International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:12, 2007

Multi-board Run-time Reconfigurable Implementation of Intrinsic Evolvable Hardware

Cyrille Lambert, Tatiana Kalganova, Emanuele Stomeo, and Manissa Wilson

***Abstract*—**A multi-board run-time reconfigurable (MRTR) system for evolvable hardware (EHW) is introduced with the aim to implement on hardware the bidirectional incremental evolution (BIE) method. The main features of this digital intrinsic EHW solution rely on the multi-board approach, the variable chromosome length management and the partial configuration of the reconfigurable circuit. These three features provide a high scalability to the solution. The design has been written in VHDL with the concern of not being platform dependant in order to keep a flexibility factor as high as possible. This solution helps tackling the problem of evolving complex task on digital configurable support.

***Keywords*—**Evolvable Hardware, Evolutionary Strategy, multi-board FPGA system.

I. Introduction

EVOLVABLE Evolvable hardware (EHW) [1] is the combination of a configurable device and an evolutionary algorithm (EA) [2]. The EA modify the data content which composes the bit string of the configurable device in order to evolve the circuit until it fulfills a task. EHW had been introduced to be applied to real-world applications but up to date only few solutions can deal with relatively large solutions such as [3][12][13]. And nowadays it is mainly seen as a way to automatically design circuits that can be digital, mixed or analogue. Different types of intrinsic EHW implementations have been developed, some based on analogue reconfigurable supports made of transistors [4][5][6] or mixed support (both digital and analogue) application specific integrated circuits (ASIC) [3] others on digital support like programmable logic array [7] or reconfigurable systems based on processors [8] but most of the digital system development has been made on field programmable gate arrays (FGPA) and almost exclusively on Xilinx products. It appears that this device prevails among the others supports mainly due to its high flexibility. Indeed an FPGA can be reconfigure almost an infinite number of times and this for a reasonable price compare to an ASIC. An FPGA does not need to be design it is ready to be configured but FPGA are not perfect and some major drawbacks for EHW lie in the impossibility to reconfigure an FPGA from itself. The important configuration time required is also a major inconvenience to have a successful evolution in the shortest delay. An evolution process in most of the case needs several trials to reach the final solution (cf. Section 2 for further description) therefore in an intrinsic EHW system each trial is downloaded inside the FPGA, in other words the FPGA is configured and the interest is to avoid wasting time during this phase. The method introduced by Layzell to have a configurable system on a top of a FPGA [9] helps bypassing these problems. The FPGA will not be reconfigured for each new generation but the virtual circuit, thus a high amount of time is gained. Some others systems has been carried out following this method such as [10][11] and as much as the authors know they successfully evolved small tasks (up to a ten of inputs). Sekanina has introduced a way to implement evolvable IP cores [12] thanks to a virtual reconfigurable circuit (VRC) constituted of configurable functional blocks (CFB). In [13] another VRC has been introduced where the array is infinitely extensible regarding the limitation that the support (i.e. FPGA) introduces. These two VRC are extremely promising. In Sekanina one a CFB has been designed as a configurable logic block (CLB) that is present in the Xilinx FPGA [14] so a set of functions can be stored inside a CFB. His VRC is made of an array of column of CFB and the connections between these columns are configurable. In Haddow et al. VRC the Sblock approach allows to configure any connections between each sblock but the functions set capacity seems to limited by the size of the FPGA LUT used to implement the VRC. Therefore if both features are merged and the possibility to configure each cell individually, we could reach a very flexible and highly scalable system.

Section 2 explains the evolutionary algorithm used in the proposed system. Section 3 introduces the proposed solution and details the main components of it. It follows an implementation cost study of the proposed system in section 4 and the document is ended by a conclusion.

