A Reconfigurable Chip for Evolvable Hardware

Yann Thoma* and Eduardo Sanchez

Swiss Federal Institute of Technology at Lausanne (EPFL), Lausanne, Switzerland
yann.thoma@epfl.ch

Abstract. In the recent years, Xilinx devices, like the XC6200, were the preferred solutions for evolving digital systems. In this paper, we present a new System-On-Chip, the POEtic chip, an alternative for evolvable hardware. This chip has been specifically designed to ease the implementation of bio-inspired systems. It is composed of a microprocessor, and a programmable part, containing basic elements, like every standard Field Programmable Gate Array, on top of which sits a special layer implementing a dynamic routing algorithm. Online on-chip evolution can then be processed, as every configuration bit of the programmable array can be accessed by the microprocessor. This new platform can therefore replace the Xilinx XC6200, with the advantage of having a processor inside.

1 Introduction

Engineers and scientists have much to learn from nature, in term of design capabilities. Living beings are capable of evolution, learning, growth, and self-repair, among others. Each of these fields can serve as inspiration to build systems that are more robust and adaptable. Three life axis define what makes nature a good candidate from which we can draw inspiration: Phylogenesis (P), Ontogenesis (O), and Epigenesis (E).

Phylogenesis is the way species are evolving, by transmitting genes from parents to children, after a selection process. Based on the principles of the neo-darwinian theory, scientists have designed evolutionary algorithms, and more particularly genetic algorithms [1], that are used to solve complex problems for which a deterministic algorithm can not find a solution in an acceptable period of time.

Ontogenesis corresponds to the growth of an organism. In living beings, after fertilization, a single cell, the zygote, contains the genome that describes the entire organism and starts dividing, until the organism is totally created. Ontogenesis takes also care of self-healing, a very important feature of living beings, that prevents them from dying after a light injury. In electronics, self-repair based on ontogenetic principles has been applied to building more robust systems [2,3,4,5].

Finally, epigenesis deals with learning capabilities. A brain, or more generally a neural network, is the way life solved the learning problem. Taking inspiration

* Corresponding author

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 816–827, 2004.

of real neurons, scientists have designed a huge variety of neural networks, to solve different tasks, like pattern recognition [6] and robot learning [7].

These three life axis have often been considered separately for designing systems, or as a conjunction of learning and evolution. Until now, no real POE system has been constructed. The POEtic project is therefore the logical continuity of bio-inspired systems. A new chip has been specially designed to ease the development of such systems. It contains a microprocessor, and a reconfigurable array offering capabilities of dynamically creating paths at runtime.

This paper focuses on the way POEtic, a promising alternative to the XC6200, can be used as a platform for evolvable hardware [8,9]. Next section presents briefly the principles of evolvable hardware and why field programmable gate arrays are good candidates for such systems. Section 3 describes the POEtic chip, with an emphasis on its usefulness for evolvable hardware. Section 4 presents the way POEtic will be used for this purpose, and finally section 5 concludes.

2 Evolvable Hardware

Evolvable hardware (EHW), on the phylogenetic axis, deals with the design of analog or digital circuits using genetic algorithms. This technic replaces an engineer in the design task, and can act in many different areas. For instance, basic systems like adders or multipliers can be built, while robot control can also be generated. EHW processes can be evolved in simulation in many cases, but software implementations are very slow, and cannot always fit real conditions. Therefore, hardware platforms are needed, to generate operating circuits, in case of analog design, and to speed up the entire process, in case of digital design.

2.1 FPGAs and the Xilinx XC6200 Family

Field Programmable Gate Arrays (FPGAs) [10] are digital circuits that can be reconfigured, and thus make them excellent candidates for implementing EHW. Every commercial FPGA is based on a 2-dimensional array of cells, in which it is possible to define the cells' functionalities and the routing. The most widely used for EHW, the Xilinx Virtex XC6200 family, has been utilized in many experiments [11,12,13,14,15], due to its routing implementation based on multiplexers rather than on anti-fuse or memory bits (short circuits can be generated in almost every other types of FPGAs). The architecture of the XC6200 is very simple, with cells based on some multiplexers and a flip-flop. Moreover, the configuration bits arrangement is public, giving a programmer total control over the configuration. Unfortunately, these devices are not available any more, and no equivalent FPGA is available as of today.

