PROC. INT'L WORKSHOP ON LOGIC AND ARCHITECTURE SYNTHESIS
INP GRENOBLE, DEC. 1996, PP. 103-112

MICROTREE: Towards a Binary Decision Machine-based FPGA with Biological-like Properties

André STAUFFER, Daniel MANGE, Maxime GOEKE, Dominik MADON, 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: stauffer@di.epfl.ch

Serge DURAND, Pierre MARCHAL, Christian PIGUET
Centre suisse d'électronique et de microtechnique SA - CH 2000 NEUCHATEL (Switzerland)
Phone: (+41 38) 205 662 Fax: (+41 38) 205 630
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. This program, called genome, is the blueprint of the organism and its interpretation is the source of inspiration for the Embryonics (embryological electronics) project. The final objective of the project is the conception of a wafer scale FPGA endowed with self-reproduction and self-repair properties. In this circuit, 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. Thanks to the considerable redundancy introduced by the presence of a genome in each cell, 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. Both will be illustrated by a classical example of a specialized Turing machine: a parenthesis checker. Even if the described FPGA seems exceedingly complex, we believe that such circuits endowed with biological-like properties will be 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 ($2x10^{9}$) 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.

Our research, inspired by this basic process of molecular biology [1], adopts certain features of cellular organization and transposes them to the 2-dimensional world of integrated circuits on silicon. Properties unique to the living world, such as self-reproduction and self-repair, are thus applied to artificial objects (integrated circuits). Our final objective is the development of a wafer scale FPGA capable of self- reproduction and self-repair. These two biological-like properties seem particularly desirable for 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, i.e. the quasi-biological development of multicellular automata, is based on a *general hypothesis*, which describes the environment in which the development occurs, and on *three characteristics*, which roughly approximate the biological mechanism of cellular development [2], [3].

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)

2-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.

2.2 The three characteristics

The first characteristic is that of *multicellular organization*. The artificial organism is divided into a finite number of cells (Fig. 1), 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, we will confine ourselves to a simple example of a 2-dimensional artificial organism (Fig. 1): a *specialized Turing machine* detailed later. This machine is implemented with ten cells and features two distinct genes, the *tape* gene and the *head* gene. Each cell is associated with some initial condition; in our example the head cells are distinguished by the initial values "0" and "→", the tape cells by "A", "(", and ")" values.

The second characteristic is that of *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, by an initial condition, and by a position (its coordinates X,Y). Fig. 1 then shows the genome of our Turing machine, with the corresponding horizontal (X) and vertical (Y) coordinates. Let then each cell contain the entire genome or global program (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 (with its initial condition) which configures it.

