From Data and Signals Cellular Automata to Self-organizing Circuits

André Stauffer and Joël Rossier

Logic Systems Laboratory, Ecole polytechnique fédérale (EPFL),
IN-Ecublens, CH-1015 Lausanne, Switzerland
{name.surname}@epfl.ch
lslwww.epfl.ch

Abstract. Self-organizing circuits are able to grow, to self-replicate, and to self-repair. These properties are implemented in hardware thanks to configuration, cloning and cicatrization mechanisms. They are realized by the configuration layer of the POEtic tissue, a data and signals cellular automata based circuit. Specified as a data-flow processor, the application and routing layers of the circuit compose a timer as an application example.

1 Introduction

Borrowing three structural principles (multicellular architecture, cellular division, and cellular differentiation) from living organisms, we have already shown how embryonic hardware [1] is able to implement bio-inspired properties in silicon. This hardware implementation leads to self-organizing circuits, based on data and signals cellular automata (DSCA), capable to deal with faults in a fully automatic way. In Section 2, the self-organizing mechanisms are implemented as a configuration layer in the POEtic tissue, a reconfigurable circuit that draws inspiration from the structure of complex biological organisms. The application and routing layers of the circuit are then specified in order to define a data-flow processor, the MOVE processor (Section 3). Using four such processors, a timer is realized as an application example. A brief conclusion (Section 4) summarizes our paper and opens new research avenues.

2 DSCA Based Reconfigurable Circuit

2.1 Circuit Characteristics

The *POEtic tissue* [2] is a reconfigurable circuit that draws inspiration from the structure of complex biological organisms to implement the three main models commonly used in bio-inspired systems: (1) *Phylogeny*, the history of evolution of the species through time, (2) *Ontogeny*, the development of an individual as directed by its genetic code, and (3) *Epigenesis*, the development of an individual through learning processes. This tissue is the first hardware substrate

H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 531–536, 2008.

dedicated to the implementation of systems able to combine the three axes of bio-inspiration into one single circuit. Physically, the tissue is a 2-dimensional array of molecules and each molecule is composed of the three layers described in the following paragraphs.

2.2 Configuration Layer

The *configuration layer* implements the self-organizing mechanisms and their constituting processes [3]. This layer is designed as data and signals cellular automaton (DSCA) cell [4], resulting from the interconnection of a processing unit handling the data and a control unit computing the signals. In the detailed architecture of our layer (Fig. 1a), six resources implement the *growth* and *branching processes*:

- An input multiplexer DIMUX, selecting one out of the four input data NDI, EDI, SDI or WDI.
- A 2N-level stack organized as N genotypic registers G1 to GN (for mobile data), and N phenotypic registers P1 to PN (for fixed data).
- An output buffer DOBUF producing the output data DO.
- An encoder ENC for the input signals NSI, ESI, SSI, and WSI.
- A transmission register I for the memorization of the input selection.
- A generator GEN producing the output signals NSO, ESO, SSO, and WSO.

In order to implement the *load*, *repair* and *reset processes*, the architecture of the configuration level (Fig. 1a) requires four supplementary resources:

- A decoder DEC defining the mode and the type of the molecule.
- A signal register S.
- A mode register M.
- A type register T.

To allow the bypassing of the spare, faulty or repair molecules, data and signals transmission multiplexers and demultiplexers are added to the configuration layer. Depending on its molecular type T, the layer finally controls its output signals with buffers. These buffers limit the propagation of the load and reset signals according to the boundaries of the cell.

2.3 Application Layer

The *application layer* implements the logic design of the system under development as well as its short range connections between neighboring molecules. The core of this layer (Fig. 1b) is made up of four resources:

- An input multiplexer AIMUX, selecting four inputs out of the four application data NAI, EAI, SAI, WAI, and the routing data RO.
- A 16-bit look-up table LUT.

