A Dynamic Routing Algorithm for a Bio-Inspired Reconfigurable Circuit

Yann Thoma[1,3], Eduardo Sanchez[1], Juan-Manuel Moreno Arostegui[2], and Gianluca Tempesti[1]

[1] Swiss Federal Institute of Technology at Lausanne (EPFL), Lausanne, Switzerland
[2] Technical University of Catalunya (UPC), Barcelona, Spain
[3] Corresponding author. E-mail: yann.thoma@epfl.ch

Abstract. In this paper we present a new dynamic routing algorithm specially implemented for a new electronic tissue called POEtic. This reconfigurable circuit is designed to ease the implementation of bio-inspired systems that bring cellular applications into play. Specifically designed for implementing cellular applications, such as neural networks, this circuit is composed of two main parts: a two-dimensional array of basic elements similar to those found in common commercial FPGAs, and a two-dimensional array of routing units that implement a dynamic routing algorithm which allows the creation of data paths between cells at runtime.

1 Introduction

Life is amazing in terms of complexity and of adaptability. After the fertilization of an ovule by a spermatozoid, a simple cell is capable of recursively dividing itself to create an entire living being. During its lifetime, an organism is also capable of self-repair in case of external or internal aggressions. Living beings possessing a neural network can learn tasks which allow them to adapt to their environment. And finally, at the population level, evolution allows a population to evolve in order to survive in an ever-changing environment. The aim of the POEtic project [7][8][10] is to design a new electronic circuit, drawing inspiration from these three life axes: Phylogenesis (evolution) [6], Ontogenesis (development) [9], and Epigenesis (learning) [4].

Ontogenetic methods, which are used to develop a self-repair circuit, need to change the functionality of the circuit at runtime. Epigenetic mechanisms, using neural networks, could also need to create new neurons, and therefore new connections between neurons at runtime. As commercially FPGAs usually don't have any dynamic self-reconfiguration capabilities, a new circuit capable of self-configuration is essential.

In the next section, we present the general architecture of the POEtic chip, decomposed into two subsystems. In section 3, we describe the basic elements of the circuit: the molecules. Section 4 fully explains the dynamic routing algorithm implemented in order to ease the creation of long distance paths into the chip.

2 Structural Architecture

The POEtic circuit is composed of two parts (figure 1): the organic subsystem, which is the functional part of the circuit, and the environmental subsystem. Cells, and thus organisms, are implemented in the organic subsystem. It is composed of a grid of small molecules and of a cellular routing layer. Molecules are the smallest unit of programmable hardware which can be configured by software, while dedicated routing units are responsible for the inter-cellular communication. The main role of the environmental subsystem is to configure the molecules. It is also responsible for the evolution process, and can therefore access and change every molecule's state in order to evaluate the fitness of an organism.

Each cell of an organism is a collection of molecules, which are the basic blocks of our circuit. The size and contents of the cells depend on the application. Therefore, for each application, a developer will have to design cells fitting into the molecules.

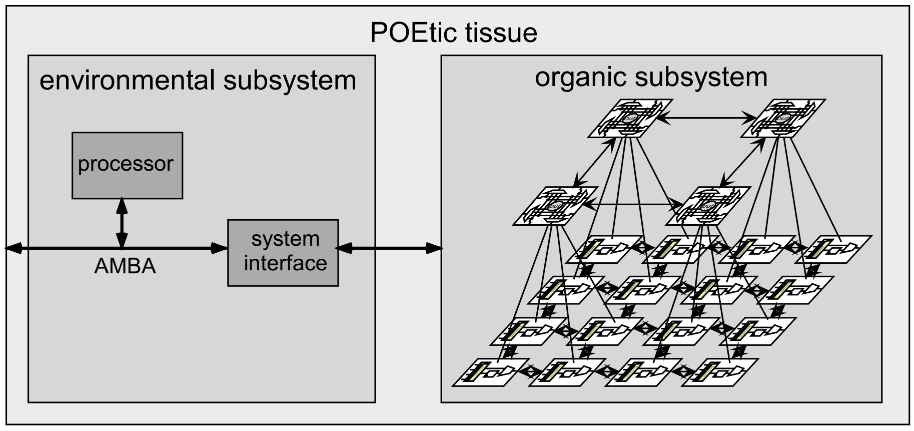

Fig. 1. The POEtic chip, composed of an environmental and an organic subsystems.