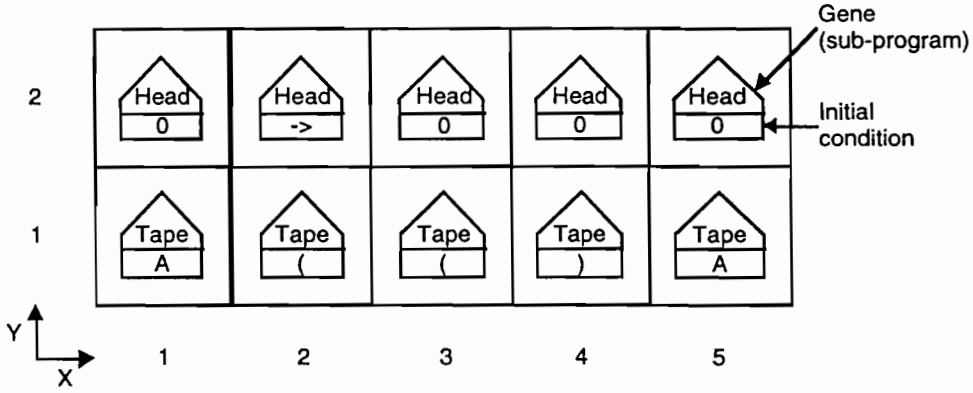

Fig. 1

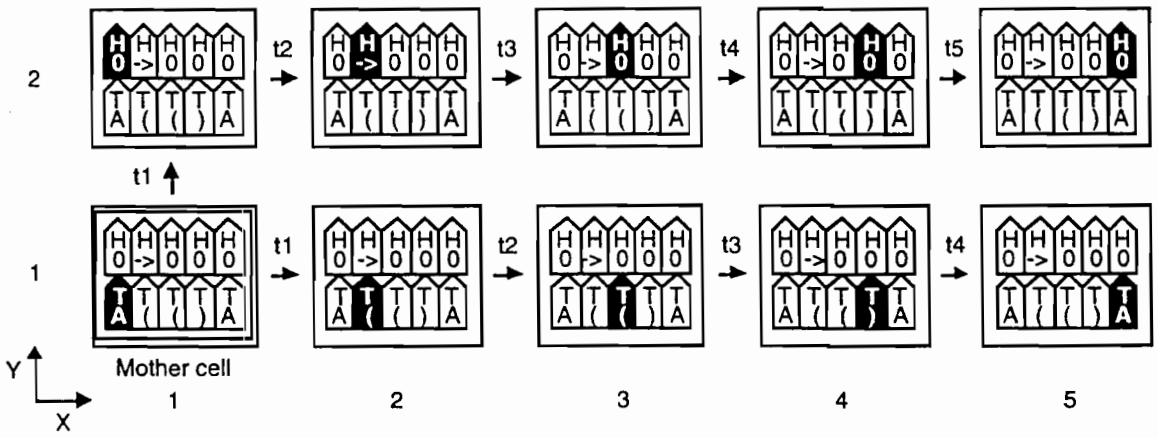

Fig. 2

The third characteristic is that of *cellular division*. At startup, the mother cell or *zygote* (Fig. 2), arbitrarily defined as having the coordinate X,Y = 1,1, holds the one and only copy of the genome. At time t1, the genome of the mother cell is copied into the two neighboring (daughter) cells to the north and to the east. The process then continues until the 2-dimensional space is completely programmed. In our example, the furthest cell is programmed at time t5.

3. EXAMPLE: A PARENTHESIS CHECKER

According to Minsky [4] (pp. 121-123), the problem is to decide whether a sequence of left (open) and right (closed) parentheses is well-formed. A good procedure for checking parentheses consists of searching to the right for a right (closed) parenthesis, then searching to the left its mate (open), and removing both. This procedure is repeated until no more pairs are found. If any unmatched symbols remain, the expression is not well-formed, and conversely.

3.1 Turing machine realization

A specialized Turing machine for checking parentheses consists of a tape, decomposed in squares, and a finite-state machine with a read/write head. For checking the expression "(()", for example, we prepare the tape in the form of Fig. 3, where the beginning and end of the expression are marked by "A" symbols. The read/write head starts by reading the first left parenthesis (Fig. 3). The initial state of the finite state-machine (Fig. 4) is the "→" state. For each couple H,T (*head state, tape state*), the state table of Fig. 4 produces a new couple H+,T+ giving the next head state H+, belonging to the set {→, 0←, 1←, ↑} and the next tape state T+, belonging to the set {X, (, A,), 0, 1}.

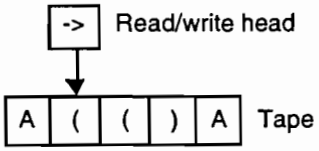

Fig. 3

H+,T+ (H) \ T	X	(	A	)	0	1
->	->,X	->,(	1<-,A	0<-,X	-,-	-,-
0<-	0<-,X	->,X	↑,0	0<-,)	-,-	-,-
1<-	1<-,X	↑,0	↑,1	-,-	-,-	-,-
↑	-,-	-,-	-,-	-,-	↑,0	↑,1

