Self-replicating and Self-repairing Field-Programmable Processor Arrays (FPPAs) with Universal Construction

Daniel MANGE, André STAUFFER and 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

Content Areas: artificial life
Tracking Number: 000

Abstract

The Embryonics project (for embryonic electronics), inspired by the basic process of molecular biology, adopts certain features of cellular organization and tranposes them to the 2-dimensional world of integrated circuits on silicon. Properties unique to the living world, such as self-replication and self-repair, are thus applied to artificial objects. Our final objective is the development of a wafer scale integrated circuit capable of self-replication and self-repair. These two biological-like properties seem particularly desirable for artificial systems meant for hostile (nuclear plants) or inaccessible (space) environments. Self-replication 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.
In this paper, we shall demonstrate that a new kind of field-programmable gate array (FPGA) is able to implement a multicellular automaton (a field-programmable processor array: FPPA) which verifies the property of universal construction as defined by von Neumann in his historic work. In a companion paper, it is shown that this automaton can also verify the property of universal computation.

Self-replicating and Self-repairing Field-Programmable Processor Arrays (FPPAs) with Universal Construction

Content Areas: artificial life
Tracking Number: 000

Abstract

The Embryonics project (for embryonic electronics), inspired by the basic process of molecular biology [Wolpert, 1993], adopts certain features of cellular organization and tranposes them to the 2-dimensional world of integrated circuits on silicon. Properties unique to the living world, such as self-replication and self-repair, are thus applied to artificial objects. Our final objective is the development of a wafer scale integrated circuit capable of self-replication and self-repair. These two biological-like properties seem particularly desirable for artificial systems meant for hostile (nuclear plants) or inaccessible (space) environments. Self-replication 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.

In this paper, we shall demonstrate that a new kind of field-programmable gate array (FPGA) is able to implement a multicellular automaton (a field-programmable processor array: FPPA) which verifies the property of universal construction as defined by von Neumann in his historic work [Neumann, 1966]. In a companion paper [Mange, 1997], it is shown that this automaton can also verify the property of universal computation.

1 Ordered Binary Decision Diagrams (OBBDs) for the synthesis of digital systems

In order to implement any digital system (in particular the field-programmable processor array or FPPA elements of our Embryonics project [Mange, 1997]) into a reconfigurable array, we needed to find a method capable of generating, starting from a set of specifications, the configuration of a homogeneous network of elements, the *molecules*, where each molecule is defined by an identical architecture and a usually distinct function (the *molecule code*).

To meet our requirements, we have selected a particular representation: the *ordered binary decision diagram* (OBDD). This representation, with its well-known intrinsic properties such as canonicity [Bryant, 1992], was chosen for two main reasons:

(a) it is a graphical representation which exploits well the 2-dimensional space and immediately suggests a physical realization on silicon;

(b) its structure leads to a natural decomposition into elements realizing a logic test (a diamond), easily implemented by a multiplexer.

We will illustrate the handling of ordered binary decision diagrams through a simple example, an up-down counter. Our choice will lead us to define a field-programmable gate array (FPGA) as a homogeneous array where each element (molecule) contains a programmable multiplexer with one control variable, implementing precisely a logic test. Such an FPGA is said to be *fine-grained*.

Let us consider the relization of the aboved-mentioned modulo-4 up-down counter, defined by the following sequences:

(a) for M = 0: Q1,Q0 = 00 -> 01 -> 10 -> 11 -> 00 (counting up);

(b) 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. 1 (where each diamond represents a test, each square an input boolean value and each diamond embedded in a square a 1-bit memory, i.e., a flip-flop) correspond to a possible realization of the counter.

2 Hardware implementation: a new field-programmable gate array (FPGA) based on a multiplexer

Our design choice has been to implement directly the ordered binary decision diagrams on silicon, and to build our fine-grained basic element (our molecule) around a test element (a diamond). Such an implementation is possible if one replaces each test element with a 2-to-1 multiplexer, keeps the same interconnection diagram, and assigns the values of the leaf elements to the corresponding multiplexer inputs. The two state functions Q1 and Q0 are available at the outputs of the top multiplexers which are serially connected to two flip-flops: the ordered diagrams of Fig. 1 are read bottom-up.

