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

FPGA-based Systems for Evolvable Hardware

Cyrille Lambert, Tatiana Kalganova, and Emanuele Stomeo

***Abstract*—**Since 1992, year where Hugo de Garis has published the first paper on Evolvable Hardware (EHW), a period of intense creativity has followed. It has been actively researched, developed and applied to various problems. Different approaches have been proposed that created three main classifications: extrinsic, mixtrinsic and intrinsic EHW. Each of these solutions has a real interest. Nevertheless, although the extrinsic evolution generates some excellent results, the intrinsic systems are not so advanced. This paper suggests 3 possible solutions to implement the run-time configuration intrinsic EHW system: FPGA-based Run-Time Configuration system, JBits-based Run-Time Configuration system and Multi-board functional-level Run-Time Configuration system. The main characteristic of the proposed architectures is that they are implemented on Field Programmable Gate Array. A comparison of proposed solutions demonstrates that multi-board functional-level run-time configuration is superior in terms of scalability, flexibility and the implementation easiness.

***Keywords*—**Evolvable hardware, evolutionary computation, FPGA systems.

I. Introduction

EVOLVABLE hardware [1][2] is a relatively new discipline that allows to automatically reconfigure electronic chip. The chip configuration is under control of evolutionary algorithm [3]. Since Hugo de Garis has introduced EHW to the scientist community in 1992 several solutions of EHW have emerged. Hugo de Garis introduced in [1] the original idea of combining the reconfigurable hardware and evolutionary algorithms (EA). The required functionality of reconfigurable hardware (target) is generated during evolutionary process using EA. The quality of each individual located in the "target" is evaluated using fitness function. It has been stated that ideally these two parts have to be located on a common chip and perform at hardware speed. Although, the author had this idea since he has been up on the existence of the Field Programmable Gate Array (FPGA) it doesn't mean that it is the only possibility to implement such a system.

Some projects using an Application Specific Integrated Circuit (ASIC) as platform have been designed. Those include the Processing Integrated Grid (PIG) [4], a reconfigurable hardware constituted of transistors [5][6]. Although a number of EHW-oriented platforms have been introduced recently, up to date the most of the research has been carried out on widely manufactured chips such as Programmable Logic Array (PLA) [7] or FPGA in order to keep the factor of flexibility as high as possible. It has also been considered an alternative solution: a reconfigurable system based on a processor, the PACT eXtreme Processing Platform (XPP) [8] but this solution will not be discussed in the present paper mainly due to the fact the XPP is not a widely manufactured chip. Nevertheless, this final solution is based on manufactured specific platform. Currently Xilinx is the most popular platform for implementation of EHW system [1][9][10] that attracted large number of researchers because of the flexibility that incorporated inside of chip. With development of EHW systems over the past decade, the development of new FPGA configurations have been carried out. The new FPGA designs featured not only with significant change in chip structure, such as introduction of JBits, but also in limitation of accessible information about chip performance as it will be shown later. Besides of this, a number of chip designs have been taken out from manufacturing process. This resulted in necessity for development of new FPGA-based EHW structures or introduction of generalised structure that can be acceptable for the most FPGA-like systems.

This research aims to investigate the potential problems that can be occurred with design of EHW system using FPGAs while the structure of FPGA chip has been changed. As a result of this research the authors attempted to design the general EHW system that may avoid in future the problems related to significant change in available information about chips and their behaviour. We will concentrate on the system structures, where the hardware plays the dominant role. Then the high amount of documentation given by Xilinx influenced the choice of FPGA chips. This being so the paper gives some description of some intrinsic EHW solutions that are dedicated to be implemented on Xilinx FPGA.

The paper is structured as follows. The following section discusses the requirements to design intrinsic EHW system as well as introduces the recent advances in implementation of FPGA-based EHW systems. Section 3 will discuss various issues related to development of intrinsic EHW system. Sections 5 – 7 cover the architecture of FPGA that followed by introduction and analysis of 3 proposed solutions: FPGA-based Run-Time Configuration system, JBits-based Run-Time Configuration system and Multi-board functional-level Run-Time Configuration system. And finally conclusion section draws the main conclusions and identifies future work.

Manuscript received February 15, 2006. This work was supported in part by the EPSRC under grant number GR/S17178/.

E. Stomeo, C. Lambert are PhD students at School of Engineering and Design, Brunel University, West London, UK. T. Kalganova is lecture at the same university. UB8 2TR, Uxbridge, Middlesex, UK (Tel: 0044 01895 266777; e-mail: cyrille.lambert@brunel.ac.uk).

