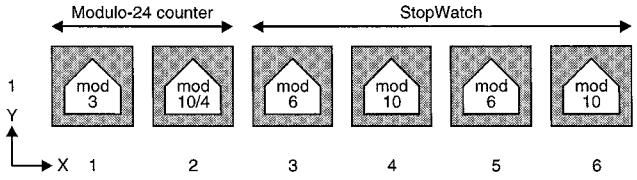


(a)

(b)

Fig. 14. BioWatch, a full digital watch. (a) Multicellular organization (`X=1`: tens of hours; `X=2`: units of hours). (b) Experiment with eight MICTREE cells and two spare cells.

that is, an architecture which is itself configurable. This architecture will be realized using a novel fine-grained FPGA.

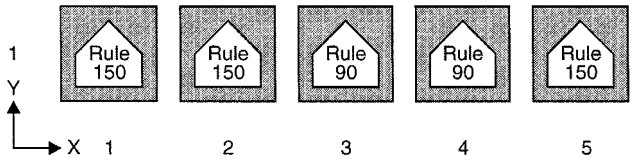
A consequence of our choices is that we require a methodology to generate, starting from a set of specifications, the configuration of our FPGA, consisting of a homogeneous network of elements, the *molecules*, defined by an identical architecture and a usually distinct state (the *molecular code*, or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

To fulfill this requirement, we have selected a particular representation: the *ordered binary decision diagram* (OBDD) [12]–[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 Section II-D (Fig. 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`.