II. Evolutionary Algorithm

It has been stated to use a (1+λ) evolution strategy (ES) because it has been extensively tested for its performances in [15][16][17][18] has been chosen (Fig. 1) where λ reflects the number of individuals composing a population. It has been evaluated that an ES gives good results for an evolution process in a small number of generations for a population constituted of thousands of individuals or in a much higher

Manuscript received June 1, 2006. This work was supported in part by the EPSRC under grant number GR/S17178/.

C. Lambert, T. Kalganova, E. Stomeo and M. Wilson are with the School of Engineering and Design, Brunel University (phone +44 1895 266 752; e-mail: Tatiana.Kalganova@brunel.ac.uk).

number of generations but for a small population [19]. For obvious reason of overall cost the first statement cannot be realized. Therefore it has been established to use a small population and λ equals to five have shown some interesting results if we refer to the document previously cited.

A. (1+ λ) Evolution Strategy

The ES works as follow, after a first generation has been randomly created, each chromosome (= individual) is evaluated. It results one fitness value for each chromosome, the best of them is kept in a memory called best chromosome memory. The fittest chromosome is therefore tested to know if it fully answers to the task i.e. fitness value equals to 100%. Else the best chromosome is mutated five times, one time per new chromosome. It results a mutated population also called new population that replaces the previous one. The process carries on until the fitness value is equal to 100%, the number of generation reaches a maximum allowed number for instance 500,000 or for any other condition introduced by the user for instance in our case a stalling effect of the fitness value i.e. no improvement of the fitness value for a certain amount of time such as 10% of the maximum allowed number of generations.

B. Fitness Evaluation

Each time an individual have been configured by a new generation of population a fitness evaluation is made. This will indicate which chromosome gives the best answer to the task and help to decide if the evolution process as to carry on or if the answer has been found. The evaluation results from a comparison between a set of desired outputs and the outputs of each individual.

In the proposed system each chromosome fitness values are computed one by one. When all of them are known, the best one undergoes a selection with the best chromosome fitness

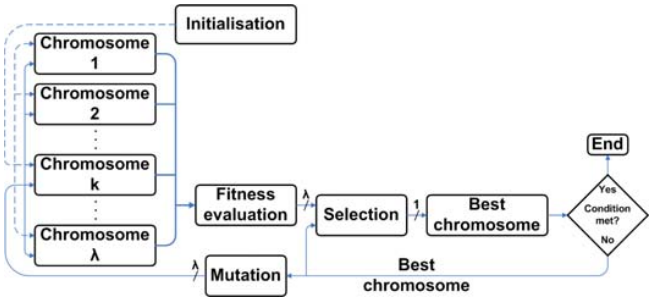

Fig. 1 (1+λ) evolution strategy

value kept in memory. The following equation (1) illustrates the fitness function used in the MRTR system:

$$F = \sum_{addr=0}^{y} \sum_{out=0}^{x} d \quad \text{where } d = \begin{cases} 0 & if\ o \neq e \\ 1 & if\ o = e \end{cases} \tag{1}$$

To have a significant fitness value a set of inputs are applied to each individuals of the MRTR. The resulting outputs are compared with another set of desired outputs. These two different sets are located in two memories and organized in truth tables. The fitness value F is thereby expressed by the sum of all the differences d between the desired output e and the output given by an individual o. If the individual output is different than the desired one the fitness value does not change else it is incremented by 1. The fitness value is computed for all the outputs (*out*) of each individual through the two truth tables (*addr*).

III. Description of the Architecture

A detailed description of the system and its main components are exposed in this section.

A. Overview of the MRTR

To have a flexible system it has been decided to use a design coded in VHDL without using any feature belonging to a dedicated FPGA but rather to write a code as generic as possible in order that the system can be implemented on most of the FPGA provided on the market. Moreover to keep a factor of scalability as high as possible the author decided to plan an implementation where each reconfigurable array is on one FPGA. It allows having a high scalability only limited by the size of the support.