II. EHW System Requirements

In most of commercial systems such features as flexibility, scalability and implementation easiness play vital role in selection of appropriate structure. The flexibility defines whether the system accepts to evolve different kinds of problem or is focused on the evolution of some special ones. The scalability defines the capability of the designed system to be able to scale from evolution of small tasks to larger problems without significant modification in the structure. And finally, the implementation easiness that defines how easy to implement the solution. These three parameters appear to have a high priority in development of an EHW system.

Fig. 1 illustrates 4 existing types of intrinsic EHW implementation.

The first intrinsic EHW system (Fig. 1(a)) was introduced and implemented by Thompson [9]. The intrinsic EHW system consists of a Reconfigurable Hardware (HW) linked to a PC. In this case an XC6216 Xilinx FPGA was used for reconfigurable hardware part. The FPTA board introduced in [5] has the structure similar to one discussed above (Fig. 1 (a)). This board was introduced to reduce the dependency to the FPGA. In this structure the PC manages the GA, the Fitness Evaluation and the best chromosome.

The best example of the system implementation illustrated in Fig. 1(b) can be found in [10]. All the system is implemented in the same reconfigurable HW. In this case an FPGA Virtex series was used in such a way that the evolutionary process, the fitness evaluation, the best chromosome memory and the target to evolve are implemented on the same chip (FPGA). The main motivation of this work was to realise an IP EHW core. The PIG has also the structure depicted in Fig. 1(b) [4]. The LSI intrinsic chip explained in [11] is based on the system architecture shown in Fig. 1(c). Similar structure of multi-chip intrinsic EHW based on the pipelining algorithm was introduced in [12][14].

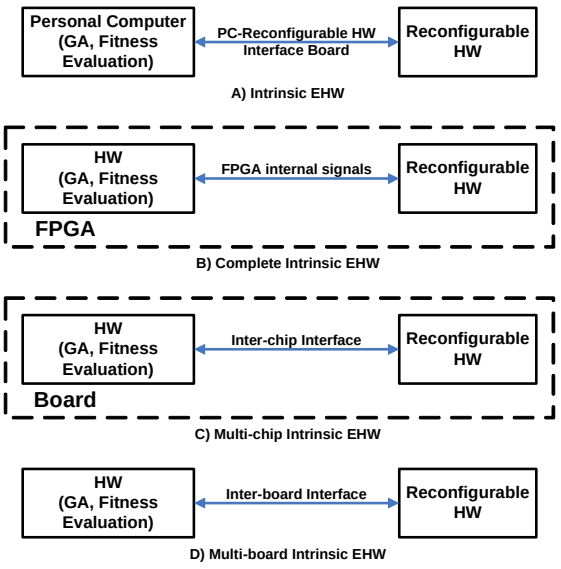

Fig. 1 Type of Implementation of Intrinsic EHW system

Finally, de Garis introduced the idea of a Multi-board intrinsic EHW with a very high number of board with the aim is to create an artificial brain [1] (Fig. 1(d)). In this paper we concentrate on analysis of intrinsic EHW systems that can be applied to design digital logic circuits. Therefore, the basic principles of CAM-Brain Machine will not be considered in this paper.

An EHW system comports several parts that are common to every evolvable system [22]: the reconfigurable system in itself, the Evolutionary Algorithm (EA), the fitness evaluation, the chromosome memory and the test patterns.

III. Key Issues in Development of EHW System

To implement a solution on chip several alternatives are available: first of all analogue or digital support. In our case we decided to work on digital support due to its simplicity during the implementation phase. Thereby our choice lies between an ASIC, a PLA, a Complex Programmable Logic Device (CPLD) or an FPGA. The choice of an ASIC involved a lot of restrictions emanating from the fact that it is "Application Specific" and due to the expensive price. Then between PLA, CPLD and FPGA it is the last one that offers the highest flexibility. A PLA offers only a configuration of the connection within the AND-array and/or OR-array or any other chosen types of planes. The possibility of fully reconfiguration of a AND/OR array has given to the CPLD a certain interest but we needed more than "AND" and "OR" gates due to initial requirements to implement the logic circuits based on any chosen functional set (that includes but not limited to AND, OR operators).

Therefore the choice of the FPGA has been in a way obvious. The Table I illustrates the specificity of each kind of platforms discussed earlier.

Therefore the aim became to develop and implement a cost-effective FPGA-based run-time configuration system. Hence a brief description of Xilinx FPGA architecture is given in following section that follows by the description of a proposed gate-level solution.