(a)

```
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.15. A random number generator. (a) Multicellular organization. (b) The operative genome OG.

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

The plan of this section is the consequence of this structure. Section IV-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. Section IV-B introduces the *space divider*, a state machine that allows multiple molecules to be grouped in order to form a cell, and defines the polymerase genome PG. Section IV-C ends the description of the molecule with the introduction of the automatic fault detection system required for self-repair. Section IV-D presents the implementation of a prototype of our molecule, while Sections IV-E and -F present two applications of widely different complexity (a counter and a binary decision machine). Last, Section IV-F deals with the range of applications for our molecule and its limitations.

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. 17).

- Each of the two inputs of the multiplexer MUX (inputs 0 and 1) is 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. 18) allows any connection between the horizontal and vertical busses.

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 buses.

All the information necessary for programming the MUXTREE molecule, that is, the 17 field-programmable bits that 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. 17). 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. 18;
- the CB bits (MC18:12) define the inputs of the multiplexer MUX, as shown in Fig. 17.

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 (Section IV-C).

The mechanism 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. 19 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. 19 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. 19 and consists of a cycle of the following states:

$$PG = C, V, V, H, S, C, \ldots$$

where C represents a corner, V a vertical band, H an horizontal band, and S a 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

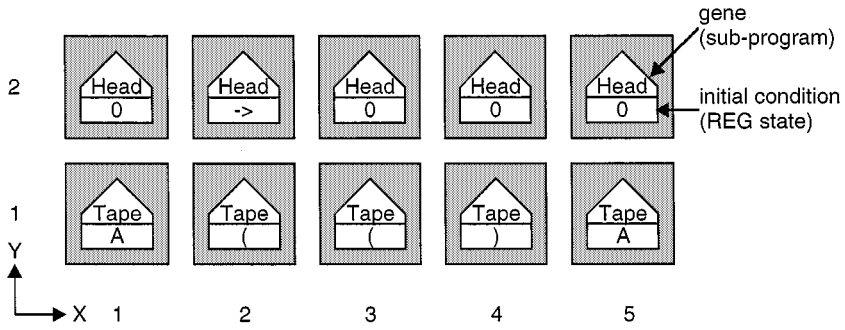


Fig. 16. Multicellular organization of a specialized Turing machine, a parenthesis checker.

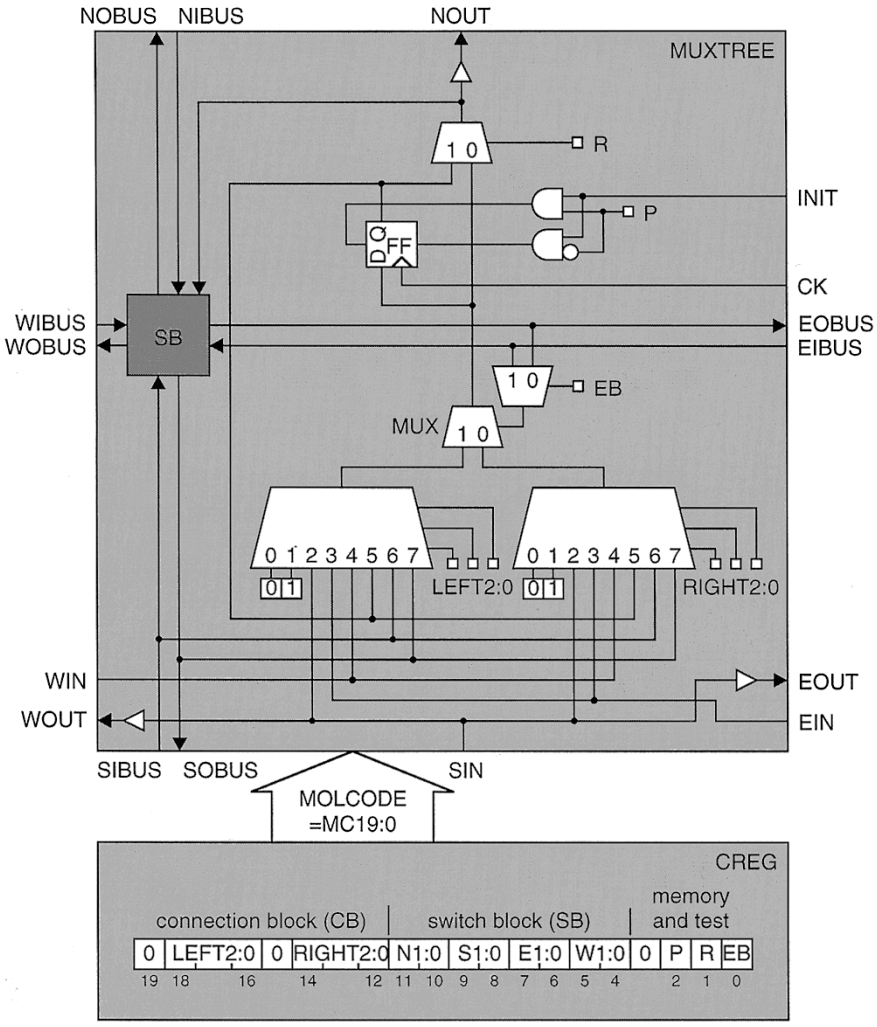


Fig. 17. Logic layout of a MUXTREE molecule, including the configuration register `CREG` and the switch block `SB`.

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.

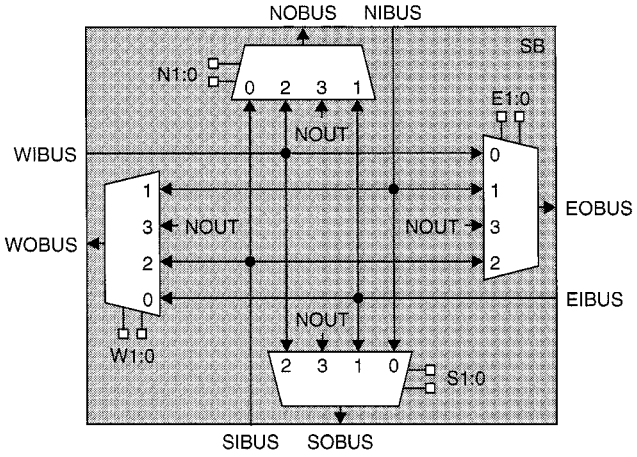


Fig. 18. Detailed architecture of the switch block SB.

- It must involve an acceptable overhead.
- 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 (Section 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. 20) [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.