Fig. 4

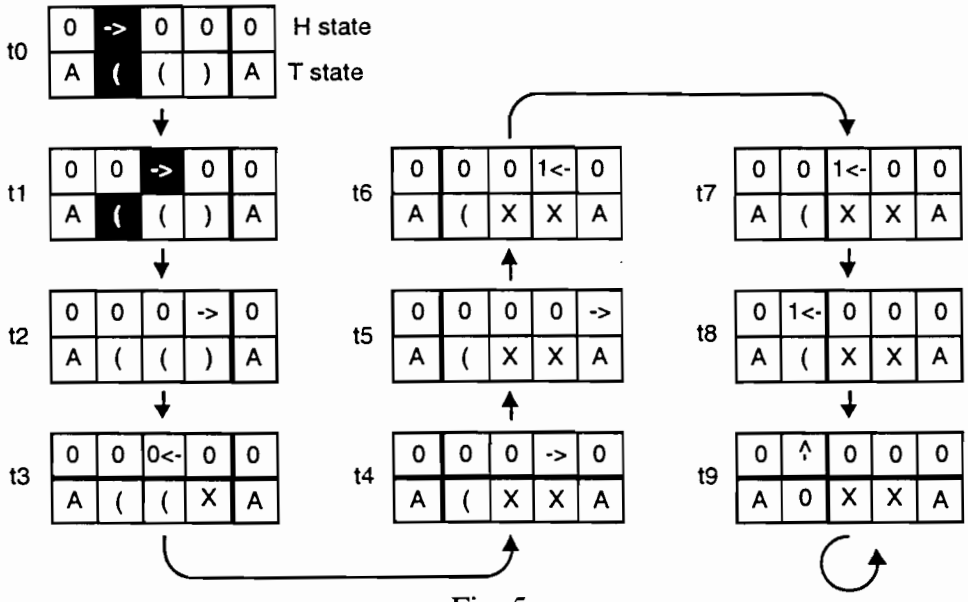

Fig. 5

In Fig. 5 at time t0, the initial couple H,T is defined as H,T = →,(and produces a next couple H+,T+ = →,(at time t1. The read/write head moves one square to the right and displays the next head state H+ = → in the new square. The square of the tape read by the head at time t0 displays the next tape state T+ = (and remains thus unchanged. For the moment, we assume that the read/write head can move and has, at a given time, one single position. All the squares with H=0 are, therefore, without any physical significance. This process continues until time t9, when no

more changes are possible. After going through the head states "→" (move to the right), "0←" (move to the left, 1st type), "1←" (move to the left, 2nd type), the finite-state machine reaches finally the "↑" state (hold). During this process, the read/write head has replaced on the tape all the left "(" and right ")" parentheses by the symbol "X". As the expression "(()" is not well-formed, the last symbol written on the tape by the head is "0". Conversely, if the expression were well-formed, the last symbol would be "1".

3.2 Cellular implementation

In order to obtain a cellular implementation for our parenthesis checker which would be compatible with the homogeneous cellular space defined above, we will realize our Turing machine as in Fig. 6. The cellular space is divided in two rows, identified by the vertical coordinate Y (Y=1 or 2), and by N columns, identified by the horizontal coordinate X (X=1...N). In our example, defined by Fig. 3, N is equal to 5. In order to demonstrate self-repair, we have decided to add two spare cells in each row, to the right of the Turing machine, all identified by the same horizontal coordinate N+1 (X=6 in our example).