TABLE I
Platform Comparison List

	ASIC	PLA	CPLD	FPGA
"Obvious reason"	It is Application Specific so no flexibility and scalability.	The configuration acts only on the routing so no flexibility.	The configuration acts only on the routing so no flexibility.	It is the best platform in term of flexibility and scalability (due to the partial reconfiguration).
Price	Expensive, relies on the production of the ASIC.	Cheap, less than 10$US.	Cheap, less than 10$US.	From 50$US to more than 800$US relying on the series and size.
Flexibility	The lowest, specification applied.	Low, only the routing can be configure.	Limited, only the routing can be configured.	The highest, the routing and the functionality can be configured.
Functional set	Relies on the task to solve.	AND/OR	AND/OR/(XOR in option)	Flexible
Reconfiguration	Depends on the task to carry out.	The routing.	The routing.	The routing and the functionality.

IV. FPGA-based Run-Time Configuration System

The proposed FPGA-based Run-time Configuration system (RTC) is mainly divided into two parts: Evolution Strategy (ES) [13] and reconfigurable hardware (Fig. 2). The reconfigurable hardware is used as the target to evolve. It executes the desired functionalities. The fitness function calculation is also partially implemented in this module. The ES block is composed of the implementation of Evolutional Strategy itself, the fitness value evaluation, the chromosomes back-up and the communication protocol management in order to configure the evolved target. A (1+λ) ES has been chosen because the previously simulated results demonstrated the superior behaviour of this algorithm [17]. This system will allow using the smallest number of gates possible and to have a very high speed during the reconfiguration as well as for performance of the evolutionary process.

We will refer to reconfigurable hardware module as "acting module" or "target" and to "evolution strategy module" as "thinking module" due to the fact that all analysis of the circuit structure is carried out by this module.

The communication between those two modules is carried out by the SelectMAP interface provided by Xilinx. The connection between the target and the "thinking" module concerning the fitness value will be carried out by some IOBs. The fitness value is calculated inside the target but evaluated inside of the "thinking" module.

The chromosome's back-up stores the 5 chromosomes that are present in the system and the best chromosome obtained after the evaluation process. The fitness value evaluation module decides if a creation of a new generation of chromosomes is useful or not based on the fitness value received from the target. The complexity of the communication between the ES and the target is expressed by understanding and respecting the protocol of communication in order to be able to reconfigure the target from the ES. In other words it is necessary to understand the bit-stream. The protocol is quite heavy and requires a lot of parameters to manage. Once the parameters have been identified and understood by the target, the addressing of the element that constitutes the target is primordial. This is the main problem at the moment. The bit-stream is not well documented by Xilinx and it complicates and slows down the development of the FPGA-based Run-time Configuration system.

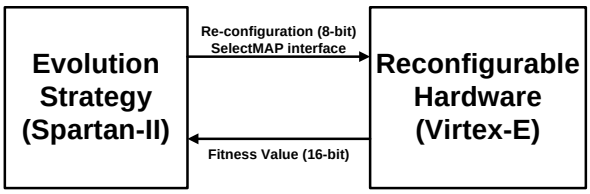

Fig. 2 FPGA-based RTC top level architecture

V. Xilinx FPGA Architecture

Although an FPGA offers the highest flexibility, its architecture is not at the very least complex. A Field Programmable Gate Array (FPGA) is a reconfigurable chip. It accepts to be reconfigured at run-time. Because of the Look-Up Tables (LUT) located in a FPGA a wide panel of possible configuration is offered. It is mainly composed of Input-Output Blocks (IOB), which are located on the periphery of the chip, a Configurable Logic Blocks (CLB) array, some Block Random Access Memory (BRAM), programmable routing and configuration circuitry. A CLB is composed of two slices, themselves composed of 2 LUT, some logic circuits and Flip-Flop (FF). In the current work the Xilinx Virtex-E FPGA will be mostly considered [15]. Its architecture is slightly different from a Virtex by the fact that there is more CLB provided and RAM available in a Virtex-E series FPGA. The BRAM columns are split and located between columns of CLB. In this work the low cost Xilinx Spartan FPGA was also considered [16] that similar architecture to a Virtex series.

The requirements to reduce as much as possible the number of gates used has imposed to work at the gate-level of the target. Therefore, the proposed solution is intended to be implemented on a Virtex-E series FPGA.