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 (Section IV-B) by limiting the reconfiguration of the array to a single molecule per line between two spare columns (Fig. 21). 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 Section II-C.

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. 22(a)] [4], [16]. These containers can easily be joined to obtain two-dimensional arrays as large as desired [Fig. 22(b)].

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

- The molecule itself (Figs. 17, 18, and 20), 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. 19). 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 push-button 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 seven-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. 23) 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).

It can be verified that the two ordered binary decision diagrams Q1+ and Q0+ of Fig. 23(a) (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

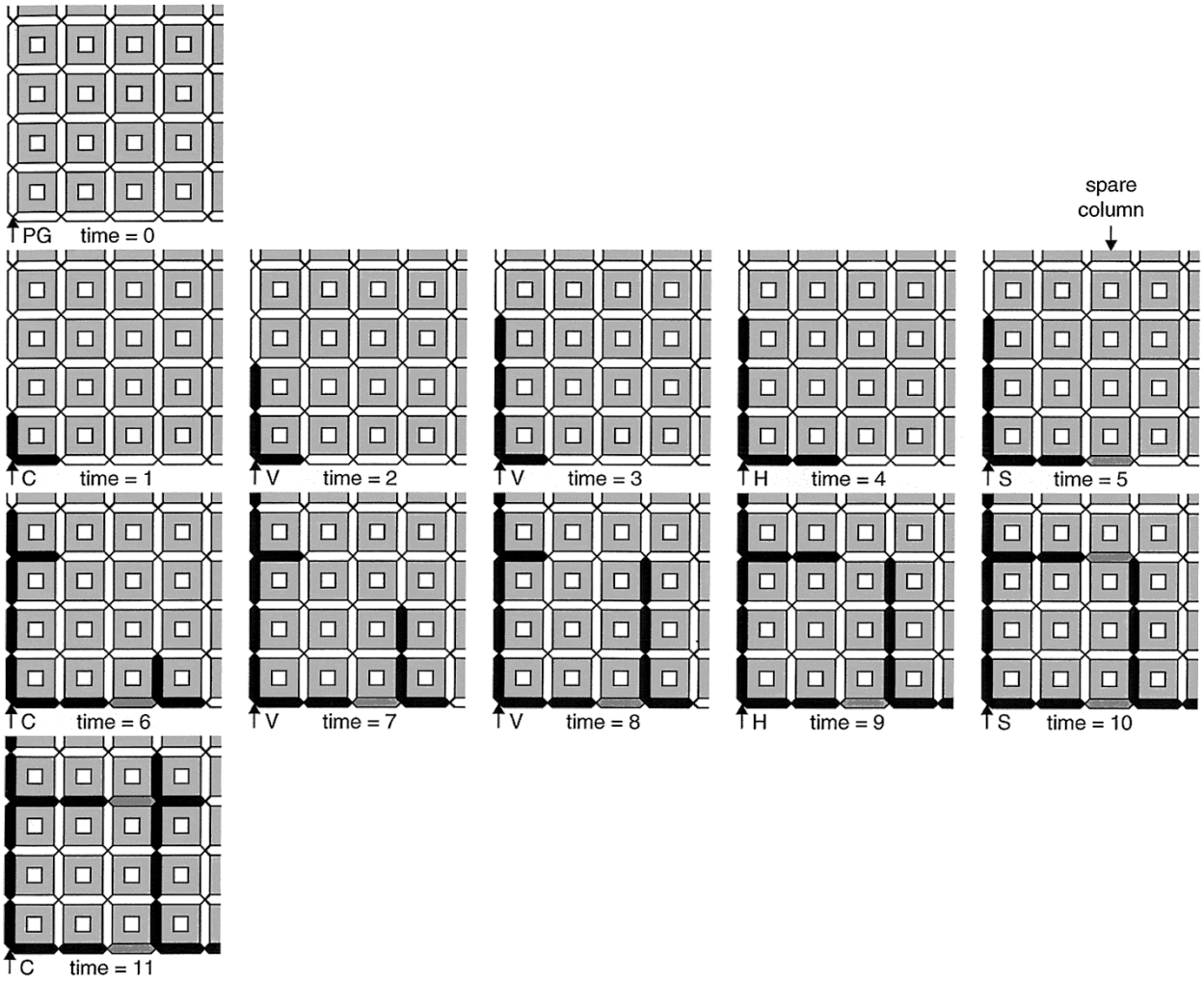


Fig. 19. Example of a space divider (height = 3, width = 3, 1 spare column out of 3); PG: polymerase genome: C, V, V, H, S, C, ...

values of the given functions (Q1+ and Q0+ in the example) computed with the following equations:

$$Q1+ = M\,(Q1 \cdot Q0 + Q1' \cdot Q0') + M'\,(Q1 \cdot Q0' + Q1' \cdot Q0)$$
$$Q0+ = Q0'.$$

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. 23(b)] 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. 23(b)] and, carried by the long-distance horizontal and vertical buses, become the control variables for the multiplexers of the bottom two rows.

The counter can be thus be implemented by an array of three rows by two columns, that is, by a cell made up of six MUXTREE molecules. From the multiplexer diagram of Fig. 23(b) and from the description of the MUXTREE molecule (Figs. 17 and 18), we can then compute the 17 control bits of each molecular code, finally generating the MOLCODEs of Fig. 23(c). 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. 23(c)].

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

In the preceding section, 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

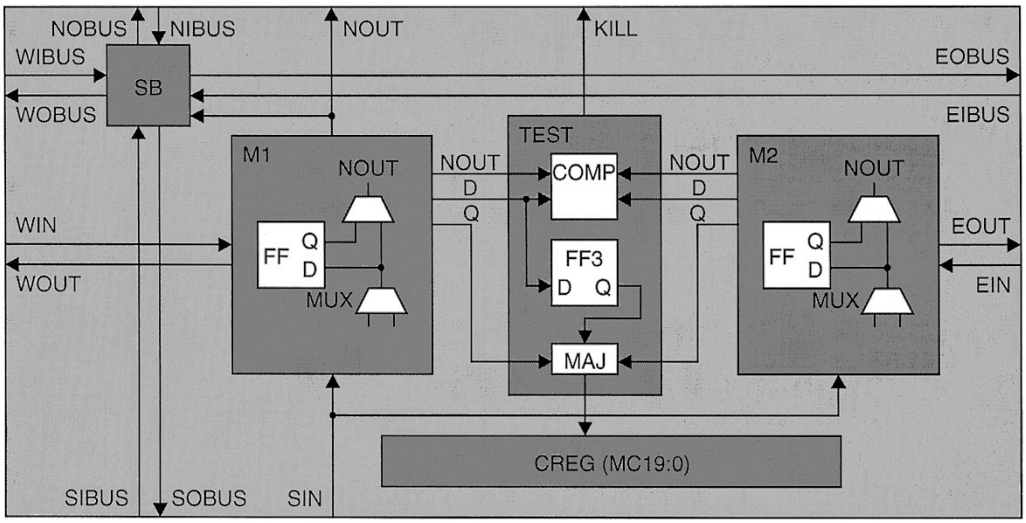


Fig. 20. A self-testing MUXTREE molecule using space redundancy.

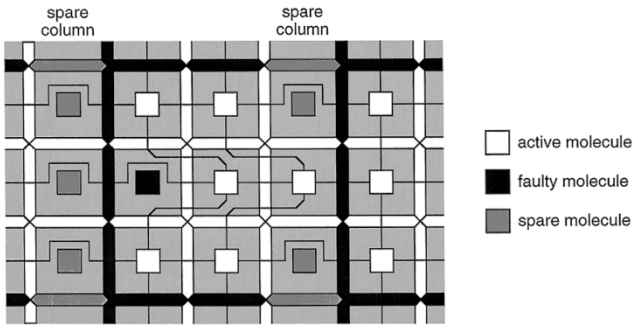


Fig. 21. The self-repair mechanism for an array of MUXTREE molecules.

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 Section III-A. In a first experimental stage, which is the subject of this section, 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. 24). The specifications of the organism are those of a modulo-60 counter, which is in fact a subset of StopWatch (Section 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.

The final cell [Fig. 25(a) and (b)] 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. 25(c).

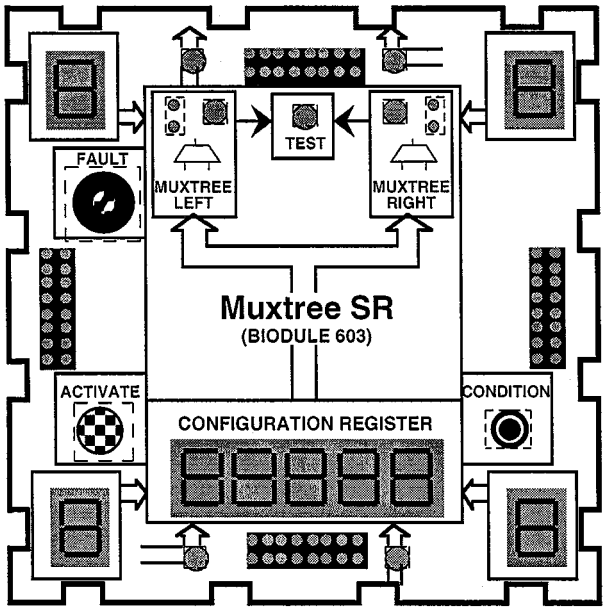