The multi-board run-time reconfigurable system is composed of:

- an evolution strategy (ES),
- a fitness evaluation,
- a multiplexer block (Sort) presenting the outputs of each target to the ES for the fitness evaluation,
- two memory blocks containing the values of the input (ITT) and output truth tables (OTT),
- and finally five reconfigurable arrays.

The Fig. 2 exposes the overview of the MRTR and shows the main data exchanges between the components that composed the system.

The ES, ITT, OTT and Sort components are planned to be implemented on a single FPGA. Each reconfigurable circuit will be implemented on one FPGA. Therefore the whole system will contain six FPGA. Moreover the RC design has been carried out with the VRC as pattern. Bear in mind that the main drawback of this approach relies on the fact that it is very greedy in term of CLB.

B. Reconfigurable Circuit

Each of one the five reconfigurable circuits are made as an array of configurable cells of *r* rows and *c* columns (see Fig. 3).

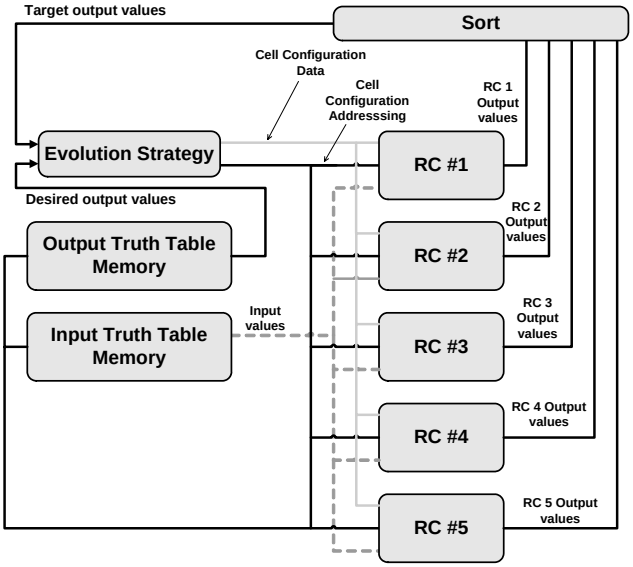

Fig. 2 Multi-board Run-time reconfigurable system overview. The reconfigurable circuits (RC) are the individuals composing the population and are at the number of 5. A sorting block that is in fact a multiplexer helps synchronizing the RC outputs sending to the ES. The fitness evaluation is made thanks to the application of the ITT to each RC and the RC outputs are compared with the OTT values (desired values)

The Fig. 4 illustrates in detailed the specificities of the three kind of RCell. An RCellA is always located in the first column while an RCellC can solely be located in the last column. An RCellB has the inverse restriction it cannot be a cell of the first or last column of the array. The routing selectors create some connections with the inputs of an RCell. An RCellB for instance can be linked to an output of an RCellA or of a previous RCellB but as well to an input of the RC. In summary only the RCellC cannot have their inputs connected to the inputs of the cell array. The function selector has a similar role than a Xilinx FPGA LUT i.e. it can be seen as a small memory that contains a set of function (for instance AND, OR, XOR, NOT…). To configure the function of an RCell A or B means to choose one among the pre-loaded functions of their function selector.

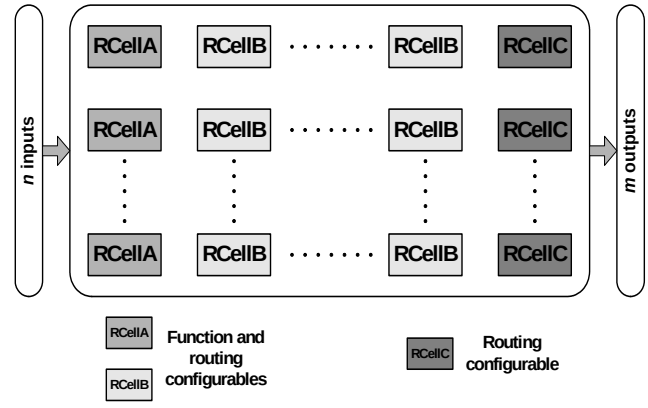

Fig. 3 Reconfigurable circuit overview. 3 types of reconfigurable cell (RCell). An RCellA is function configurable and can be connected to any of the n inputs of the RC. An RCellB is also function configurable and can be linked to any cell that is located in the previous column while an RCellC is solely routing configurable so as to be linked to any outputs of the previous RCellB column and to the m outputs of the RC