VI. Internal Organisation of a Virtex-E FPGA

To illustrate the problem discussed in the previous section it is important to expose the internal architecture of a Virtex-E. The Configuration addressing of the XCV50E is illustrated in Fig. 3. The XCV50E is composed, schematically speaking; of 5 different kinds of columns. Each of them contains 384 bits per frame [18]. A Virtex-E FPGA is divided in several columns; these columns are at the number of 35 for a XCV50E. These 5 types of column are: 1 Centre column, 24 CLB columns that contain the information for the CLB which are located in the component and for each CLB column 2 top IOB and 2 bottom IOB. 4 Block SelectRAM Interconnect columns that contain all the information on the connection of the Block SelectRAM content column which is placed side by side. 4 Block SelectRAM content and finally 2 IOB columns, one for each side, right and left. The centre column contains all the information about the configuration of the four global clock pins All this columns are divided into blocks multiples of 18 bits [18]. The following section gives a better understanding of the size of the frame.

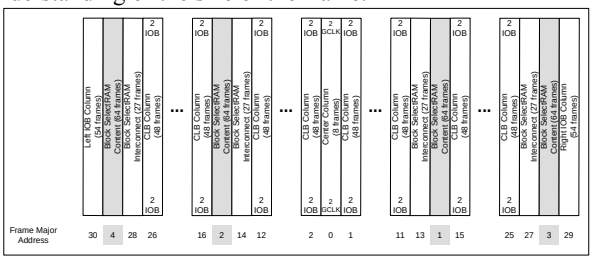

Fig. 3 Configuration addressing of a XCV50E [13]

VII. Virtex CLB Column Description

It is important to note that there are 16 rows of CLB in a XCV50E. The number of configuration bits in a frame is equal to 18*(#CLB_rows+2). So, 324 bits per frame will be used to configure the chip. But it is imposed to fit in 32-bit and some bits are added because of the pipelining implementation issues. Therefore, the number of bits is increased up to 384 bits, defined by Xilinx. A correct management and knowledge of the CLB column is the most important issue to successfully implemented the proposed architecture. This type of column is going to undergo the reconfigurations during the evolutionary process. The other columns will be fixed. Each column contains the information of some CLBs and the routing associated to these CLB. Each CLB column is composed of 48 frames. Each frame contains the information on a part of the Top IOB, the CLB configuration, CLB routing and the Bottom IOB. The term "partially" is important because, as illustrated in the Fig. 4, a frame doesn't contain all the information necessary to configure an entire column. The 48 frames, in the example, are essential to configure a column. It means to obtain a correct configuration the frames are sent one after the other one, separated by some pad bits.

A. Bit-Stream Description: CLB Column Case

The bit-stream is the name of the data packet which is sent to the target in order to configure it. Only the CLB columns are going to be subject to reconfiguration. IOB element surrounded by 16 CLB elements. A XCV50E contains a CLB array of 16 rows by 24 columns. Each element is composed of 18 bits. A frame consists of 12 words of 32 bits. It implicates to cut the elements in order to build some 32-bit words. The bit index shows where the elements are cut. For instance, the element called "CLB #1" is cut between the bit 13 and 14. So, from the bit 0 of the element "Top IOB" to the bit 13 of the element "CLB #1" it is a 32-bit word. In order to have a frame only constituted by 32-bit words some pad bits have to be added. For instance, a frame #10 (as shown in the Figure 5) starts in bit 0 of the element "CLB #16" and finishes in bit 13 of the element "Bottom IOB". So, the frame should contain 4 bits of "Bottom IOB" element. Therefore, some pad bits are required. In this case it takes 28 bits. If we count the number of 32-bit words at this level it results only 11 and [18] indicates that a XCV50E contains 12 words by frame. So, a pad word of 32 bits has to be added. At the end, 60 pad bits have been added. The pad word also called a pad frame at the end of the frames sending is described in details in [18]. Apart from the IOB blocks, a CLB column contains some "CLB" blocks, 16 in this present case. Each "CLB" block has the information on the CLB configuration by itself and the routing configuration with the other CLB blocks. The CLB configuration is the configuration of the LUT, the control logic, the flip-flops. There is definitively, a lack of information about the routing in Xilinx documentation. Based on the available information [19] one can conclude the routing between the LUT, the logic and

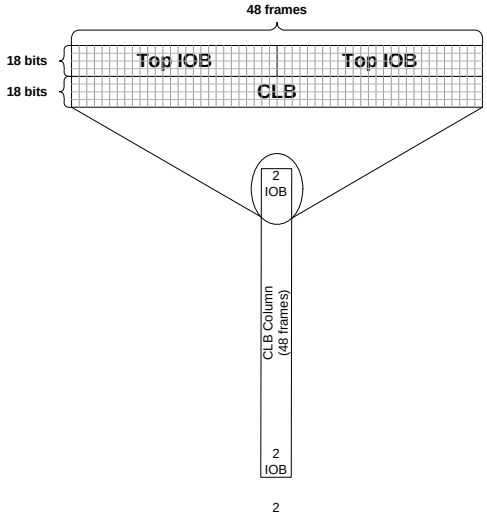

Fig. 4 CLB column detailed [22]