The shift memory requires the use of an instruction down-counter. The 36 instructions of the program are stored in the shift register SMEM and are continually shifted at each cycle of the internal clock CK. Their execution depends on the state of a logic signal EXECUTE, which detects the state ΔADR5:0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

- For EXECUTE = 0(ΔADR $\neq$ 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.

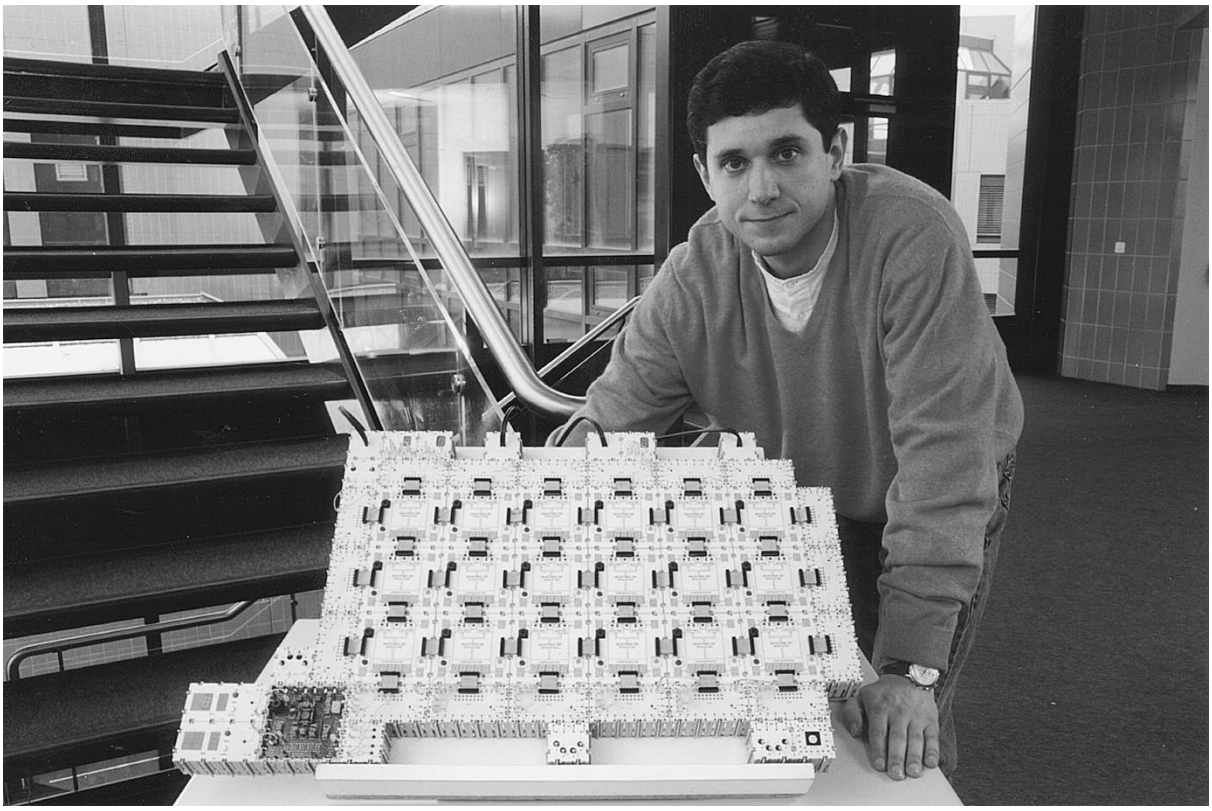
As shown in Fig.26, the modulo-60 counter program, i.e., the operative genome OG of the artificial organism, is 36 instructions long (ADR = 00 to 23 in hexadecimal notation). The layout of the cell, with one spare column every three columns, is an array of $30\times30 = 900$ MUXTREE molecules (Fig. 27), where the white molecules have no logic functionality but are used exclusively for interconnections. This structure involves the following hardware resources:

- a 36 $\times$ 10-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 $20 \times 30 = 600$ MOLCODE's of the 600 active MUXTREE molecules. The



(a)



(b)

Fig. 22. The MUXTREE molecule. (a) Front panel of a demonstration module implementing the molecule. (b) An array of MUXTREE molecules.

polymerase genome PG can be inferred from Section IV-B and has the following form:

$$\mathtt{PG} = \mathtt{C}, \{\mathtt{V}\}^*[29], \mathtt{H}, \mathtt{S}, \{\mathtt{H}, \mathtt{H}, \mathtt{S}\}^*[9], \mathtt{C}\ldots.$$

The final organism [4], consisting of at least two cells as in Fig.27, 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

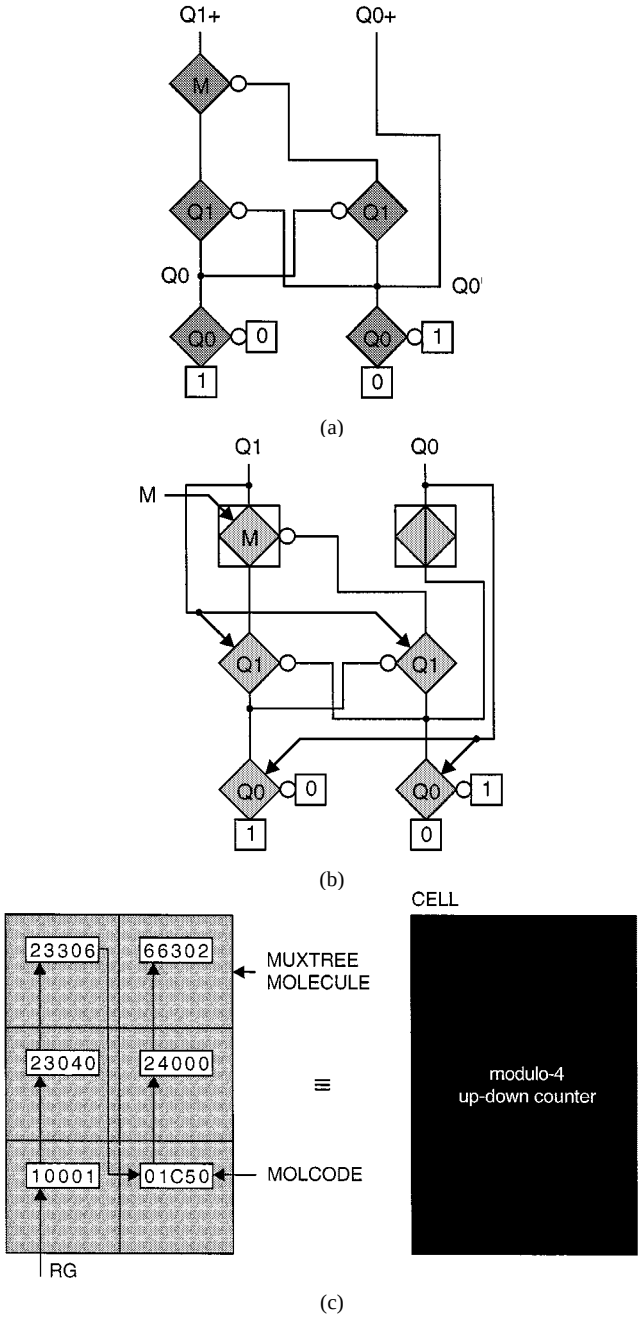


Fig. 23. Modulo-4 up–down counter. (a) Ordered binary decision diagrams for `Q1+` and `Q0+`. (b) Multiplexer diagram using MUXTREE molecules. (c) Six MUXTREE molecule cell; `RG`: ribosomic genome.

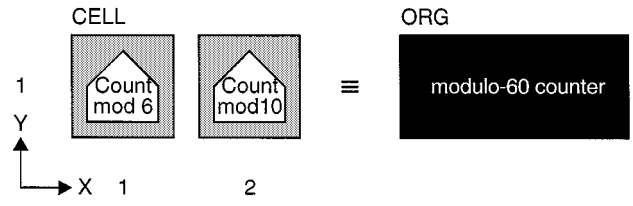


Fig. 24. A modulo-60 counter made up of two cells.

realization of an artificial cell based on an array of MUXTREE molecules.

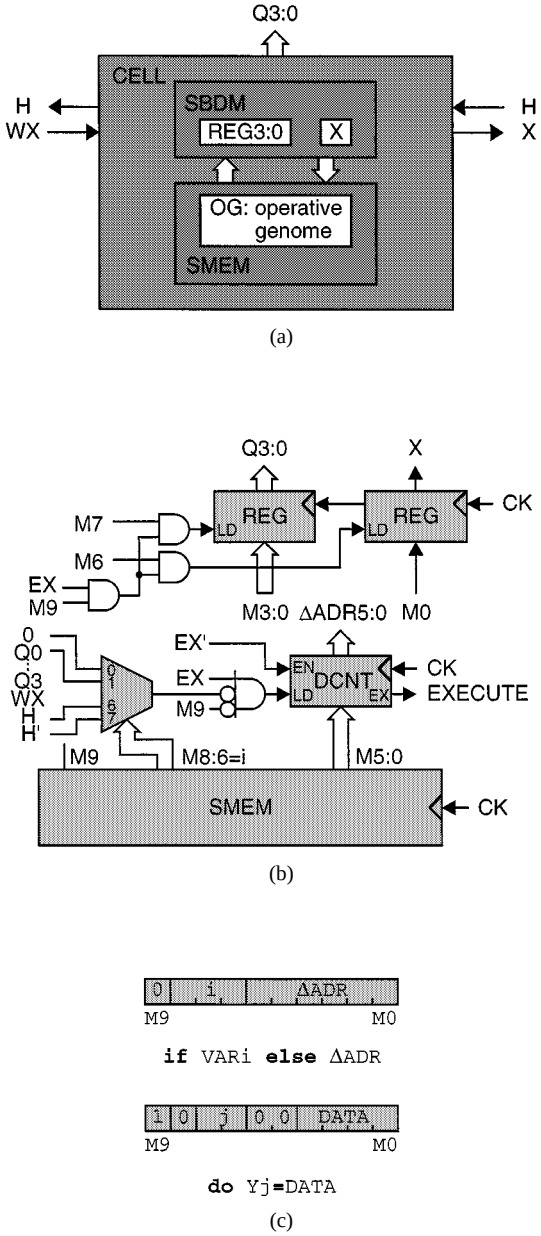


Fig. 25. Shift binary decision machine (SBDM). (a) Block diagram; `SMEM`: shift memory. (b) Logic diagram; `DCNT`: down-counter. (c) Instruction set and binary format.

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 (Section 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 Section IV-E, realized with six molecules. The nontrivial 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 Section IV-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.

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 (Section III-A), and MUXTREE, our artificial molecule (Section 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 (Section V-C), and the macroscopic molecule, currently under construction, to be used in the giant BioWatch 2001 project (Section 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 Fig. 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.