The inherent parallelism of FPGAs allows to rapidly test individuals to evaluate their fitness, but a problem remains: the configuration is very slow. One of the last family of Xilinx devices, the Virtex II Pro, embeds a microprocessor that can access a reconfigurable array, but without the capability of reconfiguring

it. The POEtic chip, as explained in the next section, will be a new hardware platform that solves this last drawback.

3 The POEtic Chip

The POEtic chip has been specifically designed to ease the development of bio-inspired applications. It is composed of two main parts: a microprocessor, in the environmental subsystem, and a 2-dimensional reconfigurable array , called the organic subsystem (figure 1). This array is made of small elements, called molecules, that are mainly a 4-input look-up table, and a flip-flop. In the organic subsystem, a second layer implements a dynamic routing algorithm that will allow multi-chip designs, letting the user work with a bigger reconfigurable virtual array.

The next section presents some features of the on-chip microprocessor. The subsequent section describes the reconfigurable array, with a special emphasis on how the different parts of the basic elements can be used to build an EHW system similar to the XC6200.

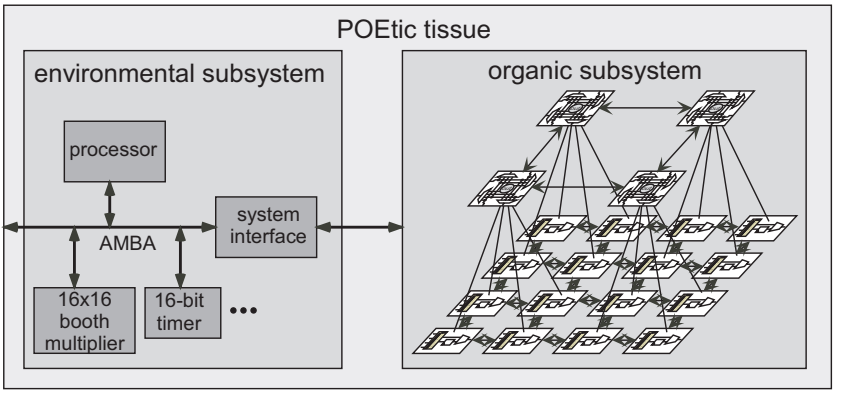

Fig. 1. The POEtic chip, showing the microprocessor and the reconfigurable array. Many elements connected to the AMBA bus, like a second timer, serial and parallel ports, are omitted in order to simplify the schematics. On the right, the organic subsystem shows molecules on the bottom, and routing units on the top.

3.1 The Microprocessor

The microprocessor is a 32-bit RISC processor, specially designed for the POEtic chip. It exposes 57 instructions, two of which give access to a hardware pseudo-random number generator, that can be very useful for evolutionary processes. This small number of instructions limits the size of the processor, leaving more room for the reconfigurable array.

An AMBA bus [16] allows communication with all internal elements, as shown in figure 1, as well as with the external world. It also permits to connect many POEtic chips together, in order to have a bigger reconfigurable virtual array.

The microprocessor can configure the array, and also retrieve its state. The access is made in a parallel manner, the array being mapped on the microprocessor address space. As a result, so that it is very fast to configure, or to partially reconfigure the array, since the configuration of one molecule requires only three write instructions. For instance, when dealing with evolutionary processes, the retrieved state can be used to calculate the fitness of an individual and evolution can be performed by the microprocessor, avoiding fastidious data transmission with a computer.