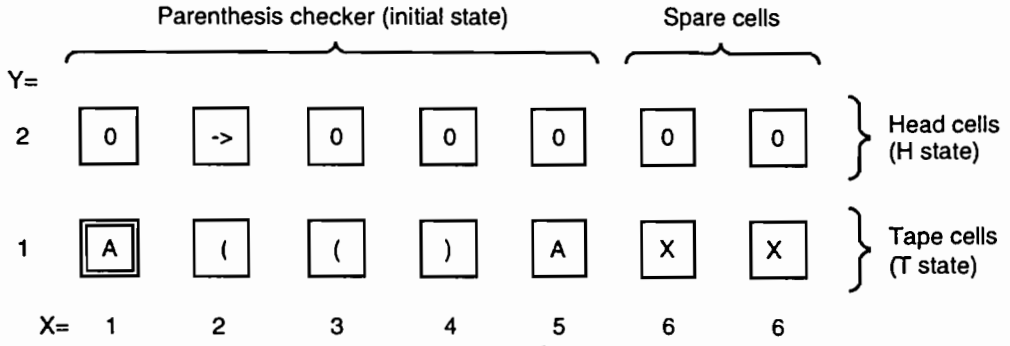

Fig. 6

The upper row, identified by Y=2, realizes the moving read/write head; all the cells of this row are in a new *quiescent head state* (H=0), except for one cell which implements the moving head with a non-quiescent state (H = "→", "0←", "1←" or "↑"). The lower row, identified by Y=1, realizes the conventional tape. At the start, the checker cells of this row are in one of the three tape states T = "A", "(" or ")" and the spare cells are in the *quiescent tape state* (T=X). At the end of the computation process, all the cells in the "(" and ")" states are replaced by "X" state, except for the unique cell giving the final result (T=1 for a well-formed expression, T=0 in the contrary).

3.3 Tape gene computing

In all living beings, the string of characters which makes up the genome is executed sequentially by a chemical processor, the *ribosome*. Drawing inspiration from this biological mechanism, we will use a microprogram to compute first the coordinates of the artificial organism, then the initial conditions of each cell, the tape gene and the head gene, and finally the complete genome.

In this paper, we will only demonstrate how to establish the sub-program corresponding to the tape gene of the parenthesis checker. The use of Karnaugh map for simplifying trees [5], [6] shows nine *blocks* (i.e. patterns formed by 2^m adjacent cells) containing each the same next state (Fig. 7; the Φ symbol describes a don't care condition). We obtain therefore a *simplified binary decision tree* (Fig. 8) having nine branches. For a microprogrammed realization, the binary decision tree of Fig. 8 is the flowchart for the gene of the tape cells. The software implementation of this flowchart (Fig. 9) 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) **Tapegene** describing the tape cells of the checker is then written by means of the mnemonics and of the labels L1 to L8 and End.

T+	X 000	(001	A 011	) 010	0 100	- 101	- 111	1 110	T T2:0
-> 0001	NOP	NOP	NOP	X	Φ	Φ	Φ	Φ	
0<- 0100	NOP	X	0	NOP	Φ	Φ	Φ	Φ	
1<- 1100	NOP	0	1	Φ	Φ	Φ	Φ	Φ	
^ 0010	Φ	Φ	Φ	Φ	NOP	Φ	Φ	NOP	
0 0000	NOP	NOP	NOP	NOP	Φ	Φ	Φ	Φ	
H H3:0									