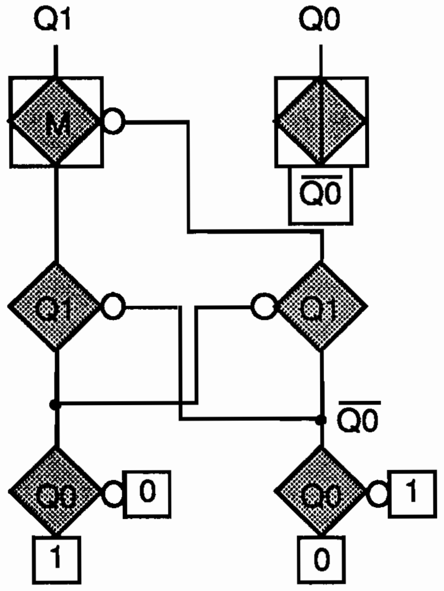

Fig. 1. Ordered binary decision diagrams of a modulo-4 up-down counter.

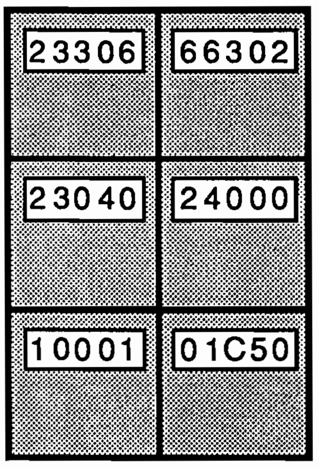

Fig. 2. 9-MUXTREE element implementation of the up-down counter.

A simple examination of Fig. 1 allows us to identify the main features of the programmable element, henceforth referred to as MUXTREE (for multiplexer tree) element:

(a) each of the two inputs of the multiplexer (the direct and the "dotted" inputs of each diamond) will be programmable; the input will be either a logic constant (0 or 1) or the output of the multiplexer of one of the neighboring elements to the South, Southeast, or Southwest;

(b) the output of the multiplexer will be, therefore, connected to the inputs of the multiplexers in the neighboring elements to the North, Northeast, and Northwest;

(c) the realization of sequential systems requires the presence, in each element, of a synchronous memory element, a D-type flip-flop, which will allow, in our example, to obtain directly the values Q1 and Q0 needed for the display and for the retroaction of the secondary variables;

(d) long-distance connections are necessary to connect an element to any other element in the array; in our example, the variable Q1 (itself obtained at the output of the mentioned flip-flop) must be brought back to the control inputs of the multiplexers of the middle row of elements; this type of connection demands a system of universal busses, running through the entire array.

In brief, the heart of the molecule remains the 2-to-1 multiplexer, optionally followed by a flip-flop. Inputs and outputs are programmable and can be connected either to immediate neighbors, according to a topology proper to binary decision diagrams (where information flows from the bottom to the top), or to faraway molecules through a network of perfectly symmetric busses.

Fig. 3 shows the core of the MUXTREE molecule, where SB designates a *switch block* allowing any connection between the horizontal and vertical busses. In order to facilitate the hexadecimal representation of its molecule code, the 17 field-program bits of the element are organized as a 20-bit data MOLCODE19:0; these 17 bits are stored in a *configuration register* CREG.

We will now illustrate the realization, using MUXTREE elements, of the above mentioned example, represented by the ordered binary decision diagrams of Fig. 1. It can be immediately seen that this counter can be realized using an array of 3 lines and 2 columns (that is, by a total of 6 MUXTREE elements). Starting from the binary decision diagrams of Fig. 1 and using the definitions of the MUXTREE element (Fig. 3) allow us to compute the 17 control bits of each molecule code, and finally produce the molecule codes of Fig. 2 [Marchal, 1994].

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

3 Assembling molecules into cells

Our goal aims at assembling a number of molecules (the MUXTREE elements) into a larger pattern, a cell, in order to get a functional unit; in our example, this unit is a very simple modulo-4 counter. Yet our final objective is the design of a complete artificial cell, i.e., a binary decision machine with its random access memory, able to act as an element of a field-programmable processor array (FPPA) described in the companion paper [Mange, 1997].