```
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. 26. Modulo-60 counter operative genome OG.

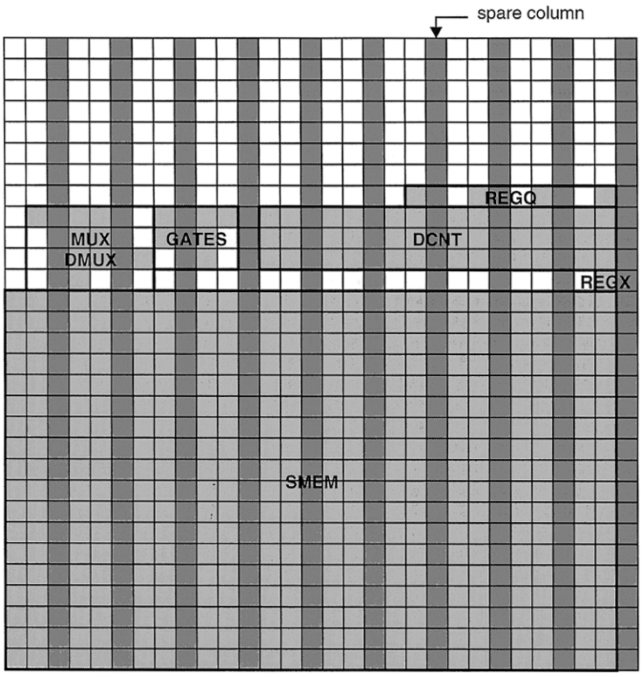


Fig.27. Floor plan of the shift binary decision machine (array of $30 \times 30 = 900$ MUXTREE molecules, with a spare column every third column).

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.

Table 1
Embryonics Project: Experimental Data. To Simplify, We Omitted in the Calculation of the Complete Genome the Contribution of the Polymerase Genome (PG)

Basic Elements	References	Applications	Operative Genome (OG) Number of instructions	Operative Genome (OG) Number of bits	Ribosomic Genome (RG) Number of bits	Complete Genome (OG+RG) Number of bits
BIODULE 600	[15]	Modulo-4 Up-Down Counter	107	856 (1)	146'549 (2)	147'405
MICTREE	Subsection III.B [4]	StopWatch (Modulo-3600 Counter)	81	1296 (4)	168'311 (5)	169'607
MICTREE	Subsection III.B [4]	BioWatch (Modulo-24x3600 Counter)	129	2064 (4)	168'311 (5)	170'375
MICTREE	Subsection III.C [4] [5]	Random Number Generator	53	848 (4)	168'311 (5)	169'159
MICTREE	Subsection III.D [4] [9]	Specialized Turing Machine (Parenthesis Checker)	152	2432 (4)	168'311 (5)	170'743
MICTREE (3)	Subsection V.B	Universal Turing Machine (Asynchronous Counter)	510	8160 (4)	168'311 (5)	176'471
MUXTREE	Subsection IV.E [4]	Modulo-4 Up-Down Counter	0	0	120 (7)	120
MUXTREE	Subsection IV.F [4]	Shift Binary Decision Machine (Modulo-60 Counter)	36	360 (6)	12'000 (7) (8)	12'000 (9)
NEW MUXTREE (10)	Subsection V.A Subsection V.C	BioWatch 2001 (Modulo-24x3600 Counter)	128	1536 (11)	12'000 (7) (8)	12'000 (12)

BIODULE 600

(1) Instructions of the operative genome (OG) are 8 bits wide.

(2) The ribosomic genome (RG) is the configuration string necessary for programming an Actel 1020A FPGA, used to implement the BIODULE 600.

MICTREE

(3) This application is an ongoing work; final results may yet change.

(4) Instructions of the operative genome (OG) are 16 bits wide.

(5) The ribosomic genome (RG) is the configuration string necessary for programming an Actel 1020B FPGA, used to implement the MICTREE demonstration module.

MUXTREE and NEW MUXTREE

(6) Instructions of the operative genome (OG) are 10 bits wide.

(7) The number of bits of the ribosomic genome (RG) is the product of the number of molecules and the number of bits per molecule (20).

(8) Without spare molecules.

(9) Information of the operative genome (OG) is included in that of the ribosomic genome (RG), where the P value (bit MC2 of MOLCODE) determines the Q value of the flip-flop, i.e., the bit of the shift memory.

NEW MUXTREE

(10) Ongoing work; final results may yet change.

(11) Instructions of the operative genome (OG) are 12 bits wide.

(12) Information of the operative genome (OG) is included in that of the ribosomic genome (RG), where 8 to 16 bits per MOLCODE may be used to store 8 to 16 bits of the shift memory.