As it has been written in the introduction, the RCs are configured by the way of a bit string provided by the ES. The bit string organisation has to reflect the one of the RCs. The

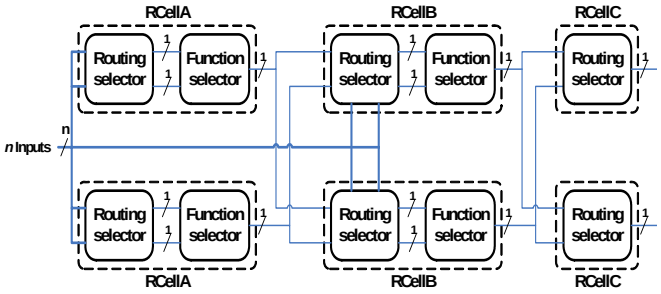

Fig. 4 Configurable circuit architecture made of 3 columns and 2 rows with n inputs and 2 outputs. The inputs and outputs are all on 1 bit

Fig. 5 exposes how the RCells are represented in the bit string. As introduced earlier in the document each cell can be addressed independently, partial reconfiguration. Then this feature is translated in the bit string by the data field called *Address*. In order to address a cell the column and row of this one has to be indicated in *@Col* and *@Row*. *CConf1* and *CConf2* are used to know if the routing selector is dealing with the outputs of the previous cell or with the inputs of the RC. The *Functionality & routing* field is used to configure the routing and function selectors. *Input1* and *Input2* are the configuration data of the routing selector while the function selector receives the data from *Function*. The size of each field has been chosen to have an interesting panel of possible RC shape and size for the simulations of the MRTR.

Therefore the full size of the bit string per chromosome will be a multiple of the bit string for one cell by the number

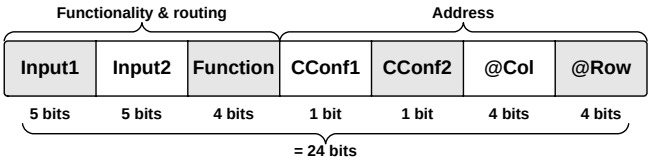

Fig. 5 Bit string representation of a configurable cell (RCell) also called RCell bit string. This organisation is common to the three types of cell

of cell that composes the RCs. The size of a RC is obviously the number of row by the number of column of this one, see equation (2).

$$N_{BitSize} = N_{rows} \times N_{cols} \times RCell\ \ bit\ \ string \quad (2)$$

where $N_{BitSize}$ is the bit string size, N_{rows} and N_{cols} are the number of rows and columns in the reconfigurable target. The size of RCell bit string is 24 bits.

The bit string format can easily be modified without interfering in the behavior of the system, by adjusting the mutation process to the new format.

Each cell can be individually configured, this allows a high flexibility and thus it offers a good scalability. Indeed it is possible to increase the size of the configurable array adding other cells. The size limit of this array is imposed by the physical support therefore the size of the FPGA on which the array is implemented. Furthermore, this feature joined with a variable chromosome length permit to partially configuring an array as shown in Fig. 4. Bear in mind that modify the size of the full bit string length is not more than adding or subtracting some *RCell bitstrings*. A column of cell, a group of cell or some isolated cells can thus be configured. Therefore the finest grain of configuration is a cell.

IV. Estimation and First Results

In this section, different implementation cost estimation is introduced. It shows the versatility of the MRTR solution. Then it is followed by some results pre-synthesis of some simulations made with the MRTR.