2.1 Environmental Subsystem

The environmental subsystem is primairly composed of a micro-controller: a 32-bit RISC-like processor. Its function is to configure the molecules, to run the evolutionary mechanisms, and to manage chip input/output. In order to speed up evolution processes, a random number generator has been added directly in the hardware. An AMBA bus [1] is used to connect the processor to a system interface that takes care of the communication between the processor and the organic subsystem. This bus is also connected to external pins in order to allow multi-chip communication, as well as the use of an external RAM.

2.2 Organic Subsystem

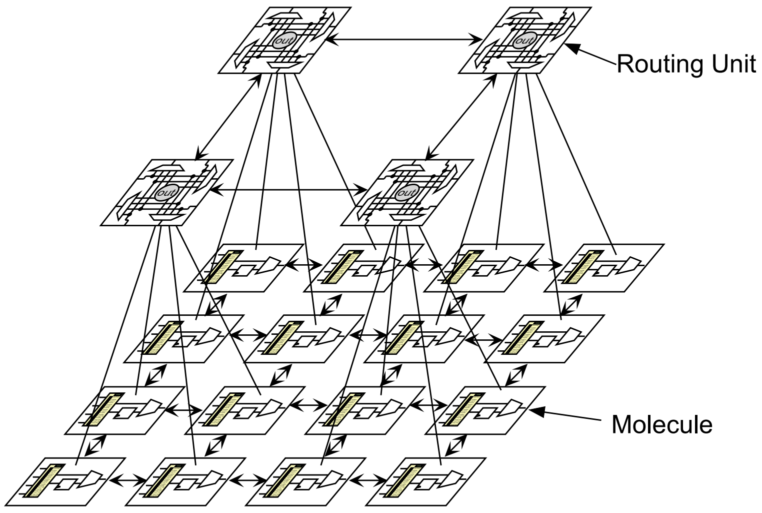

Fig. 2. The organic subsystem is composed of 2 layers: the molecules and the routing units.

The organic subsystem is made up of 2 layers (cf. figure 2): a two-dimensional array of basic logic elements, called molecules, and a two-dimensional array of routing units. Each molecule has the capability of accessing the routing layer that is used for inter-cellular communication. This second layer implements a dynamic routing algorithm allowing the creation of data paths between cells at runtime.

3 Molecular Structure

As briefly presented above, the first layer of the POEtic tissue is a two-dimensional array of small programmable units, called molecules. Each molecule is connected to its 4 neighbors and to a routing unit (4 molecules for 1 routing unit), and contains a 16-bit look-up table (LUT) and a flip-flop (DFF). This structure, while seemingly very similar to standard FPGAs [2], is however specially designed for POEtic applications: different running modes let the molecule act like a memory, a serial address comparator, a cell input, a cell output, or others. With a total of 8 modes, these molecules allow a developer to build cells that are capable of communicating with each other, of storing a genome, of healing, and so on.

The 8 modes of operation of the molecule are the following:

- *Normal*: the LUT is a simple 16-bit look-up table.
- *Arithmetic*: the LUT is split into two 8-bit look-up tables: one for the molecule output, and one for a carry. A carry-chain physically sends the carry to the south neighbor, allowing rapid arithmetic operations.
- *Communication*: the LUT is split into one 8-bit shift register and one 8-bit look-up table. This mode can be used to implement a packet routing algorithm that will not be presented in this paper.
- *Shift memory*: the LUT is considered as a 16-bit shift register. This mode is very useful to efficiently store the genome in every cell. Shift memories can be chained in order to create memories of depth 32, 48, etc.
- *Configure*: the molecule has the capability of reconfiguring its neighbor. Combined with shift memory molecules, this mode can be used to differentiate the cells. A selected part of the genome, stored in the memory molecules, can be shifted to configure the LUT of other molecules (for instance to assign weights to neural synapses).
- *Input address*: the LUT is a 16-bit shift register and is connected to the routing unit. The 16 bits represent the address of the cell from where the information arrives. The molecule's output is the value coming from the inter-cellular routing layer (this mechanism will be detailed in the next section).
- *Output address*: the LUT is a 16-bit shift register and is connected to the routing unit. The 16 bits represent the address of the cell, and the molecule sends the value of one of its inputs to the routing unit (this mechanism will be detailed in the next section).
- *Trigger*: the LUT is a 16-bit shift register, and is connected to the routing unit. Its task is to supply a trigger every `n` clock cycles (where `n` is the number of bits of the addresses), needed by the routing algorithm for synchronization.

To be capable of self-repair and growth, an organism needs to be able to create new cells and to configure them. The configuration system of the molecules can be seen as a shift register of 80 bits split into 5 blocks: the LUT, the selection of the LUT's input, the switch box, the mode of operation, and an extra block for all other configuration bits. Each block contains, as shown in figure 3, together with its configuration, one bit indicating whether the block has to be bypassed in the case of a reconfiguration coming from a neighbor. This bit can only be loaded from the micro-processor, and remains stable during the entire lifetime of the organism.

The special configure mode allows a molecule to partially reconfigure its neighborhood. It sends bits coming from another molecule to the configuration of one of its neighbors. By chaining the configurations of neighboring molecules, it is possible to modify multiple molecules at the same time, allowing, for example, the synaptic weights in a neuron to be changed.

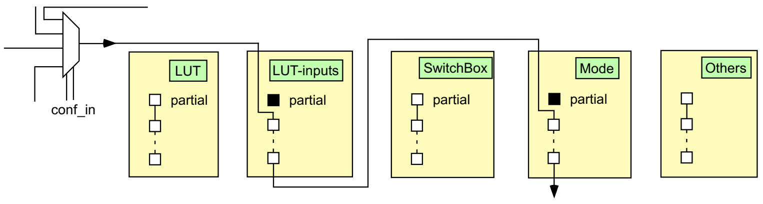

Fig. 3. All configuration bits of a molecule, split up into 5 blocks. The partial configuration bits of blocks 2 and 4 are set, enabling the reconfiguration of the LUT inputs and of the mode of operation by a neighboring molecule.

4 Dynamic Routing

As presented above, our circuit is composed of a grid of molecules, in which cells are implemented. In a multi-cellular system, cells need to communicate with each other: a neural network, for example, often shows a very high density of connections between neurons. Commercial FPGAs have trouble dealing with these kinds of applications, because of their poor or nonexistent dynamic routing capacity. Given the purpose of the POEtic tissue, special attention was payed to this problem. Therefore, a second layer was added on top of the molecules, implementing a distributed dynamic routing algorithm. This algorithm uses an optimized version of the dynamic routing presented by Moreno in [5], to which we supplied a distributed control to where there is no global control of the routing process.

4.1 From Software to Hardware

Our dynamic routing algorithm finds the shortest path between two points in the routing layer. In 1959, Dijkstra proposed a software algorithm to find the shortest path between two nodes in a graph in which every branch has a positive length [3]. If we fix all branches to have a weight of 1, we can dramatically simplify the algorithm. It then becomes a breadth-first search algorithm, as follow:

```
1:  paint all vertices white;
2:  paint the source grey and enqueue it;
3:  repeat
4:    dequeue vertex v;
5:    if v is the target, we found a path - exit the algorithm;
6:    paint v black;
7:    for each white neighbor w of v
8:      paint w grey;
9:      set parent w to v;
10:     enqueue w
11: until the queue is empty
12: if we haven't yet exited, we didn't find the target
```

This algorithm acts like a gas expanding in a labyrinth, but in a sequential manner, one node being expanded at a time, with a complexity of O(V+E) where V is the number of vertices and E is the number of edges. Taking advantage of the hardwares' intrinsic parallelism, it is possible, based on the same principle as the breadth-first search algorithm, to expand all grey nodes at the same time. This dramatically decreases the time needed to find the shortest path between two points, the complexity becoming O(N+M), for a NxM array.