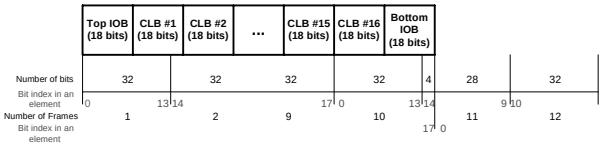

Fig. 5 CLB column in bitstream [22]

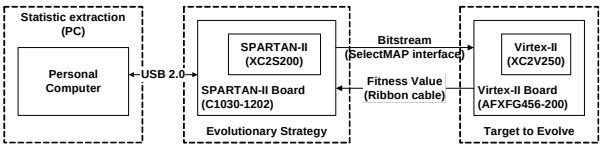

Fig. 6 FPGA-based Run-Time Configuration system architecture

the FF at gate level is managed as well as the routing between the other CLB by the way of the Switch Boxes (SB) and the Connection Boxes (CB). The CBs are dedicated to the direct communication between the CLB and also as interface with the SB. The SB allows the connection between some CLB all over the CLB array. With all these information it has been possible to design architecture of the FPGA RTC system. The system is composed of two Xilinx FPGA: a Spartan-II and a Virtex-E. The Fig. 6 demonstrates the proposed architecture of the FPGA-based RTC system. It contains two boards with FPGAs mentioned earlier. An interface board links a PC and the EHW system in order to record the results on the behaviour of the evolutionary process. Although this solution looks very attractive a point has been reached, where a lack of information kept going further. It appears that the organisation of the bit inside an element of a CLB column frame is not exposed. Indeed, from the documentation provided by Xilinx nothing permits to find this data. A reverse engineering process could be done, but it is long and legally not allowed process. Thereby, the bit-stream cannot be fully managed and a way to solve this problem has to be identified. The next section exposes one of the possible solutions that allowed solving this problem.

VIII. JBits-based Run-Time Configuration System

The problem exposed in the previous section has conducted to further research to design alternative run-time intrinsic EHW system. One of the solutions considered is based on the use of JBits software to reconfigure target. A shareware java Application Program Interface (API) called JBits provided by Xilinx; configures the target because of the data sent from the ES unit. JBits contains a set of Java classes that allow any user to configure any Xilinx FPGA from the Virtex or Virtex-II series. JBits version 3.0 is dedicated to Virtex-II series and the anterior ones (i.e. JBits 2.8) are devoted to Virtex series. The JBits is the only alternative that allows one to interact with the bit-stream. Although a read back function is provided, the structure of the bit-stream has not been revealed. Its main role takes into account the generation produced by the ES board, transforms it into configuration bit-stream and sends this one to the target. In the anterior versions of JBits (up to v2.8) other API were provided such as a tool to simulate a configuration without any hardware and to visualise the internal configuration of the target this tool is called BoardScope, JRoute was also provided and some other ones to help the development. But in the latest version of JBits no such tools are provided. This issue could significantly complicate the development. It is important to note that only the JBits v3.0 is available on the Xilinx website and it allows working on Virtex-II exclusively. Thereby a processor has to be included and more precisely a Java processor, because the JBits software runs under this language. The Slave SelectMAP configuration mode available with a Virtex-II FPGA offer a better data exchange mode. Indeed, the Virtex-II offers the possibility to use a high-speed connection through the SelectMAP interface in slave mode [20]. This solution gives a good alternative to by-pass the bit-stream problem explained in the previous section and remains operating at gate-level. Unfortunately even for this system architecture, the bit-stream data content still remains unknown to the user. Therefore this solution has not been considered in more details.

IX. Multi-Board Functional-Level Run-Time Configuration System

The idea of development of target part at functional level rather then gate level has been initially introduced by Sekanina [10][21]. Considering that analysis of gate-level implementation of EHW system revealed that the developed architecture will be strongly dependant on the available information about chip, the use of functional level implementation (even if it requires large number of logic gates) can be considered as an option. The functional level implementation has the potential to perfectly match with the original idea to have a fully hardware system, the possibility to entirely manage the bit-stream and also offers some extensions. Sekanina exposed and carried out a solution where the target to configure and the GA unit are implemented on the same FPGA.