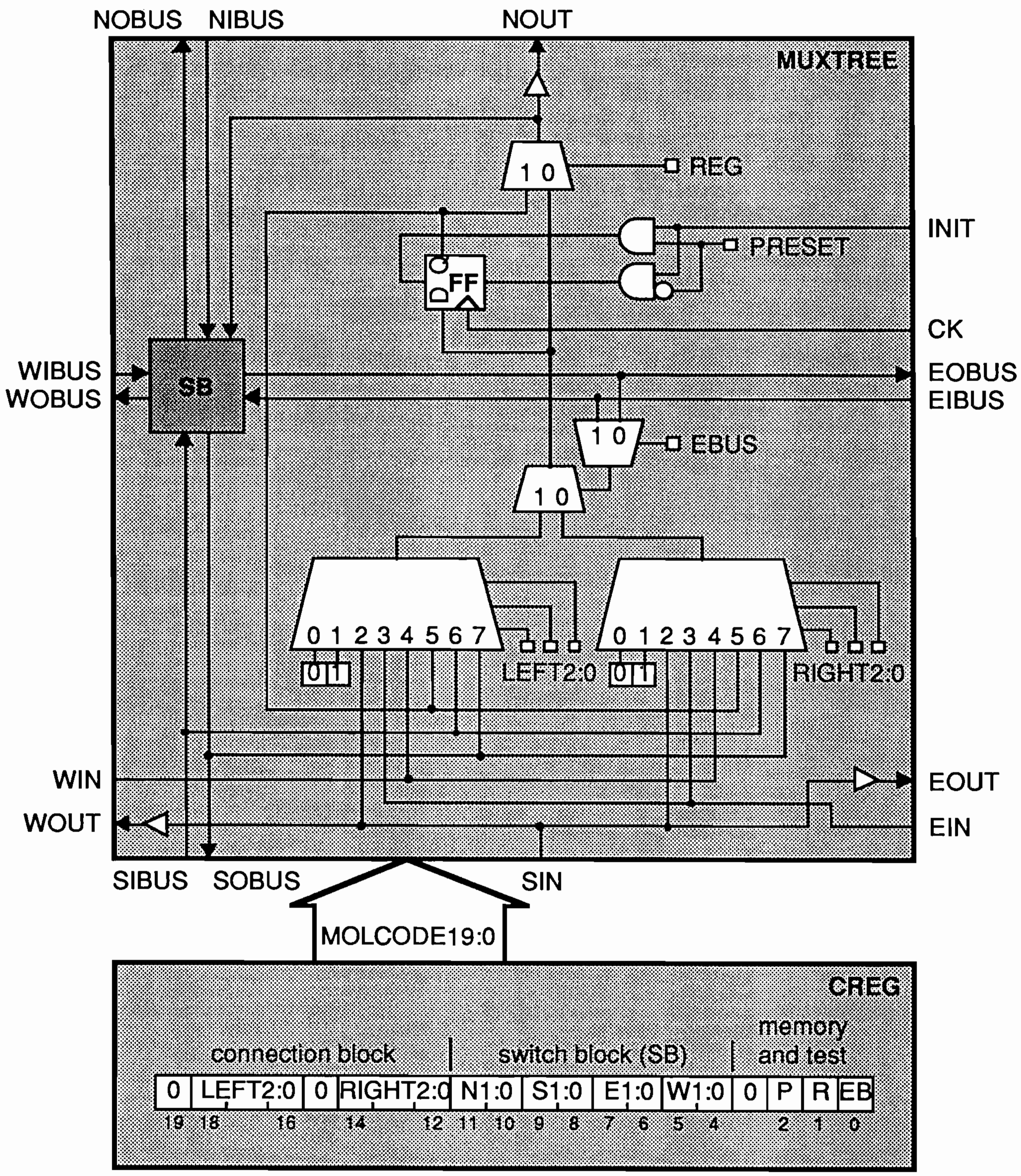

Fig. 3. Logic layout of a MUXTREE element, including the configuration register CREG and the switch block SB.

Since we cannot know the dimensions of a cell before we configure the array of molecules, we need a mechanism capable of "colonizing" the MUXTREE array, subdividing the entire array into cells. The mechanism we have adopted is to introduce a very simple molecular automaton (Fig. 4), capable of creating a set of *boundaries*; it then becomes possible to configure the entire array by entering the configuration of a single cell, which will