Fig. 7

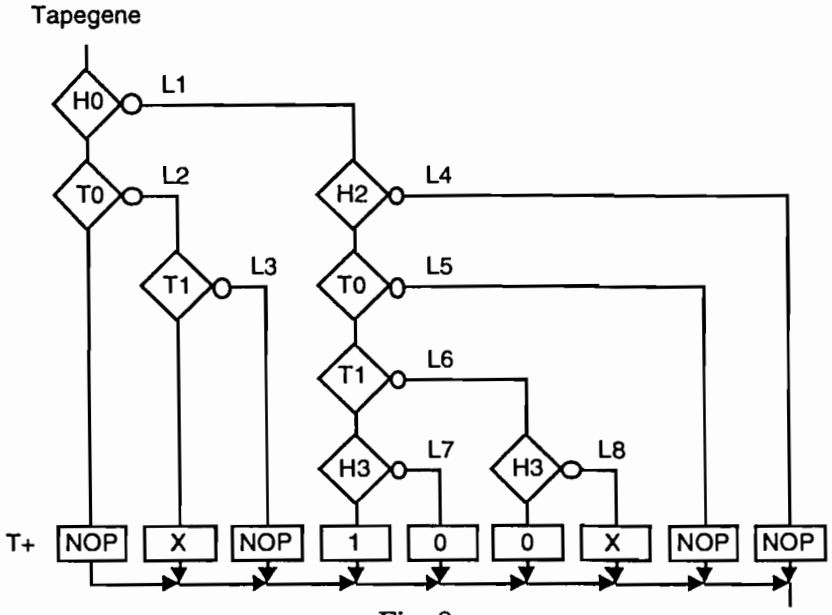

Fig. 8

```
Tapegene:  if H0 else L1                    goto End
           if T0 else L2          L7:       do T =0
           goto End                         goto End
L2:        if T1 else L3          L6:       if H3 else L8
           do T =X                          do T =0
           goto End                         goto End
L3:        goto End               L8:       do T =X
L1:        if H2 else L4                    goto End
           if T0 else L5          L5:       goto End
           if T1 else L6          L4:       goto End
           if H3 else L7          End:      ...
           do T =1
```

Fig. 9

4. A BINARY DECISION MACHINE-BASED FPGA

While our long-term objective is the conception of a wafer scale FPGA, 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. 10).

4.1 MICROTREE cell specification

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

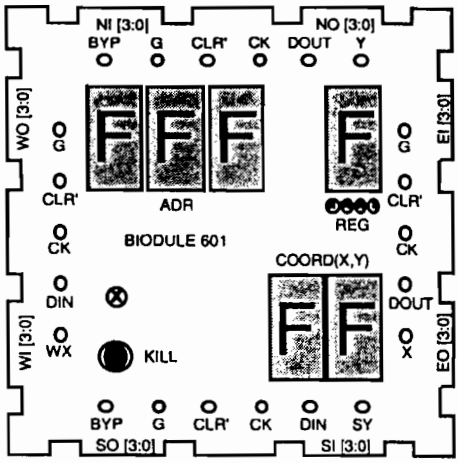

Fig. 10

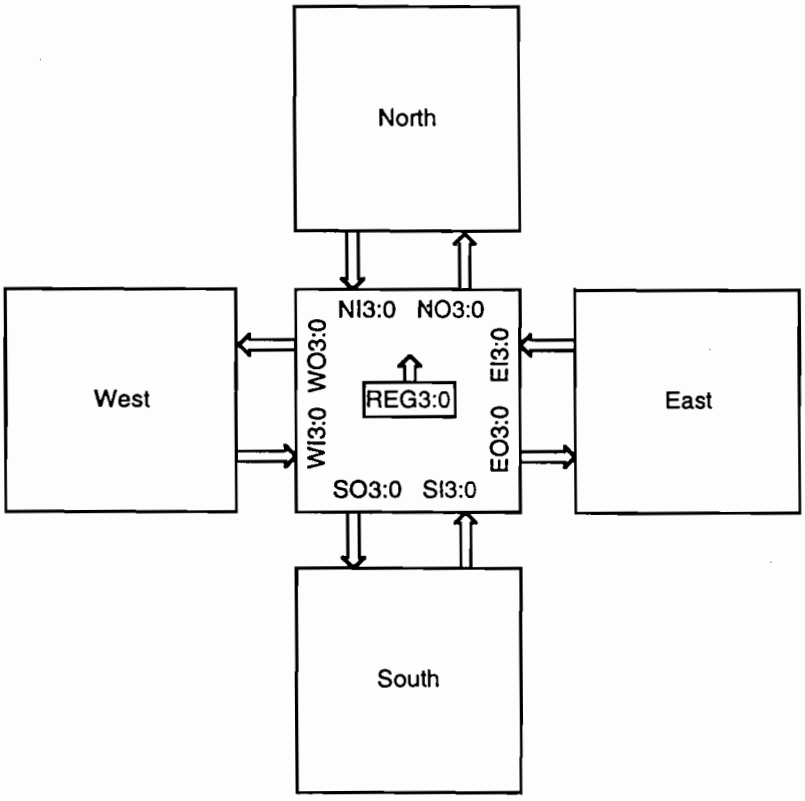

Fig. 11

The first three instructions were introduced above in the computing of the tape gene of the parenthesis checker 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). Each MICROTREE cell (Fig. 11) 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. 12). For example, output NO3 can take one of the following 16 values:

- the 4 bits REG3:0 of register REG;
- the 4 bits SI3:0 of the south input bus SI;
- the 4 bits WI3:0 of the west input bus WI;
- the 4 bits EI3:0 of the east input bus EI.

