IN: RECONFIGURABLE ARCHITECTURES: HIGH PERFORMANCE BY CONFIGWARE
IT PRESS-VERLAG, BRUCHSAL, 1997, PP. 45-48

FPPA: A Field-Programmable Processor Array with Self-Repair and Self-Reproducing Properties

Daniel MANGE, Maxime GOEKE, Dominik MADON, André STAUFFER, Gianluca TEMPESTI
The Swiss Federal Institute of Technology - CH 1015 LAUSANNE (Switzerland)
Phone: (+41 21) 693 26 39 Fax: (+41 21) 693 37 05
e-mail: mange@di.epfl.ch

Serge DURAND, Pierre MARCHAL, Pascal NUSSBAUM
Centre suisse d'électronique et de microtechnique SA - CH 2000 NEUCHATEL (Switzerland)
Phone: (+41 32) 720 56 62 Fax: (+41 32) 720 57 63
e-mail: marchal@csemne.ch

ABSTRACT

The growth and the operation of all living beings are directed through the interpretation, in each of their cells, of a chemical program, the DNA. This program, called *genome*, is the blueprint of the organism and consists of a sequence of four discrete characters: A, C, G, and T. This process is the source of inspiration for the Embryonics (embryological electronics) project, whose final objective is the conception of very large scale integrated circuits endowed with properties usually associated with the living world: self-repair (cicatrization) and self-reproduction. Within this framework, we will present a family of field-programmable processor arrays. Each cell is a binary decision machine whose microprogram represents the genome, and each part of the microprogram is a gene whose execution depends on the physical position of the cell in the array, i.e. on its coordinates. The considerable redundancy introduced by the presence of a genome in each cell has significant advantages: self-reproduction (the automatic production of one or more copies of the original organism) and self-repair (the automatic repair of one or more faulty cells) become relatively simple operations. Even if the described system seems exceedingly complex, we believe that computer architectures inspired by molecular biology will allow the development of FPPAs endowed with quasi-biological properties extremely useful in environments where human intervention is necessarily limited (nuclear plants, space applications, etc.).

1. INTRODUCTION

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 processes: the chemical 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). Each group of three bases is decoded in the cell to produce a particular amino acid, a future constituent of the final protein (thus, the triplet ACG will produce threonine [1]).

Our research is inspired by the basic processes of molecular biology [2]. 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-reproduction and self-repair, can also be applied to artificial objects (integrated circuits). Our final objective is the development of very large scale integrated circuits (wafer scale integration) capable of self-repair and self-reproduction. These two properties, characteristic of the living world, seem particularly desirable for very complex artificial systems meant for hostile (nuclear plants) or inaccessible (space) environments. Self-reproduction allows the complete reconstruction of the original device in case of a major fault, while self-repair allows a partial reconstruction in case of a minor fault.

2. EMBRYONICS

Embryonics (embryological electronics), i.e. the quasi-biological development of multi-cellular automata, is based on a *general hypothesis*, which describes the environment in which the development occurs, and on *three features*, which roughly approximate the biological mechanism of cellular development [3], [4].

2.1 The general hypothesis: the environment

The general hypothesis describes the selected environment in which the quasi-biological development occurs. In the framework of electronics, it consists of a finite (but as large as desired) two-dimensional space of silicon, divided into rows and columns (the third dimension introduced by the physical realization of the system is irrelevant for our purposes). The intersection of a row and a column defines a *cell*, and all cells have an identical physical structure, i.e. an identical network of connections (wires) and an identical set of operators

(combinational and sequential logic operators). The physical space or *cellular array* is therefore *homogeneous*, that is, made up of absolutely identical cells: only the *state* of a cell, that is, the combination of the values in its memories, can differentiate it from its neighbors.

The transformations, in particular the development of an artificial multicellular organism from a single cell, the mother cell, occur then without supply of material (the network of silicon is present at the start), with a trivial supply of energy (the classical power supply of an electronic system), and with a crucial supply of information.

2.2 First feature: multicellular organization

The first feature is that of *multicellular organization:* the artificial organism is divided into a finite number of cells, where each cell realizes a unique function, described by a sub-program called the *gene* of the cell. The same organism can contain multiple cells of the same kind (in the same way as a living being can contain a large number of cells with the same function: nervous cells, skin cells, liver cells, etc.). In this presentation, for clarity's sake, we will confine ourselves to a simple example of a 1-dimensional artificial organism (Fig. 1): a *modulo 3600 counter*, implemented with four cells and featuring two distinct genes (a modulo 10 counter for units of seconds and minutes, a modulo 6 counter for tens of seconds and minutes).