- 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

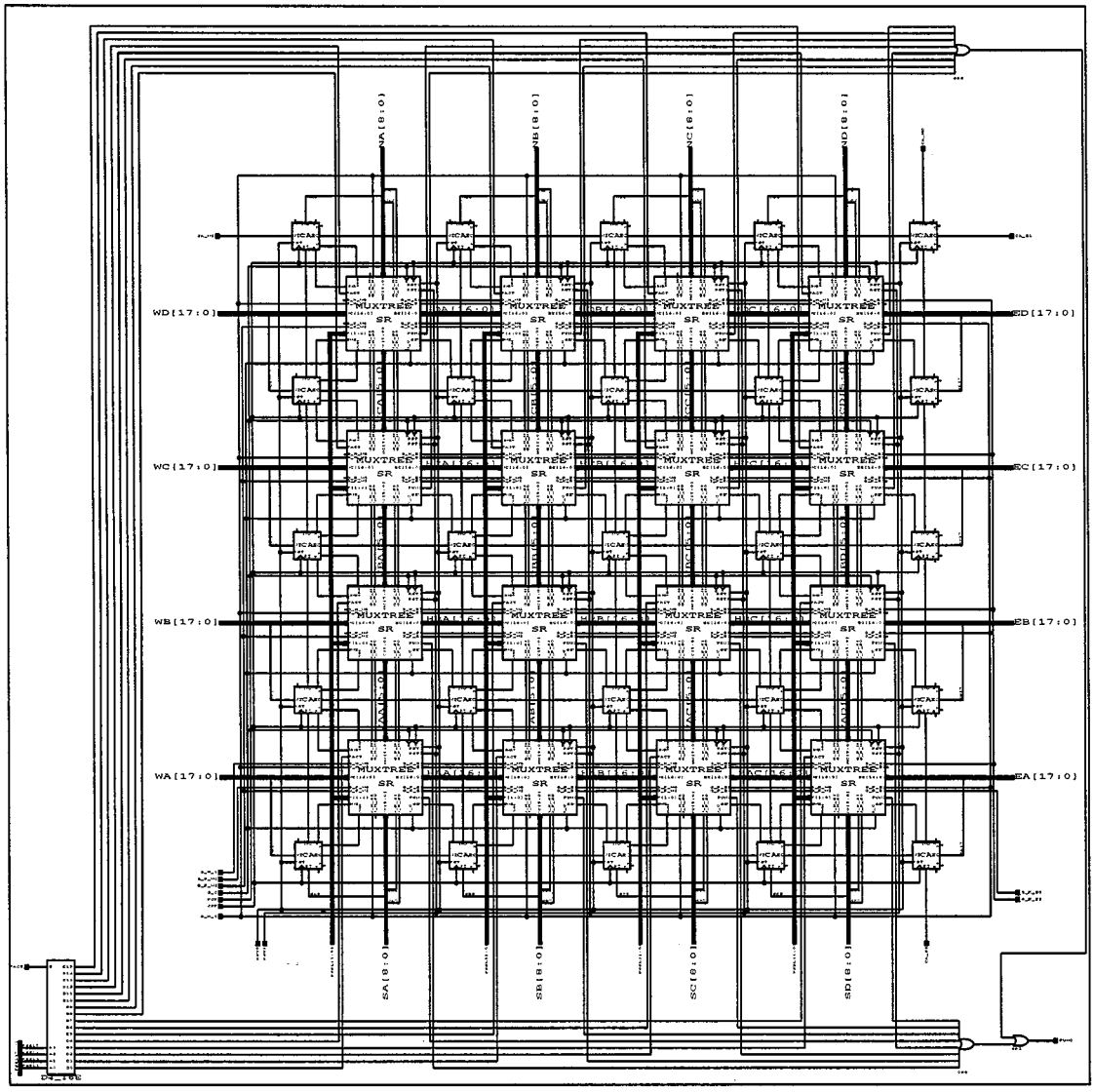


Fig. 28. Layout of a 4 × 4 array of MUXTREE molecules using a single Xilinx XC 4025 HQ 240 FPGA.

artificial genome configures a two-dimensional physical substrate—our FPGA—to generate the desired application:

$$\text{Genome} + \text{FPGA} = \text{Application}.$$

Table 1 presents experimental data concerning the genomes of the applications described in [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 FPGA's, 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 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

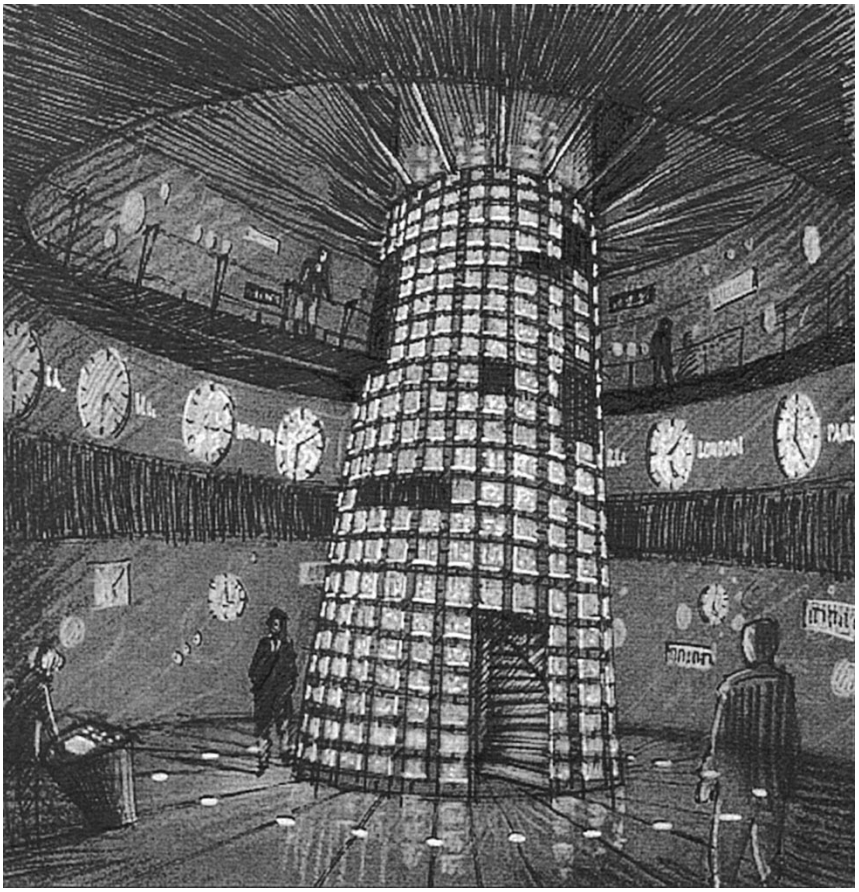


Fig. 29. An artist's rendition of a possible realization of the BioWatch 2001 [Art by Anne Renaud].

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.