Finding the shortest path is not enough for the POEtic tissue, since we don't have a God telling us which routing unit is the source and which one is the target. In order to have a standalone circuit capable of self-configuration, we need a mechanism to start routings. Molecules, as explained in the previous section, have special modes to access the routing layer. Therefore, input or output molecules have the capability of initiating a dynamic routing.

4.2 Routing Algorithm

The dynamic routing system is designed to automatically connect the cells' inputs and outputs. Each output of a cell has a unique identifier at the organism level. For each of its inputs, the cell stores the identifier of the source from which it needs information. A non-connected input (target) or output (source) can initiate the creation of a path by broadcasting its identifier in the case of an output, or the identifier of its source in the case of an input. The path is then created using a parallel implementation of the breadth-first search algorithm presented above. When all paths have been created, the organism can start operation and execute its task until a new routing is launched, for example after a cell addition or a cellular self-repair.

Our approach has many advantages compared to a static routing process. First of all, a software implementation of a shortest path algorithm, such as Dijkstra's, is very time-consuming for a processor, while our parallel implementation requires a very small number of clock cycles to finalize a path. Secondly, when a new cell is created it can start a routing process without the need of recalculating all paths already created. Thirdly, a cell has the possibility of restarting the routing process of the entire organism if needed (for instance after a self-repair). Finally, our approach is totally distributed without any global control over the routing process, so that the algorithm can work without the need of the central micro-processor.

The routing algorithm is executed in three phases:

Phase 1: Finding a Master

In this phase, every target or source that is not connected to its correspondent partner tries to become master of the routing process. A simple priority mechanism chooses the most bottom-left routing unit to be the master, as shown in figure 4. Note that there is no global control for this priority, every routing unit knows whether or not it is the master. This phase is over in one clock cycle, as the propagation of signals is combinational.

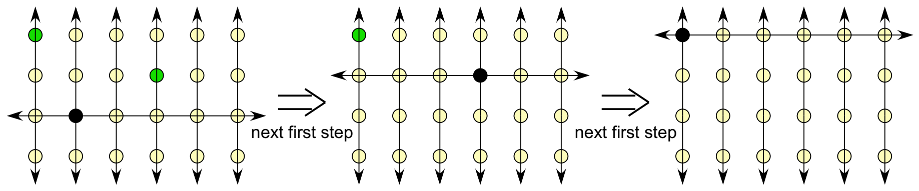

Fig. 4. Three consecutive first steps of the algorithm. The black routing unit will be the master, and therefore perform its routing.

Phase 2: Broadcasting the Address

Once a master has been selected, it sends its address in the case of a source, or the address of the needed source in the case of a target. As shown in section 3, the address is stored in a molecule connected to the routing unit. It is sent serially, in `n` clock cycles, where `n` is the size of the address. The same path as in the first phase is used to broadcast the address, as shown in figure 5.

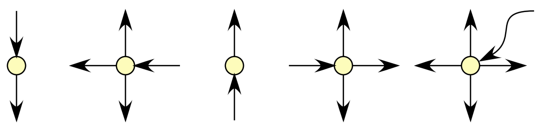

Fig. 5. The propagation direction of the address: north → south || east → south, west, and north || south → north || west → north, east, and south || routing unit → north, east, south, and west.

Every routing unit, except the one that sends the address, compares the incoming value with its own address (stored in the molecule underneath). At the end of this phase, that is, after `n` clock cycles, each routing unit knows if it is involved in this path. In practice, there has to be one and only one source, and at least one target.

Phase 3: Building the Shortest Path

The last phase, largely inspired by [5], creates a shortest path between the selected source and the selected targets. An example involving 8 sources and 8 targets is shown in figure 6, for a densely connected network.