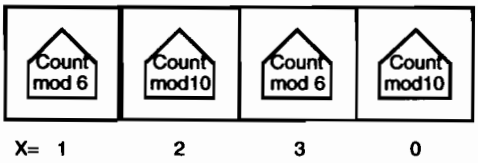

Fig. 1. A 1-dimensional non-uniform multicellular organism.

2.3 Second feature: cellular differentiation

Let us call *genome* the set of all the genes of an artificial organism, where each gene is a sub-program characterized by a set of instructions and by its position (its coordinates X,Y). Fig. 1 then shows the genome of the modulo 3600 counter, with the corresponding horizontal (X) coordinate. Let then each cell contain the entire genome (Fig. 2): depending on its position in the array, i.e., its place in the organism, each cell can interpret the genome and extract and execute the gene which configures it. In summary, storing the whole genome in each cell makes the cell universal: it can realize any gene of the genome, given the proper coordinates.

2.4 Third feature: cellular division

At startup, the mother cell or *zygote* (Fig. 2), arbitrarily defined as having the coordinate X = 1, holds the one and only copy of the genome. At time t+1, the genome of the mother cell is copied into the neighboring (daughter) cell to the east. The process then continues until the 1-dimensional space is completely programmed. In our example, the furthest cell is programmed at time t+3. In the more general case of 2-dimensional arrays, the genome of the mother cell is copied into the two neighboring daughter cells to the north and to the east, and so on until the 2-dimensional space is completely programmed.

3. EXAMPLE: A MODULO 3600 COUNTER

Our artificial organism is specified to count seconds (from 00 to 59) and minutes (from 00 to 59); such a counter is therefore a modulo 3600 counter. We choose the architecture of Fig. 3, in which the counter is divided into four partial counters: two modulo 10 counters performing the unit counting (seconds or minutes), two modulo 10 counters performing the ten counting (seconds or minutes).

In order to demonstrate the properties of self-reproduction and self-repair, we have chosen to add four spare cells at the right of the counter. The modulo 3600 counter is *locally synchronous*; the global clock of each partial counter comes from one of the outputs of the counter to its right (Q2 or Q3) or from the reference clock pulse G, with a 1 Hz frequency.

3.1 Computing the gene for the modulo 10 counter

In all human beings, the chain of characters which makes up the DNA is executed sequentially by a chemical processor, the *ribosome*. Drawing inspiration from this biological mechanism, we use a microprogram to compute first each of the genes of the artificial organism, then its coordinates, and finally its complete genome.

In this paper, we will only demonstrate how to establish the sub-program corresponding to the gene for the modulo 10 counter. The use of Karnaugh map for simplifying trees [5], [6] shows ten *blocks* (i.e. patterns formed by 2^m adjacent cells) containing each the same next state (Fig. 4; the Φ symbol describes a don't care condition). We obtain therefore a *simplified binary decision tree* (Fig. 5) having ten branches corresponding to the ten possible next states of Q3:0+, from 0000 to 1001. For a microprogrammed realization, the binary decision tree of Fig. 5 is the flowchart for the gene of the modulo 10 counter. The software implementation of this flowchart (Fig. 6) requires *test instructions* and *assignment instructions* whose mnemonics are:

- **if** VAR **else** LABEL
- **do REG** = DATA

The *non-conditional jump* is a particular case of the test instruction where the test variable is the logic constant 0. Its mnemonic is simply:

- **goto** LABEL

The sub-program (or gene) **Countmod10** describing the modulo 10 counter is then written by means of the mnemonics and of the labels C8, C70, C6, C54, C4, C30, C2, C10, C0 and End.

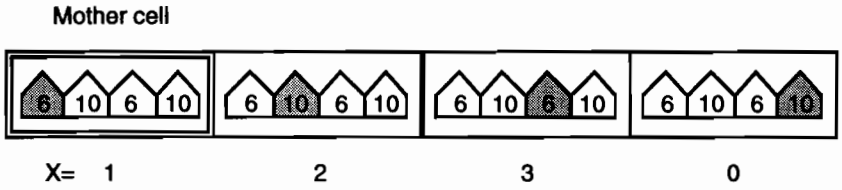

Fig. 2. The 1-dimensional organism with a genome in every cell.