A. Implementation Cost

The MRTR has two main parts:

- the ETP which is the ES along with SORT, ITT and OTT,
- and the reconfigurable circuits.

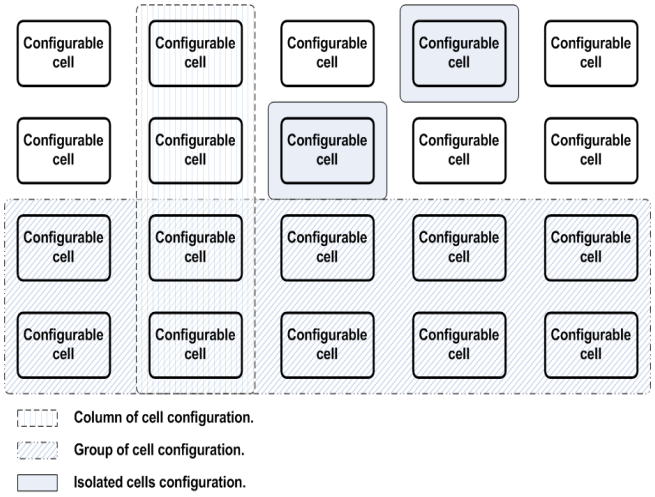

Fig. 6 Partial configuration modes. The configuration of a single cell, of a column of cell, a group of cells or the whole array are the configuration modes offered to the user

TABLE I
Implementation Cost for the Two Main Elements of the MRTR System. ETP Contains the ES, the Input and Output Truth Tables and the Sort Block

Block	Resource	Used resource
Reconfigurable circuit	Slices	10147
	Slice Flip Flops	7896
	4 Input LUTs	18053
ETP	Slices	2093
	Slice Flip Flops	2007
	4 Input LUTs	3653

This synthesis results are for a Xilinx XCV1000 FPGA with a RC made of 10 rows and 11 columns of RCell with an optimation goal set on area and an effort set on 2.

TABLE II
Implementation Cost for Various Implementation Configurations on Different FPGA. Each RC are made of 11 Columns by 10 Rows of RCell

Block	Slices		Percentage	FPGA
	Used	Available		
Reconfigurable circuit * 5	25175	32448	78%	*XCV3200E*
RTR (Full system)	52724	67584	78%	*XC4VLX160*
	27268	32448	84%	*XCV3200E*
Reconfigurable circuit	10147	12288	82%	*XCV1000E*
	10025	14336	70%	*XC2V3000*
ETP	2093	3072	70%	*XCV300*
		2352	89%	*XC2S200*
	2079	3072	68%	*XC2V500*
	2079	6144	34%	*XC4VLX15*

The Table I show the amount of slices, slice flip flops and 4 input LUTs needed to implement one RC and the ETP on a Xilinx XCV1000 FPGA.

Several implementations are possible (cf. Table II) such as one target per FPGA which will require having five XCV1000 (82 % of slices used) and the ETP on a XCV300 (70 % of slices used). It is obviously an expensive implementation however other solutions are conceivable for instance an expensive one could be to implement the whole system on a single FPGA i.e. XC4VLX160 (78% of slices used). Any other solution made of FPGA that can contain this configuration or an upper one is also suitable.

B. Results of Simulation

Some simulations have been run but none of them have reached the final solution yet. These simulations were:

- RC 10 rows by 11 columns, clocks 63 MHz and 80MHz, 1 and half multiplier i.e. 1 bit by 2 bits and results on 2 bits.
- RC 10 rows by 11 columns, clocks 63 MHz and 80 MHz, 2 by 2 multiplier i.e. 2 bits by 2 bits and result on 4 bits.