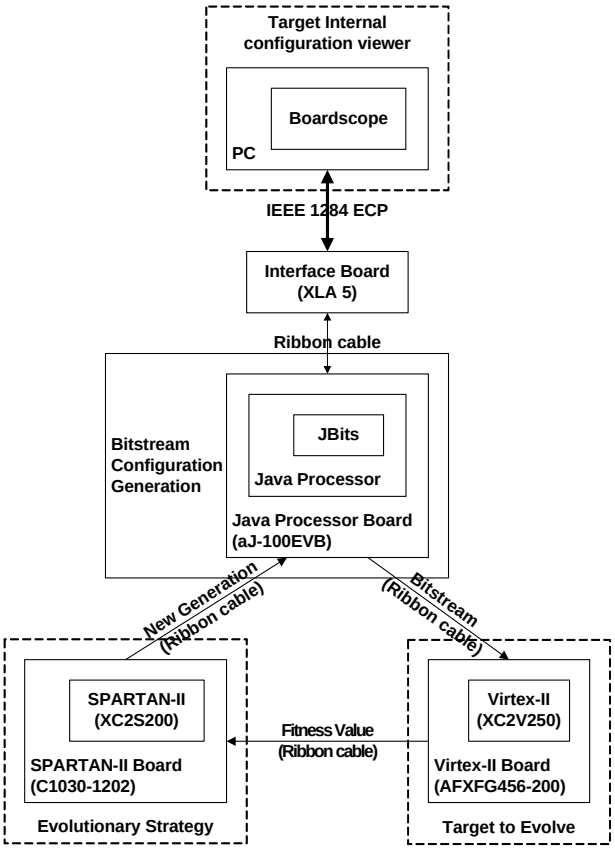

Fig. 7 JBits RTC top-level architecture

Further details of this system implementation can be found in [10]. Fig. 8 illustrates the proposed architecture of the multi-board functional-level RTC system. The system is composed of several boards. The Evolutionary Algorithm (EA), the Random Number Generator (RNG) and the fitness function evaluation are implemented on a single board. The reconfigurable hardware can be implemented either on one board or split into 5 boards, where each board implements one chromosome. The RNG produces the first population after initialisation and is used during mutation process. The fitness function evaluation compares the output results from the reconfigurable HW with the output truth tables content. The input combinations truth tables are applied to the reconfigurable HW. The content of the output combinations in the truth tables is the expected result of the evolved target. Each reconfigurable HW is composed of a Configurable Block Array (CBA) plugged with an address decoder. Each CBA is composed of 4 rows of 16 cells each, 16 inputs and 4 outputs. Each cell can be configured in order to be plugged to one of the output of the cell of the previous column or to one of the 16 inputs in the array. Only the cells of the last column can be plugged to the 4 outputs of the CBA (Fig. 9). The CBA is composed of Configurable Blocks (CB) and Switch Boxes (SB).