One should 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 Section 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: Toward New Robust Integrated Circuits

Keeping in mind that our final objective is the development of 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 reprogram) 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 FPGA's 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. 22(a)], 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 4×4 array of our logic elements (Fig. 28). 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 25 000 logic gates (many more than those required by our 4×4 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 quasidefinitive 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 FPGA's of the 6200 family. In fact, this family of FPGA's, 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).

To the best of our knowledge, there exists today few projects, industrial or academic, which aim at integrating the properties of self-repair and/or of self-replication on FPGA's. 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 FPGA's (without self-replication) are also underway at NEC (Japan) [20] and at Altera (United States) [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. 29).

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*

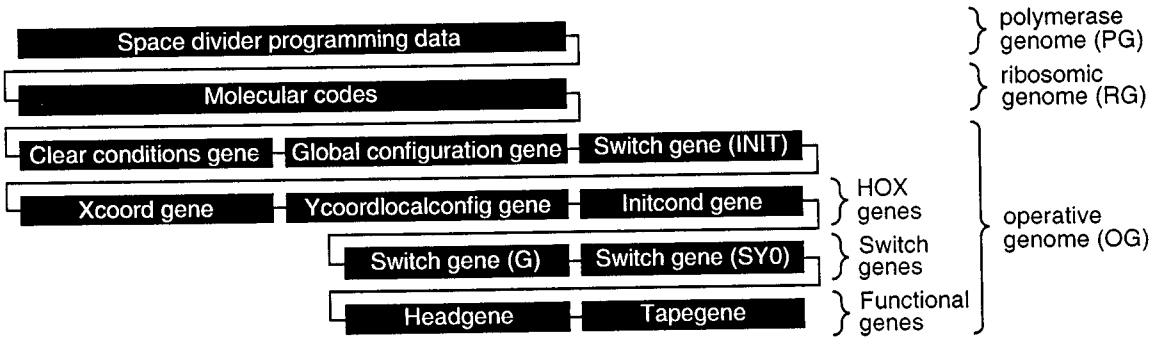


Fig. 30. The artificial genome of the parenthesis checker.

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 (Section III-D), the operative part of the genome consists of (Fig. 30): *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.

- 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].

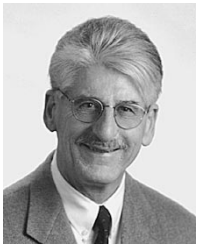
ACKNOWLEDGMENT

The authors are grateful to A. Renaud for her rendition of a possible realization of BioWatch 2001, to A. Herzog for the photographs [Figs. 14(b) and 22(b)], and to A. Badertscher for the implementation of the MUXTREE molecule and of the MICTREE cell.

REFERENCES

[1] L. Wolpert, *The Triumph of the Embryo*. New York: Oxford Univ. Press, 1991.
[2] M. Nicolaidis, "Future trends in online testing: A new VLSI design paradigm?," *IEEE Design Test Comput.*, vol. 15, no. 4, p. 15, 1998.
[3] D. Mange, *Microprogrammed Systems: An Introduction to Firmware Theory*, London, U.K.: Chapman & Hall, 1992.
[4] D. Mange and M. Tomassini, Eds., *Bio-inspired Computing Machines: Towards Novel Computational Architectures*, Lausanne, Switzerland: Presses Polytechniques et Universitaires Romandes, 1998.
[5] D. Mange, M. Goeke, D. Madon, A. Stauffer, G. Tempesti, and S. Durand, "Embryonics: A new family of coarse-grained field-programmable gate array with self-repair and self-reproducing properties," in *Toward Evolvable Hardware*, E. Sanchez and M. Tomassini, Eds, Berlin, Germany: Springer-Verlag, 1996, vol. 1062, pp. 197–220.
[6] S. Wolfram, *Theory and Applications of Cellular Automata*, Singapore: World Scientific, 1986.
[7] P. D. Hortensius, R. D. McLeod, and B. W. Podaima, "Cellular automata circuits for built-in self-test," *IBM J. Res. Develop.*, vol. 34, no. 2/3, pp. 389–405, 1990.
[8] J. von Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, Ed. Urbana, IL: Univ. of Illinois Press, 1966.
[9] D. Mange, D. Madon, A. Stauffer, and G. Tempesti, "Von Neumann revisited: A turing machine with self-repair and self-reproduction properties," *Robot. Auton. Syst.*, vol. 22, no. 1, pp. 35–58, 1997.