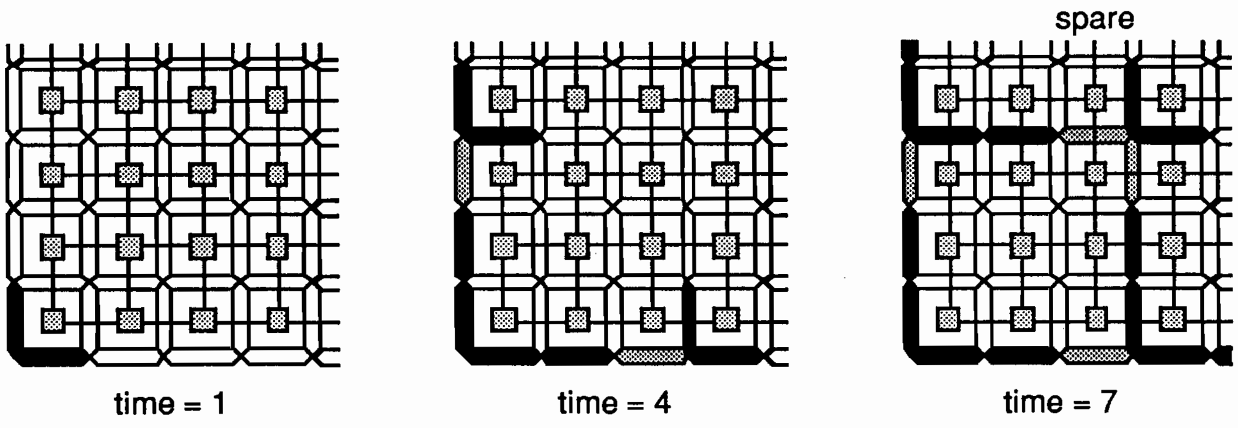

Fig. 4. The molecular automaton colonizing an array of MUXTREE elements.