A C compiler, as well as an assembler, has been developed, letting a user easily write programs for this microprocessor. Furthermore, an API will be supplied, in order to rapidly build a genetic algorithm by choosing the type of crossing-over, the selection process, and so on. Special functions will also simplify the reconfigurable array configuration.

3.2 The Reconfigurable Array

The reconfigurable array is composed of two planes. The first one is a grid of basic elements, called molecules, based on a 4-input look-up table, and a flip-flop. The second one is a grid of routing units, that can dynamically create paths at runtime between different points of the circuit. They implement a distributed dynamic routing algorithm, based on addresses. It can be used to create connections between cells in a cellular system (e.g. a neural network), to connect chips together, or simply to create long-distance connections at runtime (interested readers can see a description of this algorithm in [17]).

The so-called molecules (see figure 2) execute a function, according to an operational mode, defined for each molecule by three configuration bits (for more details, see [17]). The eight operational modes are:

- **4-LUT**: The molecule is a 4-input LUT.
- **3-LUT**: The molecule is divided into two 3-input LUTs.
- **Shift memory**: The molecule is considered like a 16-bit shift register.
- **Comm**: The molecule is divided into a 3-input LUT and a 8-bit shift register.
- **Configure**: The molecule has the possibility of partially reconfigure its neighborhood.
- **Input**: The molecule is an input from the routing plane.
- **Output**: The molecule is an output to the routing plane.
- **Trigger**: This mode is used to synchronize the dynamic routing algorithm.

3.3 Molecular Communication

In addition to its functional part, a molecule contains a switch box for inter-molecular communication. Like in the Xilinx XC6200 family, inter-molecular communication is implemented with multiplexers. This feature, although being

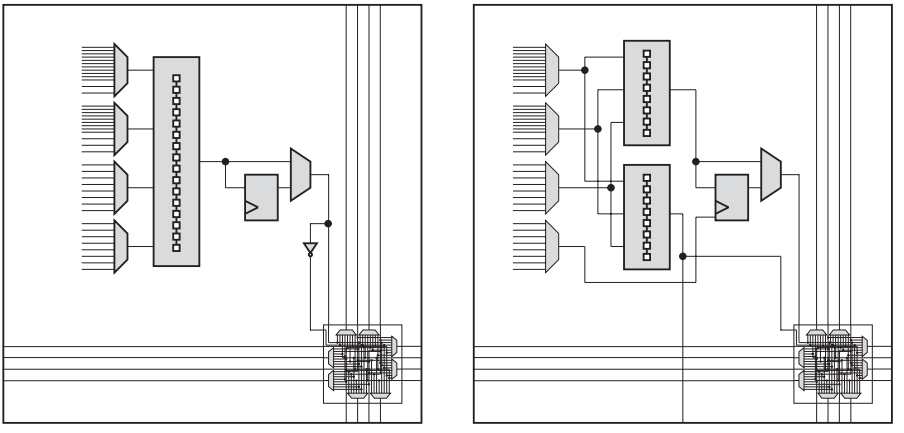

Fig. 2. A "molecule" can act in eight different operational modes, the mode being defined by three configuration bits. The left drawing shows a molecule in 4-LUT mode, while the right depicts a molecule in 3-LUT mode.

more expensive in term of space and delays, avoids short circuits that could happen when partially reconfiguring a molecule, or during an unconstrained evolution process.

Every molecule is directly connected to its four neighbors, sending them its output, while long-distance connections are implemented by the way of switch boxes (figure 3). There are two input lines from each cardinal direction, and two corresponding outputs. Each output can be selected from the six input lines from the other cardinal directions, or from the output of the molecule (or the inverse).

As there are eight possible configurations for an output multiplexer, three configuration bits are necessary for each output. The total lets the switch box being defined by (2 outputs by 4 directions by 3 bits =) 24 bits. These 24 bits could be part of the evolutionary process, or fixed, depending on the kind of system we want to evolve.