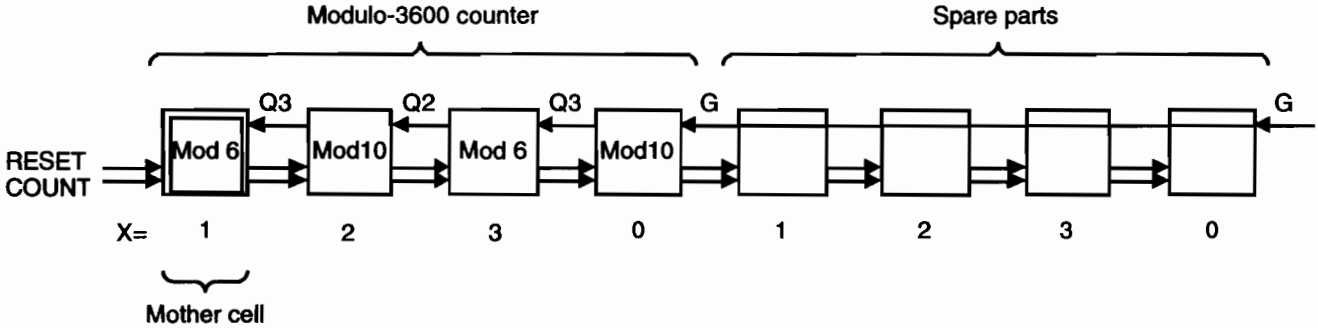

Fig. 3. A 4-cell (X=1...0) modulo 3600 counter with 4 spare cells.

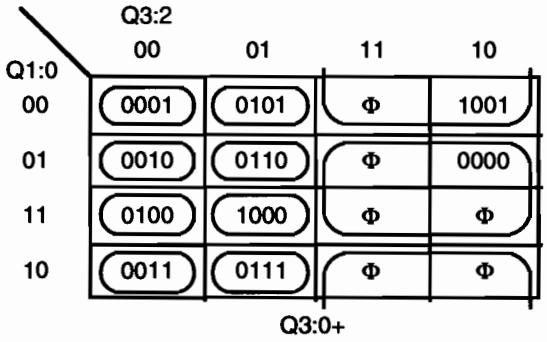

Fig. 4. Computing the modulo 10 counter: Karnaugh map.

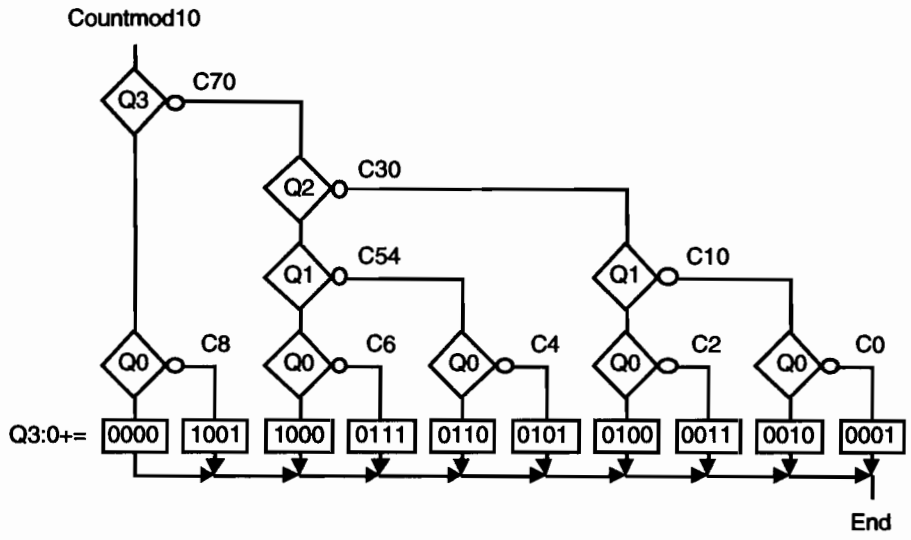

Fig. 5. Computing the modulo 10 counter: flowchart of the gene **Countmod10**.

```
Countmod10: if Q3 else C70
            if Q0 else C8
            do REG =0
            goto End
C8:         do REG =9
            goto End
C70:        if Q2 else C30
            if Q1 else C54
            if Q0 else C6
            do REG =8
            goto End
C6:         do REG =7
            goto End
C54:        if Q0 else C4
            do REG =6
            goto End
C4:         do REG =5
            goto End
C30:        if Q1 else C10
            if Q0 else C2
            do REG =4
            goto End
C2:         do REG =3
            goto End
C10:        if Q0 else C0
            do REG =2
            goto End
C0:         do REG =1
End:        ...
```

Fig. 6. Computing the modulo 10 counter: sub-program of the gene **Countmod10**.

