: Exploiting Regularity in Dynamic DSP Architectures". *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'99)*, Monterey, CA, February 1999, pp. 81-89.

[40] C. Ortega, A. Tyrrell. "Reliability Analysis in Self-Repairing Embryonic Systems". *Proc. of the First NASA/DOD Workshop on Evolvable Hardware*, Pasadena, CA, July 1999, pp. 120-128.

[41] C. Ortega, A. Tyrrell. "Self-Repairing Multicellular Hardware: A Reliability Analysis". In D. Floreano, J.-D. Nicoud, F. Mondada, eds., *Advances in Artificial Life*, Lecture Notes in Artificial Intelligence, Vol. 1674, Springer-Verlag, Berlin, 1999, pp. 442-446.

[42] R. Gordon. *The Hierarchical Genome and Differentiation Waves*. World Scientific & Imperial College Press, Singapore and London, 1999.

[43] D. Mange, M. Sipper, P. Marchal. "Embryonic electronics". *BioSystems*, Vol. 51, No. 3, 1999, pp. 145-152.

[44] M. Sipper, D. Mange, E. Sanchez. "Quo Vadis Evolvable Hardware?". *Communications of the ACM*, Vol. 42, No. 4, April 1999, pp. 50-56.

[45] E. Sanchez, M. Sipper, J.-O. Haenni, J.-L. Beuchat, A. Stauffer, A. Perez-Uribe. "Static and Dynamic Configurable Systems". *IEEE Transactions on Computers*, Vol. 48, No. 6, June 1999, pp. 556-564.