For instance, in order to use the POEtic chip like a Xilinx XC6200, every switch box should be configured as in figure 3. By fixing some configuration bits to '0', we can choose to only deal with one line to each direction, as shown in the right of the figure.

In every of its operational modes, a molecule needs up to four inputs. Multiplexers are taking care of the selection of these inputs, like the two first inputs shown in figure 4. An input can basically come from any long-distance line, but each multiplexer has special features. Some can retrieve the flip-flop value, some the direct neighbors output, and so on. By fixing some configuration bits, we can for instance force the selection of a signal coming from N0_in, E0_in, S0_in, W0_in. Therefore, only two bits are necessary to completely define every input.

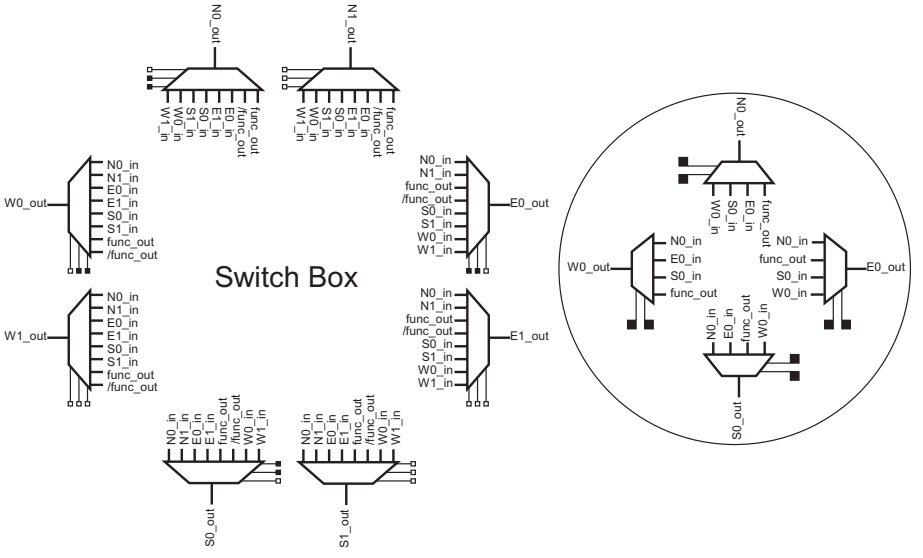

Fig. 3. On the left, the switch box contained in every molecule, as shown in figure 2, and on the right a subset of the possible configurations to reduce the genome size. Black boxes represent configuration bits that can be used for evolution, while white boxes are "don't care bits" for such applications.

This way, every input has the same potential as the others, which would not be the case if every configuration bit could be modified.

3.4 Configuration Bits

One of the advantages of the POEtic chip is the possibility to define any of the 76 configuration bits. These bits are split into five blocks, as shown in table 1. The first bit of each block indicates whether the block has to be reconfigured or not, in case of a partial reconfiguration coming from a neighbor molecule. As mentioned before, the microprocessor can access (read/write) the configuration bits with a 32-bit bus. For EHW, this feature is very important in terms of execution time. Since only two clock cycles are needed for a write and three words of 32 bits define a molecule, the configuration of the entire array or of only a part of it is very fast. In comparison with standard FPGAs, like a Xilinx with JBits [18,19], in which the entire configuration bitstream must be sent each time in serial, the reconfiguration, like the first configuration, is made in parallel, allowing a huge gain in term of time. Moreover, compared to a total reconfiguration, if we only evolve the switch box or the LUT, loading time can be divided by three, as only part of the molecule configuration needs to be reloaded.