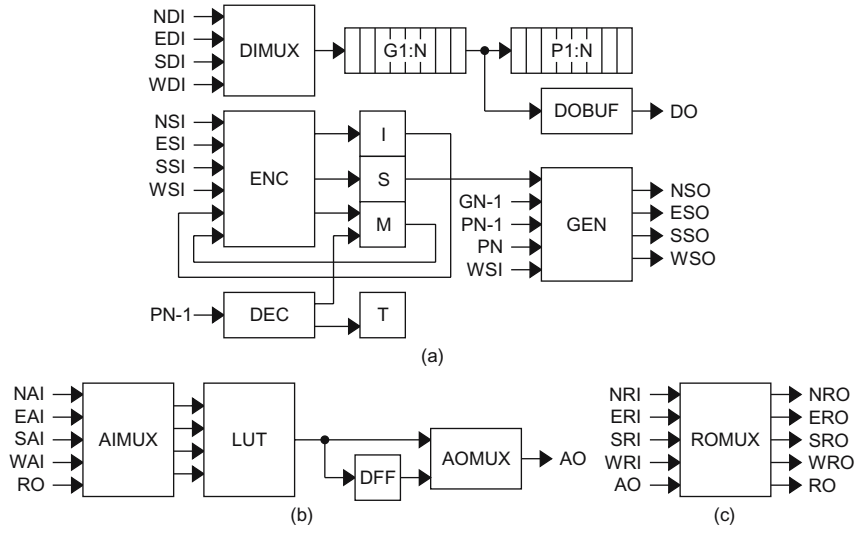

Fig. 1. Detailed architecture of a molecule. (a) DSCA cell corresponding to the configuration layer. (b) Application layer. (c) Routing layer.

- A D-type flip-flop DFF for the realization of sequential systems.
- An output multiplexer AOMUX selecting the combinational or the sequential data as application output AO.

To allow the bypassing of the spare, faulty or repair molecules, transmission multiplexers are added to the application layer.

2.4 Routing Layer

The *routing layer* handling the short range connections between distant molecules is made of a single resource (Fig. 1c):

- An output multiplexer ROMUX selecting the five outputs NRO, ERO, SRO, WRO, and RO out of the four routing input data NRI, ERI, SRI, WRI, and the application output data AO.

To allow the bypassing of the spare, faulty or repair molecules, transmission multiplexers are added to the routing layer.

3 Multi-processor Application

3.1 MOVE Processor

The *MOVE processor*, originally developed as an application-specific dataflow processor [5], relies on a set of functional units connected together by a bus. Fig. 2 details the constituting resources of the processor:

– A program memory PM.
– An instruction fetch unit IF.
– Four functional units with input registers RI and output registers RO.
– Two communication units with address registers ADR and data registers DATA.

The instructions of the processor move operands into the input registers RI of the functional units and move the result from their output registers RO. Using their address registers ADR and their data registers DATA, the communication units are handled in the same way as the functional units.

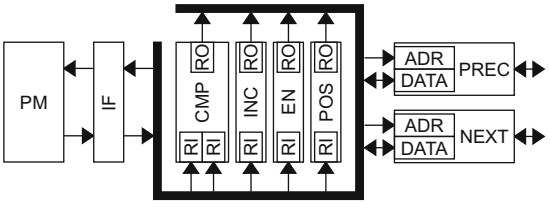

Fig. 2. Detailed architecture of the MOVE processor

3.2 Timer Application

A *timer* counting seconds (from 00 to 59) and minutes (from 00 to 59) is realized as a chain of four counters SU (units of seconds), ST (tens of seconds), MU (units of minutes), and MT (tens of minutes). This application involves four MOVE processors that are specified as modulo-10 counters for the units (seconds SU or minutes MU) and modulo-6 counters for the tens (seconds ST or minutes MT). Each processor contains consequently the following specialized functional units (Fig. 2):

– A comparator CMP.
– An incrementer INC.
– A count enable unit EN.
– A position unit POS for the location within the chain.

The chaining of the four processors is implemented at the programmable circuit level and realized by a distributed long range routing algorithm [6] that dynamically connects the output communication units NEXT to the input communication units PREC.

3.3 POEtic Implementation

Each MOVE processor, using one basic cell of the POEtic tissue for each of its 30×12 molecules, implements a counter cell of the timer organism. In order to build this organism (Fig. 3a), the structural configuration mechanism, the functional configuration mechanism, and the cloning mechanism are applied at the cellular level. Starting with the structural and functional configuration data of

the MOVE processor, these mechanisms generate successively the four counters of the timer.

The cicatrization mechanism results from the introduction in each cell of one column of spare molecules to the right (Fig. 3a), defined by the structural configuration of the MOVE processor, and the automatic detection of faulty molecules. Thanks to this mechanism, the faulty molecule of the upper right cell (Fig. 3b) is deactivated, isolated from the network, and replaced by the nearest right molecule, which will itself be replaced by the nearest right molecule, and so on until a spare molecule is reached. The functional reconfiguration mechanism takes then place in order to regenerate the counter cell of the timer organism. As shown in Fig. 3b, the display of the regenerated counter cell presents some graphical distortion.