[10] M. L. Minsky, *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ: Prentice-Hall, 1967.
[11] E. R. Berlekamp, J. H. Conway, and R. K. Guy, "What is life?," in *Winning Ways for Your Mathematical Plays*. New York: Academic, 1982, pp. 817–850.
[12] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, vol. 27, pp. 509–516, June 1978.
[13] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary decision diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.
[14] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design.*, Berlin, Germany: Springer-Verlag, 1998.
[15] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet, "Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 387–399, Sept. 1998.
[16] G. Tempesti, "A self-repairing multiplexer-based FPGA inspired by biological processes," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 1998.
[17] G. Tempesti, D. Mange, and A. Stauffer, "Self-replicating and self-repairing multicellular automata," *Artif. Life*, vol. 4, no. 3, pp. 259–282, 1998.
[18] "A robust multiplexer-based FPGA inspired by biological systems," *J. Syst. Architecture*, vol. 43, no. 10, 1997.
[19] R. Negrini, M. G. Sami, and R. Stefanelli, *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays*. Cambridge, MA: MIT Press, 1989.
[20] A. Shibayama, H. Igura, M. Mizuno, and M. Yamashina, "An autonomous reconfigurable cell array for fault-tolerant LSIs," in *Proc. 44th IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, Feb. 1997, pp. 230–231 and 462.
[21] M. Sipper, "Fifty years of research on self-replication: An overview," *Artif. Life*, vol. 4, no. 3, pp. 237–257, 1998.
[22] M. A. Arbib, *Theories of Abstract Automata*. Englewood Cliffs, NJ: Prentice-Hall, 1969.
[23] P. Marchal, C. Piguet, D. Mange, A. Stauffer, and S. Durand, "Embryological development on silicon," in *Artificial Life IV*. Cambridge, MA: MIT Press, 1994, pp. 365–370.
[24] P. Nussbaum, B. Girau, and A. Tisserand, "Field programmable processor arrays," in *Evolvable Systems: From Biology to Hardware*, M. Sipper, D. Mange, and A. Perez-Uribe, Eds, Berlin, Germany: Springer-Verlag, 1998, vol. 1478, pp. 311–322.
[25] C. Ortega and A. Tyrrell, "MUXTREE revisited: Embryonics as a reconfiguration strategy in fault-tolerant processor arrays," in *Evolvable Systems: From Biology to Hardware*, M. Sipper, D. Mange, and A. Perez-Uribe, Eds, Berlin, Germany: Springer-Verlag, 1998, vol. 1478, pp. 206–217.
[26] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner, *Molecular Biology of the Gene*, 4th ed. Menlo Park, CA: Benjamin-Cummings, 1987.
[27] S. F. Gilbert, *Developmental Biology*, 3rd ed. Sunderland, U.K.: Sinauer Associates, Inc., 1991.
[28] D. Duboule, "The evolution of genomics," *Science*, vol. 278, p. 555, Oct. 24, 1997.
[29] S. Bentolila, "A grammar describing "biological binding operators" to model gene regulation," *Biochimie 78*, pp. 335–350, 1996.
[30] R. C. Merkle, "Making smaller, faster, cheaper computers," *Proc. IEEE*, vol. 86, pp. 2384–2386, Nov. 1998.
[31] C. F. Lane, S. T. Reddy, and B. I. Wang, "Means and apparatus to minimize the effect of silicon processing defects in programmable logic devices," U.S. Patent 5 592 102, Oct. 19, 1995.
[32] M. Barbieri, "The organic codes: The basic mechanism of macroevolution," *Rivista di Biologia/Biology Forum 91*, pp. 481–514, 1998.
[33] K. Roy, Ed., "A D&T roundtable: Online test," *IEEE Design Test Comput.*, vol. 16, no. 1, pp. 80–86, Jan.–Mar. 1999.
[34] Y. Zorian, "Testing the monster chip," *IEEE Spectrum*, vol. 36, no. 7, pp. 54–60, July 1999.
[35] G. D. Watkins, "Novel electronic circuitry," *Proc. IEEE*, vol. 86, p. 2383, Nov. 1998.
[36] P. Kuekes, "Molecular manufacturing: Beyond moore's law," in *Proc. Field-Programmable Custom Computing Machines (FCCM'99)*, Napa, CA, Apr. 1999.
[37] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, no. 5370, pp. 1716–1721, June 12, 1998.
[38] R. F. Service, "Organic molecule rewires chip design," *Science*, vol. 285, no. 5426, pp. 313–315, July 16, 1999.
[39] S. R. Park and W. Burleson, "Configuration cloning: Exploiting regularity in dynamic DSP architectures," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays (FPGA'99)*, Monterey, CA, Feb. 1999, pp. 81–89.
[40] C. Ortega and A. Tyrrell, "Reliability analysis in self-repairing embryonic systems," in *Proc. 1st NASA/DOD Workshop Evolvable Hardware*, Pasadena, CA, July 1999, pp. 120–128.
[41] ——, "Self-repairing multicellular hardware: A reliability analysis," in *Advances in Artificial Life*, D. Floreano, J.-D. Nicoud, and F. Mondada, Eds., Berlin, Germany: Springer-Verlag, 1999, vol. 1674, pp. 442–446.
[42] R. Gordon, *The Hierarchical Genome and Differentiation Waves*. Singapore and London, U.K.: World Scientific and Imperial College, 1999.
[43] D. Mange, M. Sipper, and P. Marchal, "Embryonic electronics," *BioSystems*, vol. 51, no. 3, pp. 145–152, 1999.
[44] M. Sipper, D. Mange, and E. Sanchez, "Quo vadis evolvable hardware?," *Commun. ACM*, vol. 42, no. 4, pp. 50–56, Apr. 1999.
[45] E. Sanchez, M. Sipper, J.-O. Haenni, J.-L. Beuchat, A. Stauffer, and A. Perez-Uribe, "Static and dynamic configurable systems," *IEEE Trans. Comput.*, vol. 48, pp. 556–564, June 1999.

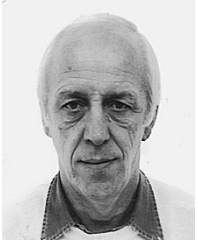


Daniel Mange (S'68–M'69) received the M.S. and Ph.D. degrees from the Swiss Federal Institute of Technology, Lausanne, Switzerland.

Since 1969, he has been a Professor at the Swiss Federal Institute of Technology. He was a Visiting Professor at the Center for Reliable Computing, Stanford University, Stanford, CA, in 1987. He is Director of the Logic Systems Laboratory, and his chief interests include firmware theory (equivalence and transformation between hardwired systems and programs), cellular automata, artificial life, and embryonics (embryonic electronics). He is an author or coauthor of several scientific papers in these areas, as well as of *Microprogrammed Systems: An Introduction to Firmware Theory* (London, U.K.: Chapman & Hall, 1992). He was Program Cochairman of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), held in Tsukuba, Japan, and General Chairman of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), held in Lausanne in September 1998.

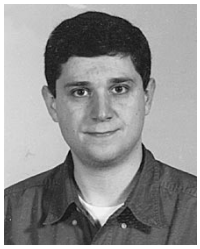
Moshe Sipper (S'87–M'89–SM'98) received the B.A. degree in computer science from the Technion—Israel Institute of Technology and the M.Sc. and Ph.D. degrees from Tel Aviv University, Israel.

He is a Senior Researcher in the Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland. His chief interests involve the application of biological principles to artificial systems, including evolutionary computation, cellular computing, bio-inspired systems, evolvable hardware, complex adaptive systems, artificial life, and neural networks. He has published more than 70 research publications in these areas, as well as the book *Evolution of Parallel Cellular Machines: The Cellular Programming Approach* (Heidelberg, Germany: Springer-Verlag, 1997). He was Program Chairman of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), held in Lausanne in September 1998.



André Stauffer (S'68–M'69) received the Diploma in electrical engineering and the Ph.D. degree from the Swiss Federal Insitutute of Technology, Lausanne, Switzerland.

He spent one year as a Visiting Scientist at the IBM T.J. Watson Research Center, Yorktown Heights, NY, in 1986. He is a Senior Lecturer in the Department of Computer Science, Swiss Federal Insutitute of Technology. In addition to digital design, his research interests include circuit reconfiguration and bio-inspired systems.



Gianluca Tempesti (S'92–M'98) received the B.S.E. degree in computer engineering from Princeton University, Princeton, NJ, in 1991, the M.S.E. degree from the University of Michigan, Ann Arbor, in 1993, and the Ph.D. degree from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1998, with a thesis based on the development of a self-repairing multiplexer-based FPGA.

Since 1994, he has been a Teaching and Research Assistant with the Logic Systems Laboratory, Department of Computer Science, Swiss Federal Institute of Technology. His research interests include self-test and self-repair, programmable logic circuits, processor design, and parallel computer architecture.