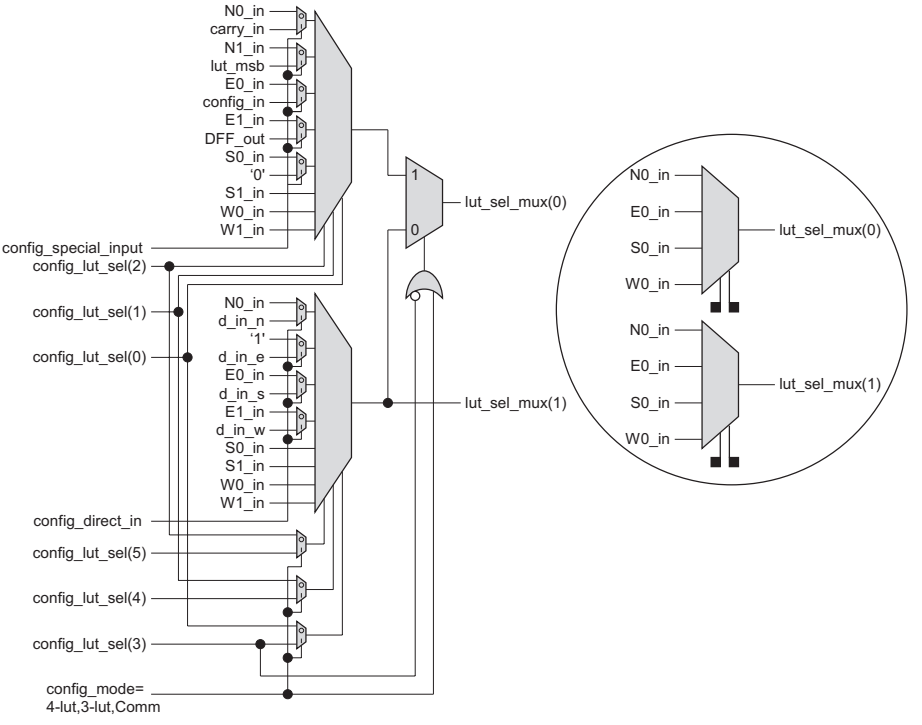

Fig. 4. The two first inputs of the molecule. The signals `config_*` are configuration bits, the two right outputs (`lut_sel_mux(X)`) are the two first inputs of the LUT, and all other signals are inputs that can be selected. The right figure shows a subset of the possible inputs, obtained by fixing some configuration bits (black boxes represent configuration bits that can be used for evolution).

4 Evolvable Hardware on POEtic

In last section, we showed different parts of the reconfigurable array that can be used in an evolvable process. The final chip being not yet available, we will not present experimental results, but concepts that will be used later to demonstrate the full potential of the POEtic chip. First we will have a look at what kind of EHW is supported by POEtic, and secondly, we will describe how we can directly evolve the bitstream as the genome.

4.1 POEtic Evolvable Characteristics

Following the classification developed by Torresen in [20,21], we can now precisely identify the capabilities of POEtic:

- The microprocessor can run a **Genetic Algorithm**.
- The target technology is **Digital**.

Table 1. The five blocks of configuration bits (the first three bits cannot be partially configured by a neighbor molecule).

Number of bits	Description
1	global partial configuration enable
2	configuration input origin
1	lut partial configuration enable
16	lut(15 downto 0) (cf. figure 2)
1	lut inputs configuration enable
14	selection of the lut inputs (cf. figure 4)
1	switchbox partial configuration enable
8x3	3 bits for each of the 8 multiplexers (cf. figure 3)
1	mode partial configuration enable
3	operational mode (cf. section 3.2)
1	other bits partial configuration enable
1	sequential or combinational output
1	flip-flop reset value
1	dff enable used or not
1	clock edge
3	local reset origin
1	local reset enable
1	asynchronous/synchronous reset
1	molecule enable
1	value of the flip-flop

- The architecture applied in evolution can be **Complete Circuit Design**, where building blocks and routing are evolved, or **Circuit Parameter Tuning**, where only configurable parameters are evolved.
- The possible building blocks can be **Gates** (the LUTs), or **Functions** (neurons, ...).
- The evolution is made **Online**, because every individual will be tested using the reconfigurable array.
- The evolution is **On-chip**, as the microprocessor is incorporated into the reconfigurable chip.
- The scope of evolution can be **Static**, or **Dynamic**, depending on the type of application.