A parallel implementation of the breadth-first search algorithm allows the routing units to find the shortest path between a source and many targets. Starting from the source, an expansion process tries to find targets. When one is reached, the path is fixed, and all the routing resources used for the path will not be available for the next successive iterations of the algorithm.

Figure 7 shows the development of the algorithm, building a path between a source placed in column 1, row 2 and a target cell placed in column 3, row 3.

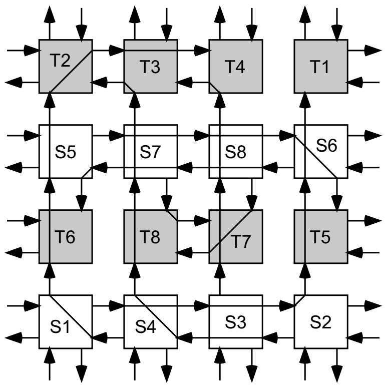

Fig. 6. Test case with a densely connected network.

After 3 clock cycles of expansion, the target is reached, and the path is fixed, prohibiting the use of the same path for a successive routing.

5 Conclusion

In this paper we presented a new electronic circuit dedicated to the implementation of bio-inspired cellular applications. It is composed of a RISC-like microprocessor and of two planes of functional and routing units. The first one, a two-dimensional array of molecules, is similar to standard FPGAs and makes the circuit general enough to implement any digital circuit. However, molecules have self-configuration capabilities that are not present in commercial FPGAs and that are important for the growth of an organism and for self-repair at the cellular level. The second plane is a two-dimensional array of routing units that implement a dynamic routing algorithm. It is used for the inter-cellular communication, letting the tissue dynamically create paths between cells. This algorithm is totally distributed, and hence does not need the control of a microprocessor. Moreover, its scalability allows the creation of cellular networks of any size.

This circuit has been tested with a simulation based on the VHDL files describing the entire system. The next step of the project, which is currently under way and which should be completed by the time the conference will take place, is to develop, from the VHDL files, the VLSI layout and to realize a testchip to validate the design of our circuit.

Due to financial considerations, the first prototype of the POEtic chip will only contain approximately 500'000 equivalent gates. This size will not have enough molecules in one chip for complex designs. It will only be possible to implement a very simple organism on such a small number of molecules. Therefore, we included in the design the possibility of implementing a multi-chip organism by seamlessly joining together any number of chips (figure 8).

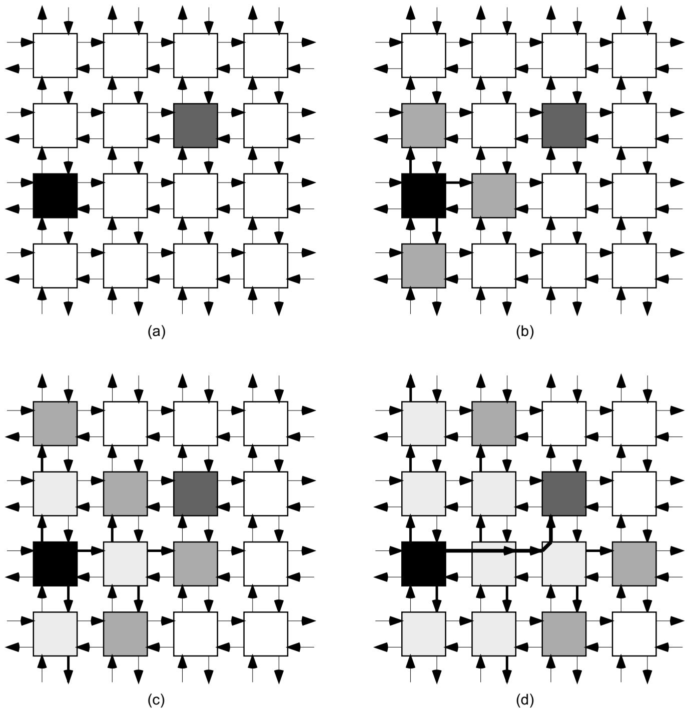

Fig. 7. Step (a) one, (b) two, (c) three and (d) four of the path construction process between the source placed in column 1, row 2 and target cell placed in column 3, row 3.