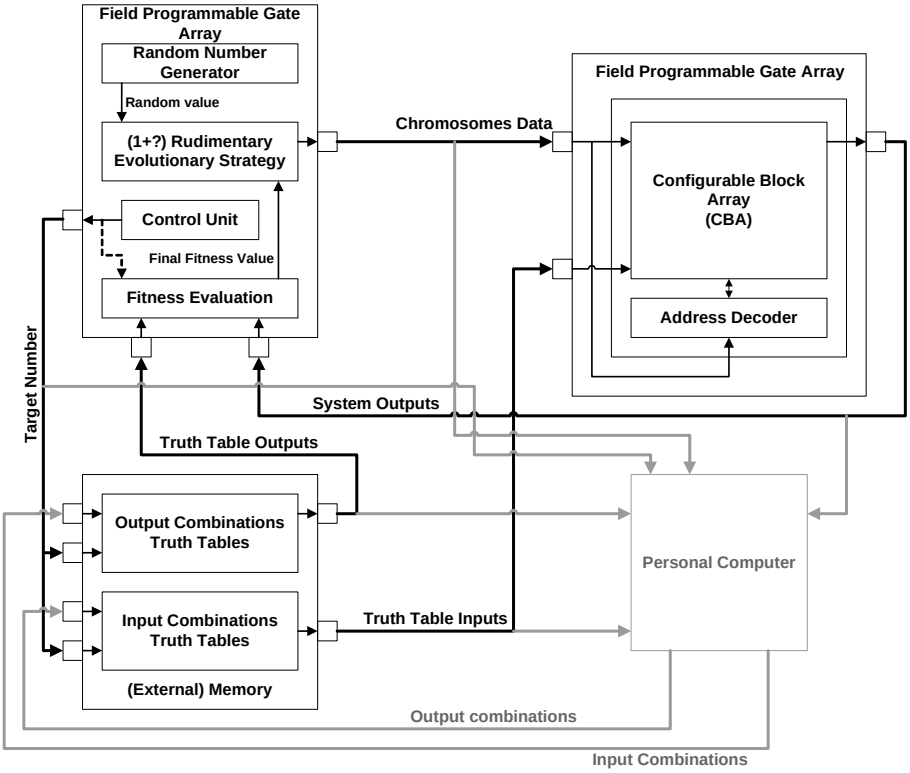

Fig. 8 Multi-board functional-level RTC top-level architecture

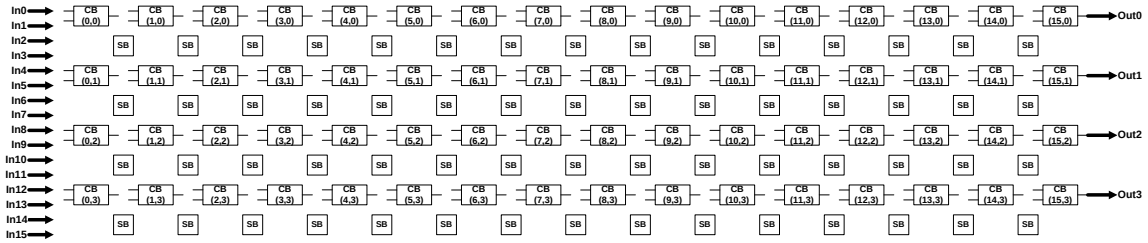

Fig. 9 Configurable Block Array

The SB allow plugging any following CB to one of the 16 inputs of the array. The CB permit plugging to one output of one of the 4 CB of the direct previous column.

To configure one cell, 20 bits are required. Therefore a chromosome consists of 1280 bits. The Virtex series FPGA accept a partial reconfiguration. So, the CBA accepts to reconfigure each cell individually due to the address decoder.

This system tends to be very flexible. Indeed, it is possible to evolve any kind of task in the limit of the EA capability. It also tends to have a good scalability, several targets are plugged, and most importantly, there is the possibility to easily increase the size of any CBA. The reduced number of signals that communicate between each FPGA makes the system easily implementable and any part can be easily updated. The first gate estimation gives as results 18094 slices that give 4524 CLB of a Virtex-II; it could be implemented on a XC2V4000 Xilinx FPGA.

X. Discussion

The use of multi-board or multi-chip system will allow one to overcome at the certain level the scalability problems and reduce the time required to re-design the system every time the new FPGA platform has been chosen or new chip has been considered. Therefore, in this paper we focus on design of multi-chip EHW systems where implementation of configuration part and evolutionary algorithm is carried out on different board or FPGA architectures. This could provide the flexibility in terms of modifications of the target in case if one desires to scale the design to larger applications.

XI. Conclusion and Future Work

The paper discusses various issues occurred in implementation of cost-effective scalable, flexible FPGA-based run time EHW system that can be easily implemented on widely manufacturing chips. Three architectures have been

considered in details: FPGA-based Run-Time Configuration system, JBits-based Run-Time Configuration system and Multi-board functional-level Run-Time Configuration system. The first two architectures can operate at gate-level at the same time as the last one is managed at function-level.

The Multi-chip FPGA-based RTC system has been considered as the cheapest alternative in terms of number of gates required to implement intrinsic EHW system. Unfortunately it has been discovered that in order to implement this system on Xilinx it is required the full understanding of bit-stream. This knowledge is required to perform the direct reconfiguration of chip from another one (in the proposed case EA-based chip). Besides of this the proposed solution strongly depends on the platform chosen and cannot be freely transferred to other chips with minor corrections. This indicates that this solution is not flexible in terms of its capability to be implemented to various FPGA platforms. Although this solution uses the smallest number of logic gates required to implement EHW system, the scalability issues remain uncertain. First of all, the system should be re-designed if one desires to implement it on larger scale chip. This includes the management of bit-stream that is crucial in EHW system implementation as it has been demonstrated earlier. JBits-based Run-Time Configuration system allows one to overcome the problem of bit-stream management that appeared to be vital during analysis of FPGA-based Run-Time Configuration system. The complexity of JBits and the lack of direct management of the configuration bit-stream are considered as the main disadvantage factors in the development of the RTC system. The Multi-board functional-level RTC system operates at the functional level. This brings automatically one of disadvantages over the other two proposed systems, since it would require a large number of logic gates to implement not only EA, but reconfigure the target as well. In this case each CLB is considered as one primitive logic gate and the additional infrastructure should be implemented in order to accommodate the implementation of high level routing. This solution provides the best flexibility in terms of its capability to extend to different size of problems with minimum implementation efforts.