POEtic, with its dynamic routing capability, could show function level evolution that involves sine generators, adders, multipliers, artificial neurons, or others. However, in this paper we only present gate level evolution, that involves OR/AND gates, or in our example, look-up tables.

Basically, an unconstrained evolution could be executed with the entire configuration bitstream, since it is impossible to create a short-circuit. However, 76 bits for each molecule signify a huge genome, if, for instance, we deal with a 10 by 10 array. Therefore, in many cases, only part of the bitstream will be evolved, in order to reduce the search space.

The experiments made by Thompson using the Xilinx XC6200 are based on the same principle of avoiding to evolve the entire bitstream. They only deal with 18 bits per element, in order to evolve oscillators, for instance. The same types of applications could be resolved with 22 bits using the POEtic chip.

4.2 Genome Representation

In the approach chosen in this paper, we evolve a system at the gate level, by evolving the routing or the function of molecules. Therefore, it is natural to directly evolve the configuration stream of the chip. Since there are 76 configuration bits, and the bus has a width of 32 bits, only three words define a molecule. In order to evolve routing and functionality, we do not want to evolve the entire bitstream, but only part of it. By using very simple logical operations we can modify the entire genome, without modifying fixed parts, as shown in figure 6.

In our example, the routing uses half of its capabilities, with the subset shown in figure 3. The molecule inputs are the same as shown in figure 4, and the operational mode is fixed to the 3-LUT mode. Therefore, the functionality can be any 3 inputs function. This case corresponds to an evolution of the basic cells of figure 5.

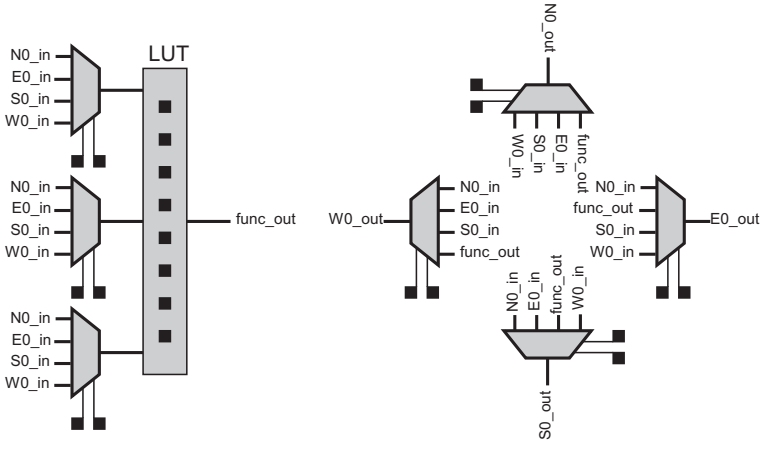

Fig. 5. The basic element, subset of the molecule, that can be evolved, defined by only 22 bits. Black boxes are configuration bits that are evolved.

The full genome is composed, for each molecule, of 96 bits (3x32), 76 defining configuration bits. However, in our example, only 22 bits really represent information used to define the phenotype, the 74 other bits being fixed. Compared to the 18 bits used by Thompson with a XC6200, we deal with 4 more bits, because we totally evolve the look-up table content, rather than just some multiplexers. This way the genome is bigger, but each element has more flexibility.