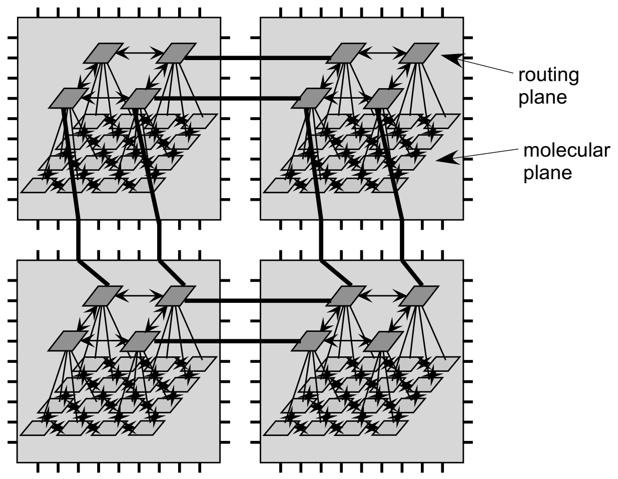

Fig. 8. A multi-chip organism shows the inter-cellular connections.

6 Acknowledgements

This project is funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to this project are supported under grant 00.0529-1 by the Swiss government.

References

[1] ARM: Amba Specification, rev 2.0. Advanced RISC Machines Ltd (arm). http://www.arm.com/armtech/amba_spec, 1999.

[2] Brown, S., Francis, R., Rose, J., Vranesic, Z.: *Field Programmable Gate Arrays.* Kluwer Academic Publishers, 1992.

[3] Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] Eriksson, J., Torres, O., Villa, A. E. P.: Spiking Neural Networks for Reconfigurable POEtic Tissue. In A.M. Tyrrell, P.C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Proc. 5th Int. Conf. on Evolvable Hardware (ICES 03)*, volume 2606 of *LCNS*, pages 165–173, Berlin, 2003, Springer-Verlag.

[5] Moreno Arostegui, J. M., Sanchez, E., Cabestany, J.: An In-system Routing Strategy for Evolvable Hardware Programmable Platforms. In *Proc. 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 157–166. IEEE Computer Society Press, 2001.

[6] Roggen, D., Floreano, D., Mattiussi, C.: A Morphogenetic System as the Phylogenetic Mechanism of the POEtic Tissue. In A.M. Tyrrell, P.C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Proc. 5th Int. Conf. on Evolvable Hardware (ICES 03)*, volume 2606 of *LCNS*, pages 153–164, Berlin, 2003, Springer-Verlag.

[7] Sanchez, E., Mange, D., Sipper, M., Tomassini, M., Perez-Uribe, A., Stauffer, A.: Phylogeny, Ontogeny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware. In T. Higuchi, M. Iwata, and W. Liu, editors, *Evolvable Systems: From Biology to Hardware*, volume 1259 of *LCNS*, pages 33–54, Berlin, 1997. Springer-Verlag.

[8] Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Perez-Uribe, A.: A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-inspired Hardware Systems. *IEEE Transactions on Evolutionary Computation*, 1:1:83–97, 1997.

[9] Tempesti, G., Roggen, D., Sanchez, E., Thoma, Y., Canham, R., Tyrrell, A.M.: Ontogenetic Development and Fault Tolerance in the POEtic Tissue. In A.M. Tyrrell, P.C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Proc. 5th Int. Conf. on Evolvable Hardware (ICES 03)*, volume 2606 of *LCNS*, pages 141–152, Berlin, 2003, Springer-Verlag.

[10] Tyrrell, A.M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno Arostegui, J.-M., Rosenberg, J., Villa, A.E.P.: Poetic Tissue: An Integrated Architecture for Bio-inspired Hardware. In A.M. Tyrrell, P.C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Proc. 5th Int. Conf. on Evolvable Hardware (ICES 03)*, volume 2606 of *LCNS*, pages 129–140, Berlin, 2003, Springer-Verlag.