3.2 A field-programmable processor array based on a binary decision machine

While our long-term objective is the conception of very large scale integrated circuits, we started by realizing a demonstration system in which each cell, called *MICROTREE* (for tree of microinstructions), is embedded into a plastic container called *BIODULE 601* (Fig. 7) [7], [8].

The MICROTREE cell consists essentially of a *binary decision machine* [5], executing the microprograms written using the following set of instructions:

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

The first three instructions were introduced above in the computing of the modulo 10 counter and the two last are dedicated to the computing of the coordinates of the cell. In this implementation, the state register REG and both coordinate registers are 4 bits wide (REG3:0, X3:0, and Y3:0).

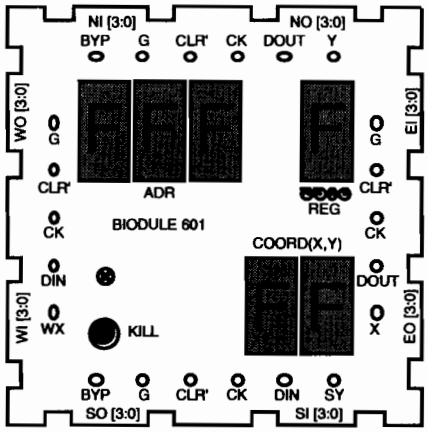

Fig. 7. BIODULE 601: demonstration module including a MICROTREE cell.

Each MICROTREE cell (Fig. 8) has four neighbors (to the south, west, north, and east). Four 4-bit busses enter the cell from its neighbors (SI3:0 from the south, WI3:0 from the west, NI3:0 from the north, and EI3:0 from the east) and, correspondingly, four output busses go out in the four cardinal directions (SO3:0 to the south, WO3:0 to the west, NO3:0 to the north, and EO3:0 to the east).

Each MICROTREE cell has, therefore, 16 outputs SO3...EO0. Each of these outputs can be programmed to take a value from one of 16 possible sources (Fig. 9). For example, output NO3 can take one of the following 16 values:

- the four bits REG3:0 of register REG;
- the four bits SI3:0 of the south input bus SI;
- the four bits WI3:0 of the west input bus WI;
- the four bits EI3:0 of the east input bus EI.

Note that it is impossible for NO3 to get the value of one of the four bits NI3:0 of the input bus corresponding to the same cardinal direction.

In our assembly language, a single assignment instruction is sufficient to perform this operation. The mnemonic for this instruction is:

- **do** VAROUT = VARIN

The variables VAROUT correspond to the four cardinal output busses, for a total of 16 bits (Fig. 8):

- VAROUT $\in$ {SO3:0, WO3:0, NO3:0, EO3:0}

while the variables VARIN correspond to the four cardinal input busses and the register REG, for a total of 20 bits:

- VARIN $\in$ {SI3:0, WI3:0, NI3:0, EI3:0, REG3:0}

remembering that VAROUT and VARIN can never refer to the same cardinal direction (Fig. 9).

The test variables VAR include the set VARIN and the following additional variables:

- VAR $\in$ {VARIN, WX3:0, SY3:0, G}

where WX is the coordinate of the next cell to the west, SY is the coordinate of the next cell to the south and G is a global variable, usually reserved for the synchronization clock.

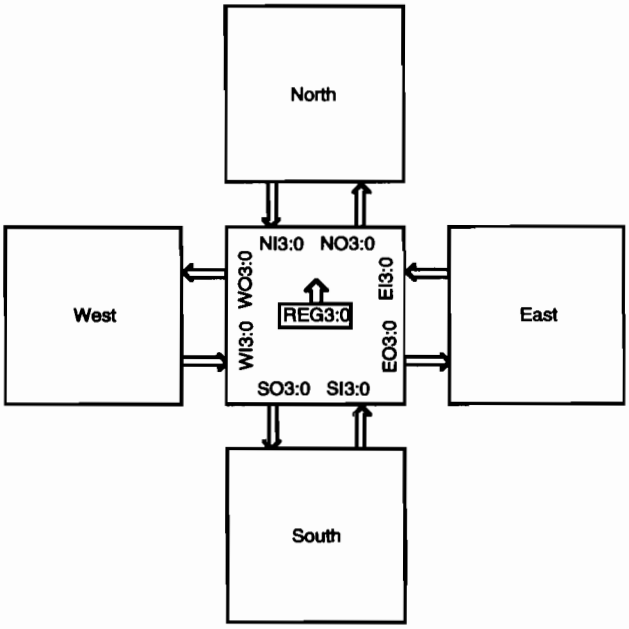

Fig. 8. MICROTREE cell: four neighboring cells connection diagram.