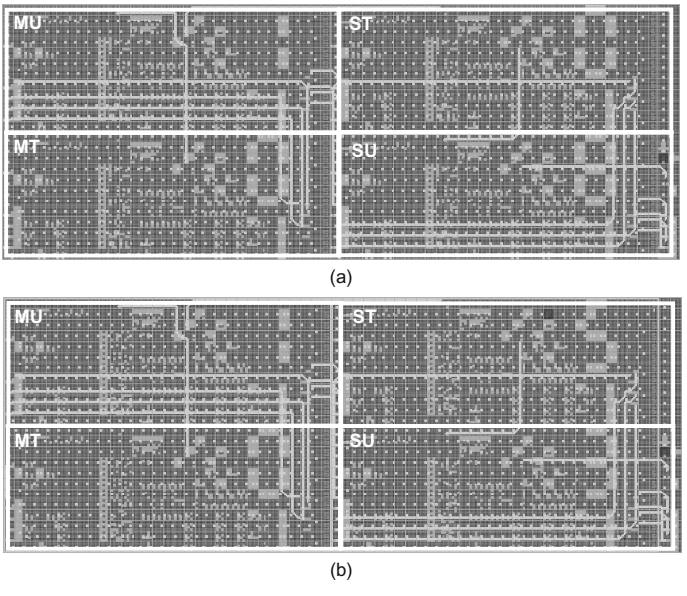

Fig. 3. POEtic implementation of the timer. (a) Original configuration of the two rows by two columns of processors displaying clockwise MT,MU:ST,SU=01:08. (b) Cicatrization of the upper right processor displaying a distorted ST=5.

4 Conclusion

The self-organizing mechanisms are made of simple processes like growth, load, branching, repair, and reset. They allow the cellular systems to possess bio-inspired properties such as:

- Cloning or self-replication at cellular and organismic levels.
- Cicatrization or self-repair at the cellular level.

Starting with the POETIC tissue, a DSCA based reconfigurable circuit, we described first the configuration layer of its basic cell implementing the self-organizing mechanisms and their underlying processes. We detailed then the corresponding application layer and routing layer as well as their specifications in order to define a MOVE processor. We finally realized a timer made up of four such processors as an application example.

In order to improve our systems, we intend to study additional hardware features such as:

- Automatic detection of faulty molecules, erroneous configuration data, and application dysfunction.
- Asynchronous implementation at the organismic level and synchronous implementation at the cellular level.

References

1. Mange, D., Stauffer, A., Petraglio, E., Tempesti, G.: Embryonics Machines that Divide and Differentiate. In: Ijspert, A.J., Murata, M., Wakamiya, N. (eds.) BioADIT 2004. LNCS, vol. 3141, pp. 328–343. Springer, Heidelberg (2004)
2. Tyrrell, A., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J.-M., Rosenberg, J., Villa, A.: Poetic Tissue: An Integrated Architecture for Bio-inspired Hardware. In: Tyrrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS, vol. 2606, pp. 129–140. Springer, Heidelberg (2003)
3. Stauffer, A., Mange, D., Rossier, J., Vannel, F.: Bio-inspired Systems with Self-developing Mechanisms. In: Kang, L., Liu, Y., Zeng, S. (eds.) ICES 2007. LNCS, vol. 4684, pp. 151–162. Springer, Heidelberg (2007)
4. Stauffer, A., Sipper, M.: The Data-and-Signals Cellular Automaton and its Application to Growing Structures. Artificial Life 10(4), 463–477 (2004)
5. Corporaal, H., Mulder, H.: MOVE: A Framework for High-performance Processor Design. In: Proceedings of the Int. Conference on Supercomputing, pp. 692–701 (2003)
6. Moreno, J.-M., Sanchez, E., Cabestani, J.: An In-system Routing Strategy for Evolvable Hardware Programmable Platforms. In: Keymeulen, D., Stoica, A., Lohn, J., Zebulum, R.S. (eds.) Proceedings of the Third NASA/DOD Workshop on Evolvable Hardware, pp. 157–166. IEEE Computer Society, Los Alamitos (2001)