Note that it is impossible for NO3 to get the value of one of the 4 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. 11):

- VAROUT ∈ {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 ∈ {SI3:0, WI3:0, NI3:0, EI3:0, REG3:0}

remembering that VAROUT and VARIN can never refer to the same cardinal direction (Fig. 12).

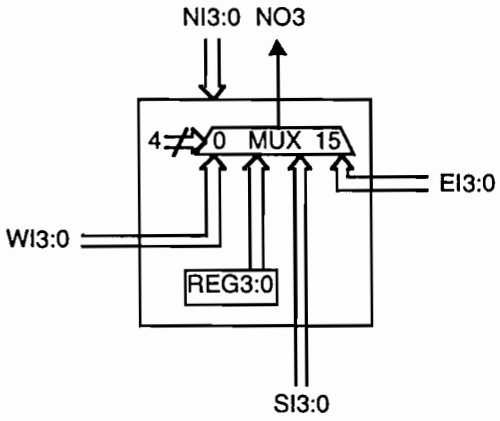

Fig. 12

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

- VAR ∈ {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.

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. 10).

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 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.

4.2 MICROTREE cell physical configuration

A physical configuration is *global* when it is realized in all the MICROTREE cells of the array, independently of the value of the coordinates (X and/or Y). For computing the initial conditions of the parenthesis checker (Fig. 5), we need a boolean variable INIT. This variable will be introduced in the north side of the cellular space and must reach each cell. We thus have the global configuration of Fig. 13, described by the assignment instruction:

- **do** SO1 = NI1

A physical configuration is *local* if it is realized by a sub-set of the MICROTREE cells of the array. Such a configuration depends therefore on the value of the X and/or Y coordinates. The architecture of our Turing machine (Fig. 6) consists of two cellular rows which are uniform, i.e., characterized by cells with identical structure and behavior (Fig. 14). The tape cells (Y=1) calculate the tape state REG3:0 = T3:0. According to Fig. 8, bits of T3:0 and of H3:0 are needed for the calculation of T+. While the internal bits T1 and T0 do not need any programmable connection, the central head cell (Y=2) must perform such connections in order to provide the bits H3, H2 and H0 to the central tape cell. The corresponding assignment instructions are written as follow:

- **do** SO0 = REG0 (≡H0)
- **do** SO2 = REG2 (≡H2)
- **do** SO3 = REG3 (≡H3)