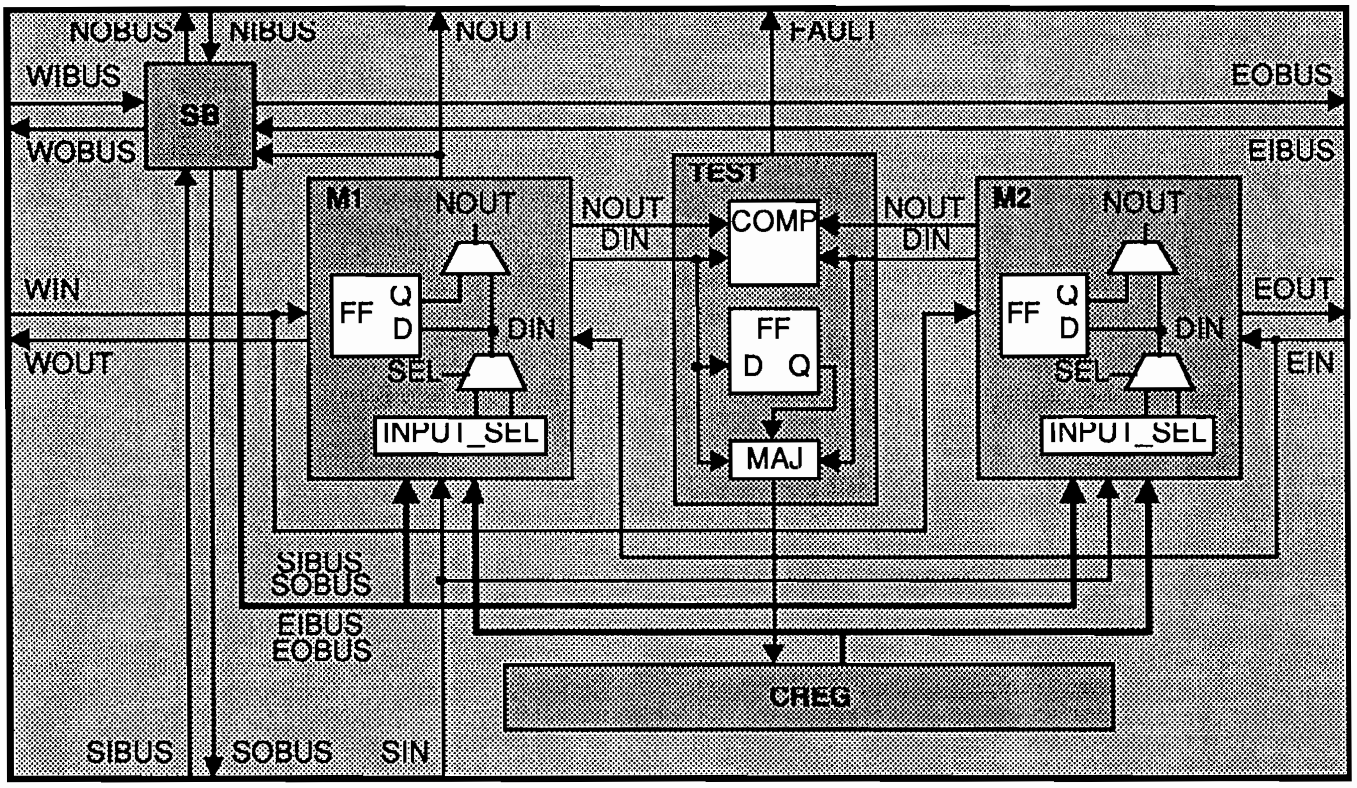

Fig. 5. A self-testing MUXTREE element using space redundancy.

automatically be replicated within all the boundaries. This molecular automaton is roughly inspired by the self-replicating Langton's loop [Langton, 1984]; its design is described elsewhere [Stauffer, 1997].

A very interesting "bonus" of this system is that it becomes possible to use this automaton to define which columns of the array will be spare columns, used for self-repair (gray in Fig. 4). The frequency of these columns, and therefore the robustness of the system, are therefore entirely programmable, rather than handwired, and can thus be set to meet the requirements of a single application.

Coming back to our original example of a modulo-4 up-down counter (Fig. 2), it is obvious to observe that the colonizing process of Fig. 4 will provide the desired MUXTREE array able to implement our design (the 6 molecules of the counter) with a spare column to the right (3 spare molecules).

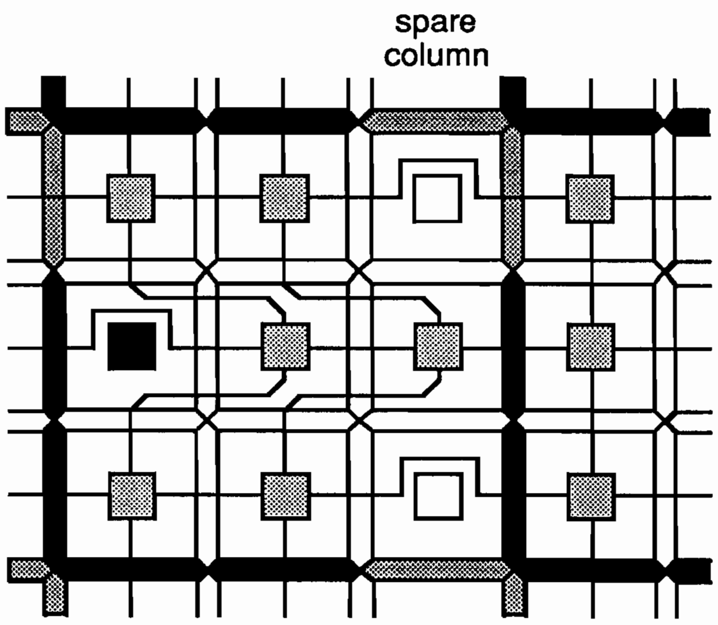

Fig. 6. The self-repair mechanism for an array of MUXTREE elements.

4 Built-in self-test and self-repair

The on-line self-test system we developed for the MUXTREE FPGA had to respect a number of very restrictive constraints [Tempesti, 1997]. The main constraint was *size*: a MUXTREE element being very fine-grained (a single 2-input multiplexer), the logic dedicated to self-test necessarily had to be itself small. This constraint forced us to adopt a set of compromises with regard to the fault-detection capabilities of the system. A MUXTREE element can be divided into three parts (Fig. 5):

(a) the functional part of the element (the multiplexer and the internal flip-flop) is tested by space redundancy: the logic is duplicated (M1 and M2) and the outputs of the two elements compared to detect a fault; a third copy of the flip-flop was added to allow self-repair (i.e., to save the flip-flop state);

(b) the configuration register (CREG) is tested as the configuration is being entered (and thus not entirely on-line); being implemented as a shift register, it can be tested using a special *test sequence* introduced in all elements in parallel *before* the actual configuration for the system;

(c) the connections are the weakest point in the system; 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 exploits the connections to reconfigure the array; in the current system, therefore, we decided not to test the connections directly; this assumption is in accordance with the present state of the art [Negrini, 1989].