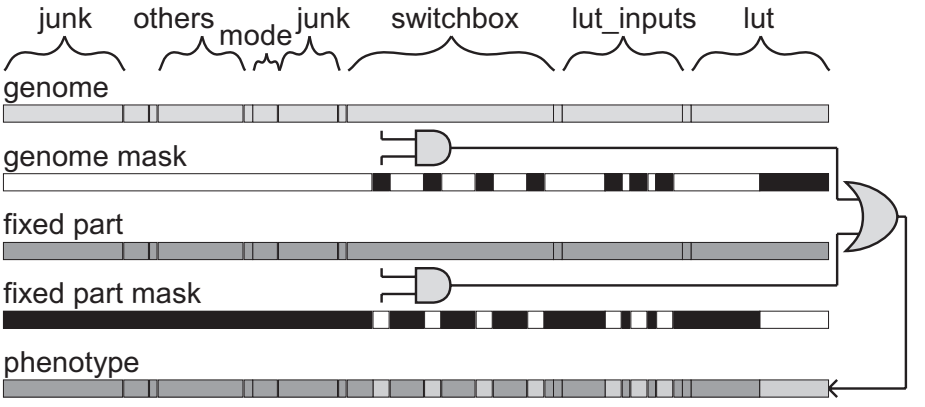

Fig. 6. This figure depicts the way a phenotype can be generated, from a variable genome and a fixed part. A line represents the 3x32=96 bits where 20 are unused bits and 76 are configuration bits of a molecule. These 76 bits are divided into the five blocks described in table 1. The first line is the genome evolved using crossing-over and mutation. The genome mask is used in a logical "and" operation with the genome. It contains '1' at every place the genome is defined by the evolutionary algorithm. Only 22 bits are relevant to define the phenotype: 8 bits for the switch box, 6 bits for the molecular inputs, and 8 bits for the 3-input LUT. The fixed part, combined with its mask (the inverse of the genome mask) corresponds to every configuration bits not defined by the evolution. By simply using an "or" operation on the two results of "and" operations, we obtain the phenotype that is the real configuration of the molecule.

In the evolution process, crossing-over and mutation will be applied to the entire configuration stream, and the very simple logical operations will erase parts of it with the fixed bits. This way, there is no need to use complex transformation, from a smaller virtual bitstream to a real one, saving execution time. Moreover, the fixed parts can be viewed like junk DNA in living beings, in which a large part of the genome is simply unused.

5 Conclusion

In this paper we presented how the POEtic chip can be useful as an EHW platform. The conjunction of a custom microprocessor and a reconfigurable array is perfect to implement an on-chip evolution process. Moreover, compared to a Xilinx Virtex II Pro where there is also a microprocessor, the advantage of POEtic is the fact it is aware of the entire memory map of the configuration bits, and that the microprocessor can configure the chip. Finally, compared to a Xilinx XC6200, POEtic has the advantage of having a microprocessor inside, allowing fast configuration of the reconfigurable array. Table 2 summarizes the features of the XC6200, the Virtex II Pro, and POEtic.

At present, a test chip is being fabricated. After functional tests on this small chip (it only contains the microprocessor and 12 molecules), the final POEtic

Table 2. Comparison of features useful for EHW between a XC6200, a Virtex II Pro and the POEtic chip.

Feature	**Xilinx XC6200**	**Xilinx Virtex II Pro**	**POEtic**
Impossible to short-circuit	Yes	No	Yes
Processor inside	No	Yes	Yes
Processor accessing the configuration bits	No	No	Yes
Bitstream detail available	Yes	No	Yes
Dynamic routing	No	No	Yes

chip, containing about 200 molecules, will be designed and sent to fabric. As soon as it is available, the concepts described in this paper will be tested with the real hardware, to show the promising usefulness of the POEtic chip as a powerful replacement of the Xilinx XC6200 for EHW.