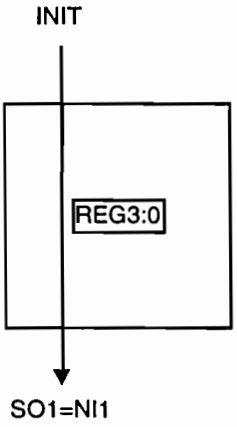

Fig. 13

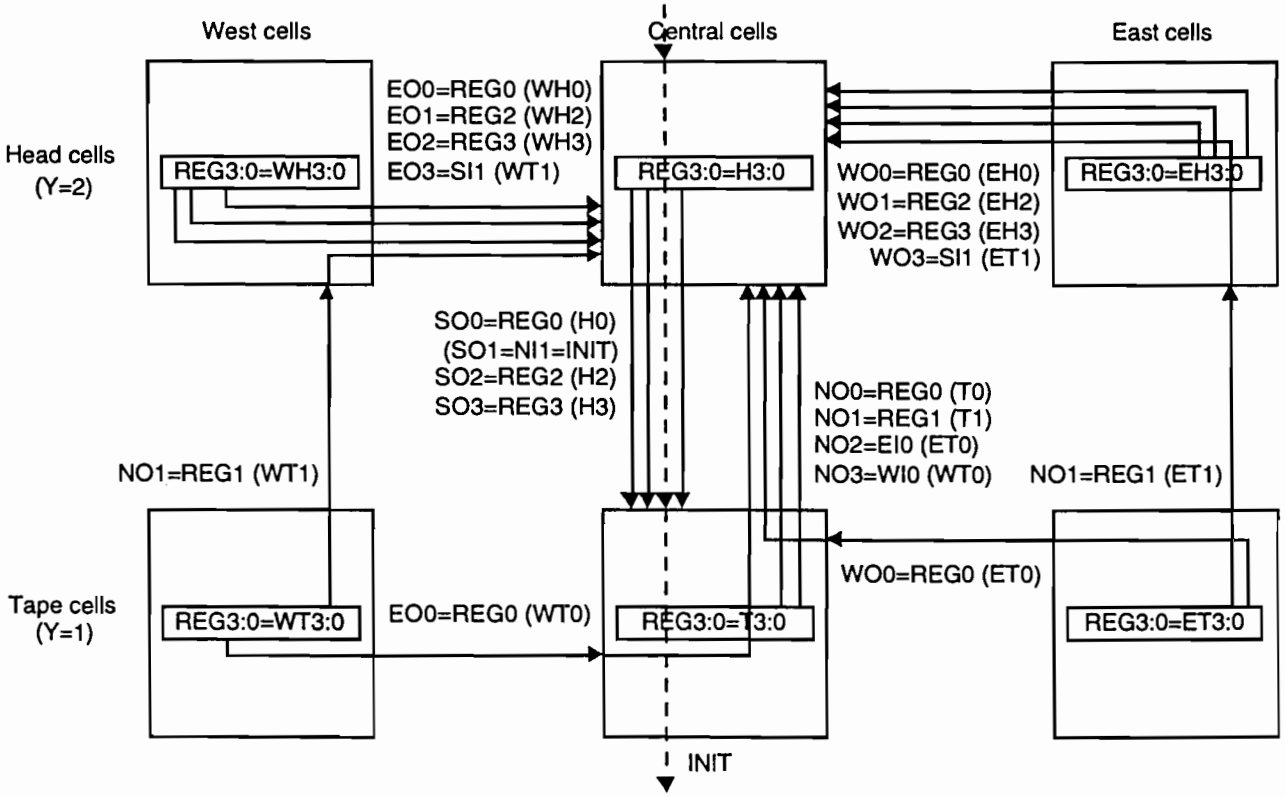

Fig. 14

5. BIOLOGICAL-LIKE PROPERTIES

The binary decision machine-based FPGA specified previously satisfies the general hypothesis about the environment as well as the three characteristics of the Embryonics project. It is therefore endowed with biological-like properties.

5.1 Self-reproduction

The self-reproduction of an artificial organism, for example the specialized Turing machine of Fig. 1, rests on two hypotheses: (1) there exists a sufficient number of spare cells (unused cells at the upper side of the array, at least ten for our example) and (2) the calculation of the coordinates produces a cycle (Y=1->2->1 in Fig. 15). 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 (Y=1->2). In our example, the repetition of the vertical coordinate pattern, i.e. the production of the pattern X=1->2->1->2 (Fig. 15), produces one copy, the *daughter automaton*, of the original or *mother automaton*. 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.