TABLE III
Initialization, Sending and Mutation Processes Durations

RC size (number of rows * number of columns)	Clocks frequencies	Task to evolve	Number of bits			Duration		
			Input 1	Input 2	Output	Initialisation	Sending	Mutation
10 * 6	63 MHz, 80 MHz	1 and half multiplier	1	2	2	4.9 us	4.86 us	26.42 us
10 * 11	63 MHz, 80 MHz	2 by 2 multiplier	2	2	4	8.8 us	8.8 us	48.3 us

However it is possible to give the durations of the initialization process, the sending process and the mutation process (Table III). It results that the sending of the bit string to the targets takes only 15% of a generation process. This stressed the interest of having a virtual RC in order to speed up the sending process. Indeed as stated in [14][20][21][22] the maximum frequency range allowed to configure an FPGA is between 50MHz and 100 MHz. Knowing that it is sent by bit frames with some pad bit the configuration time would be much higher than in the current solution (Refer to [23] for more detail).

V. Conclusion

A multi-board run-time reconfigurable system has been introduced. Several implementations were proposed. The variable chromosome length, the multi-board approach, and the partial configuration of the reconfigurable circuit offer interesting features to obtain a scalable solution in order to evolve relatively complex tasks. The authors are currently working on the implementation of the system on FPGA. Nevertheless the intermediate results of the simulations are showing us the legitimacy of the solution. Indeed the speed effectiveness and the flexibility and scalability are drastically increased compared to the existing intrinsic digital EHW solutions comporting a virtual RC and an eventual intrinsic EHW working on FPGA without virtual RC.

References

[1] Yao, X., Higuchi, T.: Promises and Challenges of Evolvable Hardware. IEEE Trans. Systems, Man and Cybernetics, Part C, Vol. 29 (1999) 87 – 97
[2] Goldberg, D. E.: Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Incorporated, Reading, Massachusetts (1989)
[3] Macias, N. J.: The PIG Paradigm: the Design and Use of a Massively Parallel Fine Grained Self-Reconfigurable Infinitely Scalable Architecture. Proc. of the First NASA/DoD Conf. on Evolvable Hardware, 19-21 (1999) 175 – 180
[4] Ozsvald, I.: Short-Circuiting the Design Process: Evolutionary Algorithms for Circuit Design Using Reconfigurable Analogue Hardware. Masters Thesis (1998)
[5] Stoica, A., Keymeulen, D., Vu, D., Zebulum, R., Ferguson, I., Daud, T., Arsian, T., Xin, G.: Evolutionary Recovery of Electronic Circuits from Radiation Induced Faults. CEC2004 conf. on Evolutionary Computation, Congress on, Vol.: 2, 19-23. Vol.2 (2004) 1786 – 1793
[6] Langeheine, J., Meier, K., Schemmel, J., Trefzer, M.: Intrinsic Evolution of Digital-to-Analog Converters Using a CMOS FPTA Chip. Proc. of the Sixth NASA/DoD Conf. on Evolvable Hardware, 24-26 (2004) 18 – 25
[7] Torresen, J.: Evolving Both Hardware Subsystems and the Selection of Variants of Such Into An Assembled System. In proc. of 16th European Simulation Multiconference. Darmstadt, Germany (2002) 451 – 457
[8] Baumgarte, V., May, F., Nückel, A., Vorbach, M., Weinhardt, M.: PACT XPP - A self-Reconfigurable Data Processing Architecture. Presented at ERSA, Las Vegas, NV, (c) CSREA Press (2001)
[9] Layzell, P.: Reducing Hardware Evolution's Dependency on FPGAs. In proc. of MicroNeuro'99, 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-inspired Systems, IEEE, Computer Society, CA (1999) 171 – 178
[10] Friedl, S., Sekanina, L.: The First Circuits Evolved in a Physical Virtual Reconfigurable Device. In: Proc. of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Bratislava, SK, SAV. ISBN 80-969117-9-(2004) 35 – 42
[11] Glette, K., Torresen, J.: A Flexible On-Chip Evolution System Implemented on a Xilinx Virtex-II Pro Device. ICES (2005) 66 – 75
[12] Sekanina, L.: Towards Evolvable IP Cores for FPGAs. Proc. of the Fifth NASA/DoD Conf. on Evolvable Hardware, Los Alamitos, USA, ICSP, ISBN 0-7695-1977-6 (2003) 145 – 154
[13] Tufte, G., Haddow, P. C.: Towards Development on a Silicon-based Cellular. Computing Machine, Natural Computing. Vol. 4, Issue 4 (2005) 387 – 416
[14] Xilinx: Virtex 2.5V FPGA Detailed Functional Description. Version 2.8.1 (2002)
[15] Bäck, T., Hoffmeister, F., Schwefel, H. P.: A survey of evolutionary strategies. In R. Belew and L. Booker, editors, Proc. of the 4th International Conference on Genetic Algorithms, San Francisco, CA, 1991. Morgan Kaufmann (1991) 2 – 9
[16] Schwefel, H. P.: Numerical Optimization of Computer Models. John Wiley & Sons, Chichester, UK (1981)
[17] Miller, J.: An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Programming Approach. In Proc. of the Genetic and Evolutionary Computation Conference. Volume 1, Orlando, USA (1999) 1135 – 1142
[18] Kalganova, T., Miller, J.: Evolving More Efficient Digital Circuits by Allowing Circuit Layout Evolution and Multi-objective Fitness. Proc. of the First NASA/DoD Conf. on Evolvable Hardware, IEEE Computer Society (1999) 54 – 63
[19] Kalganova, T.: Evolvable Hardware Design for Combinational Logic Circuits. PhD thesis, School of Computing, Napier University, Edinburgh, UK (2000)
[20] Xilinx: Virtex-E 1.8V FPGA Detailed Functional Description. Version 2.8 (2006)
[21] Xilinx: Virtex-II Complete Data Sheet. Version 3.4 (2005)
[22] Xilinx: Virtex-4 Data Sheet: DC and Switching Characteristics. Version 2.12 (2006)
[23] Lambert, C., Kalganova, T., Stomeo, E.: FPGA-based Systems for Evolvable Hardware. ICCS'06 Vienna, Austria, Volume 12, ISBN 975-00803-1-9 (2006)