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. Holland, J.: Genetic algoritms and the optimal allocation of trails. In: SIAM Journal of Computing. Volume 2:2. (1973) 88–105
2. Kitano, H.: Building complex systems using developmental process: An engineering approach. In: Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98). Volume 1478 of LNCS, Berlin, Springer Verlag (1998) 218–229
3. Mange, D., Sipper, M., Stauffer, A., Tempesti, G.: Towards robust integrated circuits: The embryonics approach. In: Proceedings of the IEEE. Volume 88:4. (2000) 516–541
4. Ortega, C., Tyrell, A.: MUXTREE revisited: Embryonics as a reconfiguration strategy in fault-tolerant processor arrays. In: Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98). Volume 1478 of LNCS, Berlin, Springer Verlag (1998) 206–217
5. Pearson, H.: The regeneration gap. Nature **414** (2001) 388–390
6. Dayhoff, J.: Pattern recognition with a pulsed neural network. In: Proceedings of the conference on Analysis of neural network applications, New York, NY, USA, ACM Press (1991) 146–159
7. Grossmann, A., Poli, R.: Continual robot learning with constructive neural networks. In Birk, A., Demiris, J., eds.: Proceedings of the Sixth European Workshop on Learning Robots. Volume 1545 of LNAI, Brighton, England, Springer-Verlag (1997) 95–108
8. Gordon, T.G.W., Bentley, P.J.: On evolvable hardware. In Ovaska, S., Sztandera, L., eds.: Soft Computing in Industrial Electronics, Heidelberg, Physica-Verlag (2002) 279–323

9. Yao, X., Higuchi, T.: Promises and challenges of evolvable hardware. IEEE Trans. on Systems, Man, and Cybernetics – Part C: Applications and Reviews **29** (1999) 87–97
10. Brown, S., Francis, R., Rose, J., Vranesic, Z.: Field Programmable Gate Arrays. Kluwer Academic Publishers (1992)
11. Fogarty, T., Miller, J., Thomson, P.: Evolving digital logic circuits on xilinx 6000 family fpgas. In Chawdrhy, P., Roy, R., Pant, R., eds.: Soft Computing in Engineering Design and Manufacturing, London, Springer Verlag (1998) 299–305
12. Huelsbergen, L., Rietman, E., Slous, R.: Evolution of astable multivibrators *in Silico*. In Sipper, M., Mange, D., Pérez-Uribe, A., eds.: ICES'98. Number 1478 in Lecture Notes in Computer Science, Berlin Heidelberg, Springer-Verlag (1998) 66–77
13. Tangen, U., McCaskill, J.: Hardware evolution with a massively parallel dynamically reconfigurable computer: Polyp. In Sipper, M., Mange, D., Pérez-Uribe, A., eds.: ICES'98. Volume 1478 of LNCS, Berlin Heidelberg, Springer-Verlag (1998) 364–371
14. Thompson, A.: Silicon evolution. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, MIT Press (1996) 444–452
15. Thompson, A.: On the automatic design of robust electronics through artificial evolution. In Sipper, M., Mange, D., Pérez-Uribe, A., eds.: ICES'98. Volume 1478 of LNCS, Berlin Heidelberg, Springer-Verlag (1998) 13–24
16. ARM: AMBA specification, rev 2.0. advanced RISC machines Ltd (ARM). http://www.arm.com/armtech/AMBA_Spec (1999)
17. Thoma, Y., Sanchez, E., Arostegui, J.M.M., Tempesti, G.: A dynamic routing algorithm for a bio-inspired reconfigurable circuit. In Cheung, P.Y.K., Constantinides, G.A., de Sousa, J.T., eds.: Proc. of the 13th International Conference on Field Programmable Logic and Applications (FPL'03). Volume 2778 of LNCS, Berlin, Heidelberg, Springer Verlag (2003) 681–690
18. Guccione, S.A., Levi, D., Sundararajan, P.: Jbits: A java-based interface for reconfigurable computing. In: 2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD). (1999)
19. Hollingworth, G., Smith, S., Tyrell, A.: Safe intrinsic evolution of virtex devices. In: proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware. (2000) 195–204
20. Torresen, J.: Possibilities and limitations of applying evolvable hardware to real-world applications. In Hartenstein, R., Grünbacher, H., eds.: FPL 2000. Volume 1896 of LNCS, Berlin Heidelberg, Springer-Verlag (2000) 230–239
21. Torresen, J.: Evolvable hardware as a new computer architecture. In: Proc. of the International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet. (2002)