The multi-board approach demonstrates the possibility to change easily the size of the target boards. This demonstrates that the system can be easily scalable to the larger problems. From another point of view the modifications in development of the possibility to change the EA just changing the suitable board increase the flexibility.

The future work will concentrate on the development of multi-board functional level RTC system and investigation of its features.

REFERENCES

[1] H. de Garis. "Evolvable Hardware: Principles and Practice". Communications of the Association for Computer Machinery (CACM Journal). August 1997.

[2] X. Yao, T. Higuchi. “Promises and challenges of evolvable hardware” IEEE Trans. Systems, Man and Cybernetics, Part C, vol. 29, Pages. 87 - 97, February 1999.

[3] D. E. Goldberg. *Genetic algorithm in search, optimization and machine learning*. Addison-Wesley Publishing Company, Incorporated, Reading, Massachusetts, 1989.

[4] N. J. Macias. “The PIG paradigm: the design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture”. Proceedings of the First NASA/DoD Workshop on, 19-21. pp: 175-180. 1999.

[5] A. Stoica, D. Keymeulen, D. Vu, R. Zebulum, I. Ferguson, T. Daud, T. Arsian, G. Xin. "Evolutionary recovery of electronic circuits from radiation induced faults". Evolutionary Computation, 2004. CEC2004. Congress on, Volume: 2, 19-23. pp: 1786-1793. Vol.2. 2004.

[6] J. Langeheine, K. Meier, J. Schemmel, M. Trefzer. “Intrinsic evolution of digital-to-analog converters using a CMOS FPTA chip”. Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on, 24-26. pp: 18-25. 2004.

[7] I. Kajitani, et al. “A gate-level EHW chip: Implementing GA operations and reconfigurable hardware on a single LSI”. (Proc. of Second International Conference on Evolvable Systems: From Biology to Hardware (ICES1998)). Springer Verlag. pp. 1-12.

[8] V. Baumgarte, F. May, A Nückel, M. Vorbach, and M. Weinhardt. ”PACT XPP - A self-Reconfigurable Data Processing Architecture”. Presented at ERSA'01, Las Vegas, NV, (c) CSREA Press. 2001.

[9] A. Thompson. "Exploring Beyond the Scope of Human Design: Automatic generation of FPGA configurations through artificial evolution." 8th Annual Advanced PLD & FPGA Conference 1998.

[10] L. Sekanina, S. Friedl. “On Routine Implementation of Virtual Evolvable Devices Using COMBO6”. In: Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware, Los Alamitos, US, ICSP. pp. 63-70, ISBN 0-7695-2145-2. 2004.

[11] M. Iwata, I. Kajitani, Y. Liu, N. Kajihara, T. Higuchi. "Implementation of a Gate-Level Evolvable Hardware Chip." Evolvable Systems: From Biology to Hardware. Lecture Notes in Computer Science 2210 (Proc. of ICES2001). Springer Verlag. pp. 38-49. 2001.

[12] G. Tufte, P. C. Haddow. "Identification of functionality during development on a virtual Sblock FPGA". Evolutionary Computation, 2003. CEC '03. The 2003 Congress on. Volume: 1. pp. 8-12. Dec. 2003.

[13] I. Rechenberg, “Evolution Strategy”, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 147-159.

[14] J. Torresen, J. W. Bakke and L. Sekanina. "Recognizing Speed Limit Sign Numbers by Evolvable Hardware." In proc. of 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII). UK. 2004.

[15] Xilinx. “Virtex-E 1.8V FPGA Complete Data Sheet”. 14/3/2003.

[16] Xilinx. “Spartan-II 2.5V FPGA Complete Data Sheet”. 9/3/2003.

[17] T. Kalganova, J.F. Miller. “Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness”. Proc. of the First NASA/DoD Workshop on Evolvable Hardware pp. 54-65.

[18] Xilinx. “XAPP151 - Virtex Series Configuration Architecture User Guide”, v1.6. 24/3/2003.

[19] W. Huang, S. Mitra, E. J. McCluskey, “Fast Run-Time Fault Location in Dependable FPGA-Based Applications”, DFT 2001.

[20] Xilinx. “Virtex-II Platform FPGA User Guide”, v1.9. pp. 206-214. 05/8/2004.

[21] L. Sekanina, “Towards Evolvable IP Cores for FPGAs”. In: Proc. of The 2003 NASA/DoD Conference on Evolvable Hardware, Los Alamitos, US, ICSP, pp. 145-154, ISBN 0-7695-1977-6. 2003.

[22] http://unit.aist.go.jp/asrc/asrc-5/en/overview.html