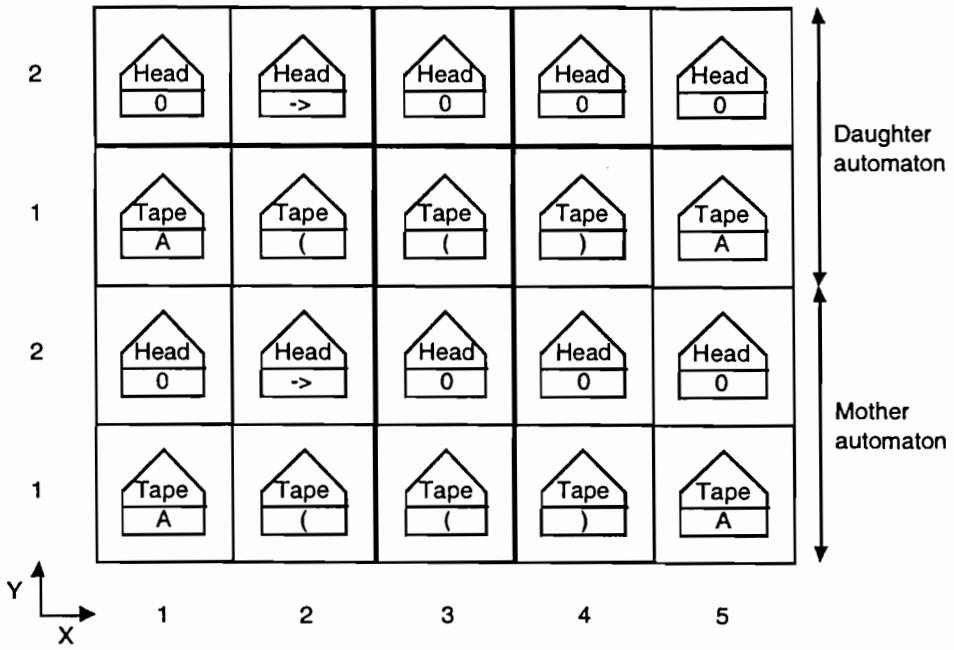

Fig. 15

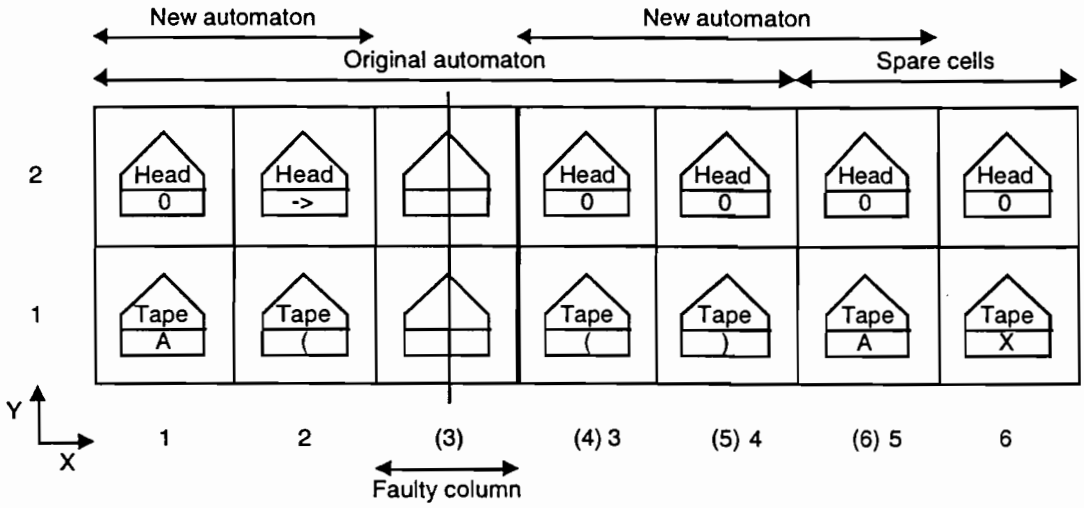

Fig. 16

5.2 Self-repair

In the BIODULE 601 (Fig. 10), 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 [7], [8], are not present in the BIODULE 601. In order to implement self-repair, we have chosen, favoring simplicity, the following process (Fig. 16):

- 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. 16);
- 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 columns, to the right of the array, as there are faulty columns to repair (two spare columns in the example of Fig. 16). 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.

6. CONCLUSION

The main result of our research is the development of a wafer scale FPGA made up of MICROTREE cells which are each based on a binary decision machine capable of executing a microprogram of up to 1024 instructions. The original features of this FPGA 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 FPGA satisfies the general hypothesis about the environment (Section 2.1), as well as the three characteristics of the Embryonics project: multicellular organization, cellular differentiation, and cellular division (Section 2.2). 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-reproduction and self- repair.

The trivial applications of the MICROTREE FPGA 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. The 1-dimensional (Wolfram's) and 2-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). The 1-dimensional (like the random number generator described in [3]) and 2-dimensional (like the present Turing machine) 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 cellular realization of our parenthesis checker (head row or tape row) represents an application of this kind.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[8] 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.