The self-repair system had to meet the same constraints as those of the self-test system, and in particular the requirement that additional logic be small. Exploiting the fact that the spare columns are distributed and that their frequency is programmable, we limited the reconfiguration of the array to a single element per line between two spare columns (Fig. 6). This 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 present state of the art [Shibayama, 1997].

It should be added that if the self-repair capabilities of the MUXTREE molecular level is exceeded, a global KILL signal is generated and the system will attempt to reconfigure at the higher (cellular) level, as mentioned in the companion paper [Mange, 1997].

5 Design methodology and biological equivalences

In our Embryonics project, the design of a multicellular automaton necessitates the following stages:

(a) the original specifications are mapped into a homogeneous array of cells (binary decision machines with their associated random access memory); the software (a microprogram) and the hardware (the architecture of the cell) are tailored according to the specific example (Turing machine, electronic watch, random number generator, etc...); in biological terms, the microprogram is considered as the *functional part* of the final genome;

(b) the hardware of the cell is implemented into a homogeneous array of molecules, the MUXTREE elements; spare columns are introduced in order to improve the global reliability and to get a square array; our artificial cell being analogous to the ribosome of a natural cell, the string of the molecule codes can be considered as the *ribosomic part* of the final genome;

(c) the dimensions of the final molecular square array, as well as the frequency of the spare columns, define the string of data necessitated by the molecular automaton, the self-replicating loop creating 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 part* of the genome.

Given the molecular array of MUXTREE elements, the corresponding programming has to take place in reverse order:

(a) inject the polymerase part of the genome in order to get the boundaries between cells;

(b) inject the ribosomic part of the genome in order to program the molecular FPGA and to get the final architecture of each cell;

(c) store the functional part of the genome into the random access memory of each cell in order to make the cell ready to execute the specifications (cellular FPPA).

6 Universal construction

Our final goal aims at developing powerful artificial cells, such as the field-programmable processor array (FPPA) element described in the companion paper [Mange, 1997]. In a first experimental stage, we have designed a FPPA cell embedding a simple binary decision machine with a random access memory able to store 36 10-bit micro-instructions [Stauffer, 1997]; the corresponding microprogram is sufficiently large for realizing a modulo-60 counter, decomposed into two cells: a modulo-10 and a modulo-6 counter. Such a cell is built on a 30 by 30 square array of MUXTREE elements (i.e., 900 molecules, with a spare column every 3 columns).

With a sufficiently large array of MUXTREE molecules, it can be possible to implement a cell of any dimensions, like that needed by the Turing machine described in the companion paper [Mange, 1997]; the property of universal construction is thus demonstrated.

Acknowledgments

This work was supported in part by grant 20-42270.94 from the Swiss National Science Foundation.

References

[Bryant, 1992] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3): 293-318, 1992.

[Langton, 1984] C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10: 135-144, 1984.

[Mange, 1997] D. Mange, A. Stauffer and G. Tempesti. Self-replicating and Self-repairing Field-Programmable Processor Arrays (FPPAs) with Universal Computation. *Proceedings of the International Joint Conference on Artificial Intelligence*, submitted. Nagoya, Japan, 1997.

[Marchal, 1994] P. Marchal and A. Stauffer. Binary decision diagram oriented FPGAs. *Proceedings of the 2nd International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, pages 1-10, Berkeley, California, February 1994.

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

[Neumann, 1966] J. von Neumann. *The Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, 1966.

[Shibayama, 1997] A. Shibayama, H. Igura, M. Mizuno and M. Yamashina. An Autonomous Reconfigurable Cell Array for Fault-Tolerant LSIs. *Proceedings of the 44th International Solid-State Circuits Conference*, pages 230-231 and 462, San Francisco, California, February 1997.

[Stauffer, 1997] A. Stauffer. Membrane Building and Binary Decision Machine Implementation. Technical Report, Swiss Federal Institute of Technology, Lausanne, 1997.

[Tempesti, 1997] G. Tempesti, D. Mange and A. Stauffer. A Robust Multiplexer-Based FPGA Inspired by Biological Systems. To be published in *Euromicro, Special Issue on Dependable Parallel Computer Systems*, July 1997.

[Wolpert, 1993] L. Wolpert. *The Triumph of the Embryo*. Oxford University Press, Oxford, 1993.