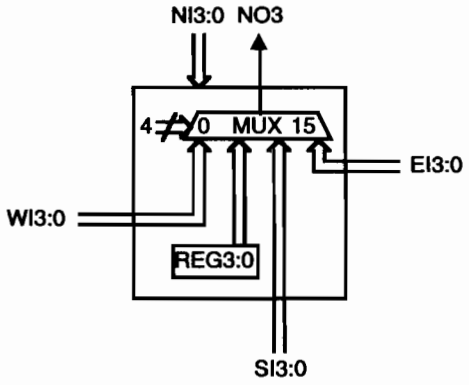

Fig. 9. MICROTREE cell: output variable NO3 example.

The coordinates are transmitted from cell to cell serially, but are computed in parallel. Therefore, each cell performs a series-to-parallel conversion on the incoming coordinates WX and SY of the western and southern neighbors respectively, and a parallel-to-series conversion of the coordinates X and Y it computes and propagates. By default (that is, with the external connections WX and SY not connected), the mother cell recognizes the values WX=SY=0.

The genome microprogram is also coded serially. It enters through the DIN pin of the mother cell and is then propagated through its DOUT pin, according to the cellular division path determined by the user (Fig. 7).

The pins CK and CLR' are used for the propagation of the clock signal and for the reset of the binary decision machine, while the signal BYP (bypass), connecting all the cells of a column, is used for self-repair.

The size of the artificial organism embedded in an array of MICROTREE 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), 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 [7], [8] 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.

3.3 Self-reproduction

The self-reproduction of an artificial organism, for example the modulo 3600 counter of Fig. 1, rests on two hypotheses: (1) there exists a sufficient number of spare cells (unused cells at the right hand side of the array, at least four for our example) and (2) the calculation of the coordinates produces a cycle (X=1->2->3->0 in Fig. 10).

As the same pattern of coordinates produces the same pattern of genes, self-reproduction can be easily accomplished if the microprogram of the genome, associated to the homogeneous network of cells, produces several occurrences of the basic pattern of coordinates (X=1->2->3->0). In our example, the repetition of the horizontal coordinate pattern, i.e. the production of the pattern X=1->2->3->0->1->2->3->0 (Fig. 10), produces one copy, the *daughter counter*, of the original or *mother counter*. Given a sufficiently large space, the self-reproduction process can be repeated for any number of specimens, both in the X and the Y axes.

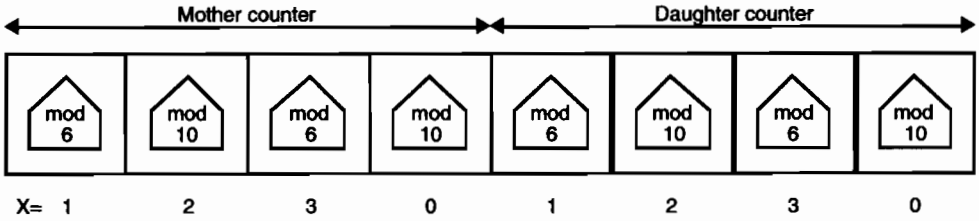

Fig. 10. Self-reproduction of the 4-cell modulo 3600 counter in a 8-BIODULE array.

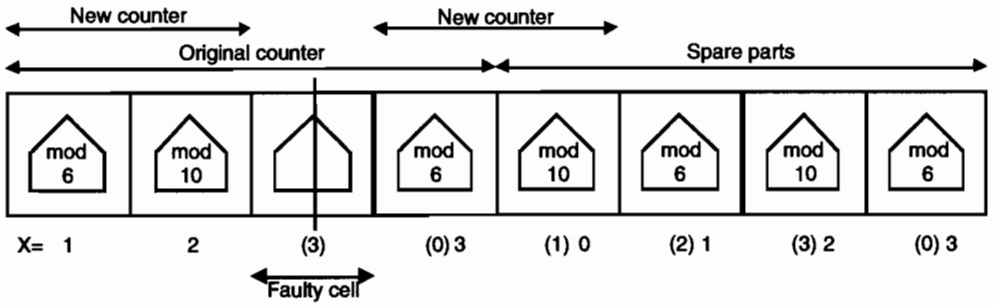

Fig. 11. Self-repair of the 4-cell modulo 3600 counter in a 8-BIODULE array.