Emanuele Stomeo received a Laurea degree in electronic engineering from Politecnico di Torino, Turin, Italy in 2003. He is currently working towards a PhD in computer science and engineering at Brunel University, West London, UK. From 2000 to 2003 he studied at RWTH Aachen University, Germany where he pursued specializations in image processing and digital design.

He carried out his Master Thesis work at Philips Research Laboratories, Aachen, Germany in 2002-2003. He is currently a member of the Bio-Inspired Intelligent Systems Research Group at Brunel University, West London, UK.

His research interests are in evolvable hardware, evolutionary computation, design of digital circuits and bioengineering applications.

Tatiana Kalganova received MSc degree from Belarusian State University of Informatics and Radioelectronics, Belarus in 1994 and PhD degree from Napier University, UK in 2000.

In August 2000 she has joined Electronic and Computer Engineering Department, Brunel University. Her research interests are evolvable hardware, ant colony algorithms, scalability in AI systems.

She was awarded a personal grant from the Education Ministry of the Republic of Belarus for distinctive achievements in the field of exact sciences in 1997, and a grant from the International Soros Science Education Program (ISSEP) for distinctive achievements in the field of exact sciences in 1996.

Cyrille Lambert received a diplôme d'éducation supérieure spécialisée in microelectronic engineering from Pierre et Marie Currie University, Paris, France in 2000.

After spending three years in the industry as a digital design engineer he joined in 2003 the computer science and engineering department at Brunel University, West London, UK. He is currently working toward the PhD. degree as a member of Bio-Inspired Intelligent Systems at Brunel University, West London, UK.

He carried out his Master Thesis work at the Swiss Centre for Electronics and Microtechnology, Inc., Neuchâtel, Switzerland in 1999-2000.