3.4 Self-repair

In the BIODULE 601 (Fig. 7), the existence of a fault is decided by the human user by pressing the KILL button of a cell. Therefore, fault detection and fault location, two features which will be indispensable in the final system, where they will be implemented using BIST (Built-In Self-Test) techniques [9], [10], [11], are not present in the BIODULE 601.

In order to implement self-repair, we have chosen, favoring simplicity, the following process (Fig. 11):

- pressing the KILL button identifies the faulty cell (the KILL light, normally green, becomes red);
- the entire column (in the general case) to which the faulty cell belongs is considered faulty, and is desactivated (column X=3 in Fig. 11);
- all the functions of the MICROTREE cell are shifted by one cell (or, in the general case, by one column) to the right.

Obviously, this process requires as many spare cells (or columns), to the right of the array, as there are faulty cells to repair (four spare cells in the example of Fig. 11). It also implies some modifications to the MICROTREE cell, so as to add the capability of bypassing the faulty cell and shifting to the right all or part of the original cellular array.

4. CONCLUSION

The main result of our research is the development of a family of FPPAs called MICROTREE and based on a binary decision machine capable of executing a microprogram of up to 1024 instructions. The original features of this FPPA are essentially:

- a completely homogenous organization of the cellular array;
- an integration of the routing into each cell, both for the short- and long-distance (bus) connections;
- a sequential execution of microprograms methodically derived from a chosen representation, the binary decision tree or diagram.

Our FPPA satisfies the general hypothesis about the environment (Section 2.1), as well as the three features of the Embryonics project: multicellular organization, cellular differentiation, and cellular division (Sections 2.2 to 2.4). The MICROTREE cell, itself realized with a commercial FPGA and a RAM, was finally embedded into a demonstration module called BIODULE 601, and we showed that an array of BIODULES 601 is capable of self-repair and self-reproduction.

The trivial applications of the MICROTREE 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 (Wolfram's) and two-dimensional (life, Langton's loop, etc.) uniform cellular automata are natural candidates for this kind of realization.

The non-trivial applications are those in which the cells of an array have different genes: the genome is then a collection 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 random number generator described in [4] or like the present modulo 3600 counter) and two-dimensional (like a Turing machine [12]) non-uniform cellular automata fall within this category.

Let us also mention that the realization of uniform cellular automata with a pre-determined initial state is an important special case which also requires separate genes and a coordinate system. The classic example of the cellular realization of a Turing machine [12], with a program stored on a tape, represents an application of this kind.

REFERENCES

[1] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. Argetsinger Steitz and A. M. Weiner, *Molecular Biology of the Gene, Fourth Edition.* Menlo Park: The Benjamin/Cummings Publishing Company, 1987.

[2] R. Ransom, *Computers and Embryos.* Chichester: John Wiley, 1981.

[3] D.Mange, A.Stauffer, "Introduction to Embryonics: Towards New Self-repairing and Self-reproducing Hardware Based on Biological-like Properties", *Artificial Life and Virtual Reality.* Chichester: John Wiley, 1994, pp. 61-72.

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

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

[6] D. Mange, "Teaching firmware as a bridge between hardware and software", *IEEE Trans. Education,* vol. 36, no. 1, pp. 152-157, 1993.

[7] M. Goeke, "BIODULE 2: documentation technique", Tech. Rep., Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, 1995.

[8] D. Madon, "BIODULE 2: description et utilisation", Tech. Rep., Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, 1995.

[9] M. Abramovici and C. Stroud, "No-overhead BIST for FPGAs," in *Proc. 1st IEEE International On-Line Testing Workshop*, July 1995, pp. 90-92.

[10] S. Durand and C. Piguet, "FPGA with self-repair capabilities," in *Proc. FPGA '94, 2nd International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, February 1994, pp. 1-6.

[11] E. J. McCluskey, *Logic Design Principles with Emphasis on Testable Semicustom Circuits*. Englewood Cliffs: Prentice Hall, 1986.

[12] D. Mange, D. Madon, A. Stauffer and G. Tempesti, "Von Neumann Revisited: A Turing Machine with Self-Repair and Self-Reproduction Properties", Tech. Rep. 96/180, Computer Science Department, Swiss Federal Institute of Technology, Lausanne, March 1996 (submitted).